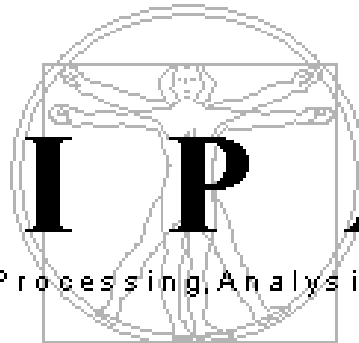


**M I P A V**

Medical Image Processing, Analysis, & Visualization



# ***User's Guide***

**Volume 2**

**ALGORITHMS**

**December 5, 2008**

**National Institutes of Health  
Center for Information Technology  
Rockville, Maryland**

December 5, 2008

Matthew McAullife, PhD [mcmatt@exchange.nih.gov](mailto:mcmatt@exchange.nih.gov)  
301-594-2432  
Building 12A, Room 2041  
National Institutes of Health  
Bethesda, Maryland 20892

*If you find a bug*, send email to [bug@mipav.cit.nih.gov](mailto:bug@mipav.cit.nih.gov). Frozen menus and JAVA exceptions dialogs are common signs. Please include as much information about what happened as you can. Please understand that we might need to get more information from you about what happened so we understand the problem.

If you have a feature idea, send an email to [wishlist@mipav.cit.nih.gov](mailto:wishlist@mipav.cit.nih.gov).



# Contents

List of Figures .....	16
List of Tables .....	28
Preface .....	29

## List of Figures 22

## List of Tables 40

## Preface 41

Scope of this guide.....	41
Who should use this guide.....	41
Skills you need to perform tasks in MIPAV .....	42
How this guide is organized .....	42
<i>Volume 1, Basics</i> .....	42
<i>Volume 2, Algorithms</i> .....	44
Where to find more information .....	44
Conventions used in this guide.....	45

## Chapter 1

### Understanding MIPAV capabilities 47

MIPAV Algorithms Overview.....	47
Introducing MIPAV .....	47
Understanding MIPAV capabilities.....	48
Developing new tools using the API .....	49

## Chapter 2

### Using MIPAV Algorithms ..... 50

Recording algorithms usage with the history feature .....	50
---	----

Introducing MIPAV algorithms.....	51
-----------------------------------	----

Cost functions used in MIPAV algorithms .....	63
---	----

<i>Background</i> .....	63
-------------------------	----

<i>Powell algorithm</i> .....	64
-------------------------------	----

<i>Correlation ratio</i> .....	64
--------------------------------	----

Image types .....	65
-------------------	----

<i>Laplacian Zero-Crossing</i> .....	65
--------------------------------------	----

<i>Gradient Magnitude</i> .....	66
---------------------------------	----

<i>Least Squares</i> .....	66
----------------------------	----

Image types .....	67
-------------------	----

<i>Normalized cross correlation</i> .....	67
---	----

Normalized cross correlation .....	68
------------------------------------	----

Image types .....	69
-------------------	----

<i>Normalized mutual information</i> .....	70
--	----

Image types .....	70
-------------------	----

<i>Degrees of freedom</i> .....	71
---------------------------------	----

<i>References</i> .....	72
-------------------------	----

Interpolation methods used in MIPAV.....	74
--	----

<i>Bilinear Interpolation</i> .....	74
-------------------------------------	----

<i>Trilinear Interpolation</i> .....	75
--------------------------------------	----

<i>B-spline basis interpolations</i> .....	76
--	----

<i>Lagrange interpolations</i> .....	78
--------------------------------------	----

<i>Gaussian</i> .....	80
-----------------------	----

<i>Windowed sinc</i> .....	82
----------------------------	----

<i>References</i> .....	84
-------------------------	----

Autocorrelation Coefficients .....	85
------------------------------------	----

<i>Background</i> .....	85
-------------------------	----

Image types .....	86
References.....	87
Applying the AutoCorrelation Coefficients algorithm.....	87
<b>Autocovariance Coefficients .....</b>	<b>88</b>
Background .....	88
Notes.....	90
Image types .....	91
References.....	91
Applying the AutoCovariance Coefficients algorithm .....	92
<b>Barrel Distortion Correction .....</b>	<b>93</b>
Background .....	93
Correcting pincushion and/or barrel distortion .....	94
Examples .....	94
References.....	95
Image types .....	96
Applying barrel distortion correction algorithm.....	96
<b>B-Spline Automatic Registration .....</b>	<b>98</b>
Background .....	98
Error measures .....	102
Least squares measure.....	103
Correlation ratio measure .....	103
Normalized mutual information measure .....	104
Improving performance .....	105
Initial placement of control points for identity map .....	106
Composition for two-pass registration.....	108
Deformation .....	109
Detect folding .....	110
Support for color images .....	112
Support for 2.5D images .....	112
Examples of B-spline registration.....	113
Image types .....	114
User Dialogs in MIPAV .....	114
Transformation options .....	124
Output files.....	126
Class implementation for MIPAV .....	126
<b>DTI Create List File.....</b>	<b>128</b>
Outline of the method .....	129
List file.....	129
Path file.....	129
Example 1: a path file structure .....	129
B-matrix file for DICOM image datasets .....	133
If the Registration option is activated .....	134
Running the DTI Create List File utility.....	136

---

For DICOM datasets .....	136
<b>DTI Color Display .....</b>	<b>140</b>
Background .....	140
Antipodal Symmetry of the Eigenvectors of $D$ .....	141
No Symmetry .....	143
Rotational Symmetry .....	143
Mirror Symmetry .....	144
Absolute Value .....	144
Color circles .....	145
Building color circles .....	146
Color processing .....	146
Color processing heuristic parameters .....	146
Blue, red and green shifts .....	149
And finally, it performs the green shift: .....	149
Gamma correction .....	150
Importance of midsagittal alignment .....	150
Image Types .....	150
References .....	151
Applying DTI Color Display .....	151
<b>Edge Detection: Zero X Laplacian .....</b>	<b>154</b>
Background .....	154
Image types .....	157
References .....	157
Applying the Zero X Laplacian algorithm .....	157
<b>Edge Detection: Zero X Non-Maximum Suppression .....</b>	<b>159</b>
Background .....	159
Image types .....	163
References .....	163
Applying the Zero X Non-Maximum Suppression algorithm .....	164
<b>Extract Brain: Extract Brain Surface (BET) .....</b>	<b>167</b>
Background .....	167
Step 1, Estimating image parameters .....	167
Step 2, Selecting an initial closed mesh .....	168
Step 3, Evolution of the mesh .....	171
Step 4, Selecting brain voxels .....	174
Image types .....	175
Special notes .....	176
Reference .....	176
Applying the Extract Brain Surface algorithm .....	176
<b>Extract Brain: Extract Brain Surface (BSE) .....</b>	<b>179</b>
Background .....	179

<i>Step 1, Filtering the image to remove irregularities .....</i>	179
<i>Step 2, Detecting edges in the image .....</i>	179
<i>Step 3, Performing morphological erosions and brain isolation ..</i>	180
<i>Step 4, Performing surface cleanup and image masking .....</i>	180
<i>Selecting parameters .....</i>	180
Edge detection kernel size parameter .....	181
Image types .....	181
References.....	181
Applying the Extract Brain Surface (BSE) algorithm.....	182
<b>Face Anonymizer (BET) .....</b>	<b>186</b>
Background .....	186
Step 1, Extract the brain.....	186
Step 2: Buffer the brain.....	187
Step 3: Define the face.....	187
Step 4: Extract the face.....	187
Step 5: Smooth the face.....	187
Image types .....	188
Notes.....	188
<b>Reference:.....</b>	<b>188</b>
Applying the Face Anonymizer Algorithm .....	188
<b>Fast Fourier Transformation (FFT).....</b>	<b>191</b>
Background .....	192
Finite Impulse Response (FIR) filters with Hamming windows ...	193
Gaussian and Butterworth filters .....	194
Butterworth 2D and 3D filters .....	194
Butterworth low-pass, high-pass, band-pass, and band-	
stop filters.....	194
Gaussian low- and high-pass filters .....	195
Image types .....	196
Special notes .....	196
References.....	197
Applying the FFT algorithm .....	197
Step 1: Converting dataset from spatial to fourier domain using	
the forward FFT command .....	198
Step 2: Enhancing and attenuating frequency components	
using Frequency Filter .....	199
Optional Step: Attenuating frequency components using	
the Paint tool .....	201
Step 3: Reconverting the image to the spatial domain .....	203
<b>Filters (Frequency) .....</b>	<b>204</b>
Background .....	204
Gaussian and Butterworth filters .....	207
Butterworth 2D and 3D filters .....	207
Butterworth low-pass, high-pass, band-pass, and band-	
stop filters.....	207

Gaussian low- and high-pass filters .....	208
<i>Windowed Finite Impulse Response</i> .....	209
References.....	211
Image types .....	212
<i>Applying the Frequency Filter algorithm</i> .....	212
<b>Filters (Spatial): Adaptive Noise Reduction .....</b>	<b>214</b>
Background .....	214
Image types .....	216
Special notes .....	216
References.....	217
<i>Applying the Adaptive Noise Reduction algorithm</i> .....	217
<b>Filters (Spatial): Adaptive Path Smooth.....</b>	<b>219</b>
Background .....	219
Image types .....	223
Special notes .....	223
Reference.....	223
<i>Applying the Adaptive Path Smooth algorithm</i> .....	223
<b>Filters (Spatial): Anisotropic Diffusion .....</b>	<b>226</b>
Background .....	227
Image types .....	228
Special notes .....	228
References.....	228
<i>Applying the Anisotropic Diffusion algorithm</i> .....	229
<b>Filters (Spatial): Coherence-Enhancing Diffusion .....</b>	<b>231</b>
Background .....	232
Image types .....	234
Special notes .....	234
References.....	235
<i>Applying the Coherence-Enhancing Diffusion algorithm</i> .....	235
<b>Filters (Spatial): Gaussian Blur .....</b>	<b>237</b>
Background .....	237
Image types .....	240
Special notes .....	240
References.....	241
<i>Applying the Gaussian Blur algorithm</i> .....	241
<b>Filters (Spatial): Gradient Magnitude .....</b>	<b>243</b>
Background .....	243
Image types .....	244
Special notes .....	245
References.....	245
<i>Applying the Gradient Magnitude algorithm</i> .....	245



---

<b>Filters (Spatial): Haralick Texture .....</b>	<b>247</b>
<i>Background .....</i>	247
<i>Outline of the algorithm.....</i>	247
<i>Image types .....</i>	249
<i>References.....</i>	249
<i>Applying the algorithm .....</i>	250
<i>Calculation of the 12 texture features .....</i>	251
<b>Filters (Spatial): Laplacian.....</b>	<b>257</b>
<i>Background .....</i>	257
<i>Image types .....</i>	258
<i>Special notes .....</i>	258
<i>References.....</i>	259
<i>Applying the Laplacian algorithm .....</i>	259
<b>Filters (Spatial): Local Normalization TBD.....</b>	<b>261</b>
<i>Background .....</i>	261
<i>Image types .....</i>	261
<i>Special notes .....</i>	261
<i>References.....</i>	261
<i>Applying the Local Normalization algorithm.....</i>	261
<b>Filters (Spatial): Mean .....</b>	<b>262</b>
<i>Background .....</i>	262
<i>Image types .....</i>	265
<i>Special notes .....</i>	265
<i>References.....</i>	265
<i>Applying the Mean algorithm.....</i>	265
<b>Filters (Spatial): Median.....</b>	<b>267</b>
<i>Background .....</i>	267
<i>References.....</i>	270
<i>Image types .....</i>	270
<i>Special notes .....</i>	270
<i>Applying the Median Filter algorithm .....</i>	271
<b>Filters (Spatial): Mode .....</b>	<b>274</b>
<i>Background .....</i>	274
<i>Image types .....</i>	276
<i>Notes.....</i>	276
<i>References.....</i>	276
<i>Applying the Mode algorithm.....</i>	276
<b>Filters (Spatial): Nonlinear Noise Reduction .....</b>	<b>279</b>
<i>Background .....</i>	279

---

Image types .....	281
Special notes .....	281
References.....	281
Applying the Nonlinear Noise Reduction algorithm.....	283
<b>Filters (Spatial): Nonmaximum Suppression .....</b>	<b>285</b>
Background .....	285
Image types .....	289
Special notes .....	289
References.....	289
Applying the Nonmaximum Suppression algorithm.....	290
<b>Filters (Spatial): Regularized Isotropic (Nonlinear) Diffusion... ..</b>	<b>292</b>
Background .....	293
Image types .....	295
Special notes .....	295
References.....	295
Applying the Regularized Isotropic (Nonlinear) Diffusion algorithm .....	296
<b>Filters (Spatial): Slice Averaging .....</b>	<b>297</b>
Background .....	297
Image types .....	300
Special notes .....	301
References.....	301
Applying the Slice Averaging algorithm.....	301
<b>Filters (Spatial): Unsharp Mask .....</b>	<b>303</b>
Background .....	303
Special notes .....	304
References.....	304
Image types .....	305
Applying the Unsharp Mask algorithm.....	305
<b>Filters (Wavelet): De-noising BLS GSM.....</b>	<b>307</b>
Background .....	307
Outline of the algorithm.....	308
Steerable pyramid .....	309
References.....	310
Image types .....	310
Applying the Wavelet BLS GSM algorithm .....	311
<b>Filters (Wavelet): Thresholding .....</b>	<b>314</b>
Background .....	314

---

<i>Using wavelets to reduce the noise in images</i> .....	315
<i>Wavelet thresholding</i> .....	317
<i>Zero padding</i> .....	319
<i>Outline of the algorithm</i> .....	319
<i>References</i> .....	319
<i>Image types</i> .....	321
<i>Applying the Wavelet Thresholding algorithm</i> .....	322
<b>Fuzzy C-Means: Multispectral and Single Channel Algorithms</b> .	<b>325</b>
<i>Hard segmentation</i> .....	325
<i>Soft, or fuzzy, segmentation</i> .....	325
<i>Background</i> .....	325
<i>Adding images to the list</i> .....	328
<i>Removing images from the list</i> .....	328
<i>Image types</i> .....	329
<i>Special notes</i> .....	329
<i>References</i> .....	329
<i>Applying the Fuzzy C-means algorithm</i> .....	330
<i>On single images</i> .....	331
<i>On multispectral images</i> .....	332
<b>2D Histogram</b> .....	<b>335</b>
<i>Background</i> .....	335
<i>Image types</i> .....	336
<i>Notes</i> .....	336
<i>Applying the 2D Histogram algorithm to grayscale images</i> .....	337
<i>Applying the 2D Histogram algorithm to color images</i> .....	338
<b>Cumulative Histogram</b> .....	<b>340</b>
<i>Background</i> .....	340
<i>Examples</i> .....	340
<i>Notes</i> .....	343
<i>References</i> .....	343
<i>Image types</i> .....	343
<i>Applying Cumulative Histogram algorithm to images</i> .....	343
<i>To apply Cumulative Histogram to grayscale images:</i> .....	343
<i>To apply Cumulative Histogram to RGB images</i> .....	344
<b>Histogram Equalization: Regional Adaptive</b> .....	<b>345</b>
<i>Background</i> .....	345
<i>Image types</i> .....	349
<i>Special notes</i> .....	349
<i>References</i> .....	350
<i>Applying the Histogram Equalization: Regional Adaptive algorithm</i> .....	352
<b>Histogram Equalization: Neighborhood Adaptive</b> .....	<b>354</b>

---

<i>Background</i> .....	354
<i>Image types</i> .....	362
<i>Special notes</i> .....	362
<i>References</i> .....	362
<i>Applying the Histogram Equalization: Neighborhood Adaptive algorithm</i> .....	363
<b>Histogram Matching</b> .....	<b>366</b>
<i>Background</i> .....	366
<i>References</i> .....	369
<i>Image types</i> .....	369
<i>Applying the Histogram Matching algorithm</i> .....	369
<b>Histogram summary</b> .....	<b>371</b>
<i>Applying Histogram Summary to the image</i> .....	371
The histogram summary table includes the following four columns, see Figure 1: .....	371
<b>Manual 2D Series</b> .....	<b>374</b>
<i>Background</i> .....	374
<i>Image types</i> .....	374
<i>Special notes</i> .....	375
<i>References</i> .....	375
<i>Applying the Manual 2D Series algorithm on 2D images</i> ....	375
To run this algorithm, complete the following steps: .....	375
Options available via the dialog menu .....	376
<i>Blended tab</i> .....	376
<i>Changing displaying proportion</i> .....	377
<i>The Blended tab options:</i> .....	377
<i>Moving and rotating the adjusted image</i> .....	379
<i>Moving the adjusted image</i> .....	380
<i>Moving the adjusted image with the direction arrows</i> .....	380
<i>Rotating the adjusted image</i> .....	381
<i>Dual tab</i> .....	382
<i>The Dual tab offers the following options</i> .....	383
<i>Using the image registration algorithms</i> .....	385
<i>Delineating the landmark points</i> .....	385
<i>Applying the image registration algorithms</i> .....	386
<i>making manual reconcilements to the adjusted image</i> .....	387
<i>Using LUT to arrange the images</i> .....	389
<i>Applying the Manual 2D Series algorithm to 2.5D images</i> ..	391
<i>Applying the Manual 2D Series algorithm to 3D images</i> ....	393
<b>Extract Surface (Marching Cubes)</b> .....	<b>395</b>
<i>Background</i> .....	395

References.....	398
Image types .....	399
Output Files.....	399
Applying the Extract Surface Algorithm.....	399
To run the algorithm, complete the following steps: .....	399
Examples.....	401
<b>Mask .....</b>	<b>407</b>
Background .....	407
Image types .....	408
Special notes .....	408
References.....	408
Applying the Mask algorithm .....	408
<b>Microscopy: Colocalization Orthogonal Regression .....</b>	<b>411</b>
Background .....	412
Images types .....	418
Special notes .....	418
References.....	419
Applying the Microscopy: Colocalization Orthogonal Regression algorithm .....	420
Colocalization histogram window.....	424
<b>Microscopy: Blind Deconvolution .....</b>	<b>429</b>
Background .....	429
Computing convolution .....	431
PSF constraints.....	431
Image types .....	432
Notes.....	432
References.....	433
Useful links .....	433
Applying the Maximum Likelihood Iterative Blind Deconvolution algorithm.....	434
Selecting the algorithm parameters .....	436
<b>Microscopy: FRAP (Fluorescence Recovery After Photobleaching) ..</b>	<b>439</b>
Background .....	439
Image types .....	452
Special notes .....	452
References.....	452
Applying the FRAP algorithm .....	453
<b>Microscopy: Colocalization Orthogonal Regression (FRET)—     Acceptor Photobleaching .....</b>	<b>468</b>
Background .....	468

Image types .....	470
Notes .....	470
References .....	471
Applying the FRET algorithm .....	471
<b>Microscopy: Fluorescent Resonance Energy Transfer (FRET) Bleed Through and Efficiency .....</b>	<b>477</b>
Background .....	477
FRET Bleed Through Algorithm .....	483
Performing the Acceptor dye only run .....	484
<b>Performing the Donor dye only run .....</b>	<b>484</b>
Moving to FRET Efficiency .....	484
Applying the FRET Bleed Through algorithm .....	485
Performing the acceptor dye only run .....	485
Performing the donor dye only run .....	491
FRET Efficiency algorithm .....	497
Applying the FRET Efficiency algorithm .....	498
Image Types .....	502
Notes .....	502
References .....	503
<b>Microscopy (Restoration): Computational Optical Sectional Microscopy (COSM) .....</b>	<b>504</b>
Background .....	504
Adjusting the speed of the processing .....	507
References .....	508
Image types .....	509
Applying the X_COSM algorithm .....	509
<b>Midsagittal Line Alignment .....</b>	<b>513</b>
Background .....	513
Outline of the method: .....	514
Iterative method used to register original image against flipped .....	514
For more information of which methods were used to .....	515
Image types .....	516
Running the Midsagittal Line Alignment algorithm .....	516
<b>Morphology .....</b>	<b>518</b>
Introduction to the basic morphological operations .....	518
data types .....	519
Background Distance map .....	519
Applying Background Distance map .....	520
Close .....	521
Applying the closing .....	522
Delete Objects .....	524
Applying the algorithm .....	524
Dilate .....	525

<i>Applying the dilation</i> .....	526
<i>Distance Map</i> .....	528
<i>Applying distance map</i> .....	528
<i>Erode</i> .....	529
<i>Applying the erosion</i> .....	529
<i>Evaluate Segmentation</i> .....	531
<i>Applying evaluate segmentation</i> .....	531
<i>Fill holes</i> .....	533
<i>Applying fill holes</i> .....	533
<i>Find Edges</i> .....	534
<i>Applying find edges</i> .....	535
<i>ID objects</i> .....	535
<i>Applying id objects</i> .....	536
<i>Morphological filter</i> .....	537
<i>Applying Morphological filter</i> .....	539
<i>Open</i> .....	540
<i>Applying the opening</i> .....	541
<i>Particle analysis</i> .....	543
<i>Image Types</i> .....	545
<i>Applying particle analysis</i> .....	545
<i>Skeletonize</i> .....	547
<i>Applying the algorithm</i> .....	548
<i>Skeletonize 3D pot field</i> .....	548
<i>Algorithm parameters</i> .....	550
<i>Applying Skeletonize 3D</i> .....	552
<i>Algorithm notes</i> .....	553
<i>Skeletonize 3D Pot references</i> .....	555
<i>Ultimate erode</i> .....	556
<i>Image Types</i> .....	556
<i>Applying ultimate erode</i> .....	556
<b>Muscle Segmentation</b> .....	<b>558</b>
<i>Background</i> .....	558
<i>Outline of the method</i> .....	558
<i>Water phantom</i> .....	559
<i>Water calibration</i> .....	560
<i>LiveWire</i> .....	560
<i>How LiveWire works</i> .....	561
<i>Level set tool</i> .....	564
<i>Muscle Segmentation steps</i> .....	565

<i>Skin identification</i> .....	565
<i>Tissue Classification</i> .....	565
<i>Fasceal detection and subcutaneous fat detection</i> .....	566
<i>Bone and bone marrow identification</i> .....	566
<i>Threshold classification and quantification</i> .....	566
<i>Quadriceps identification, classification, and quantification</i> .....	567
<i>Hamstring and Sartorius identification, classification, and quantification</i> .....	567
<i>Results</i> .....	567
<i>Image types</i> .....	568
<i>References</i> .....	569
<b>Applying the Muscle Segmentation method</b> .....	569
<i>VOI selection</i> .....	570
<i>Thigh</i> .....	571
<i>Bone</i> .....	571
<i>Muscles</i> .....	572
<i>Saving segmentation VOIs</i> .....	572
<i>Analysis</i> .....	573
<i>To show/hide LUT:</i> .....	574
<b>Abdomen Segmentation</b> .....	576
<i>Background</i> .....	576
<i>Smoothing contours</i> .....	578
<i>References</i> .....	578
<b>Applying the method for the abdomen segmentation</b> .....	579
<i>Abdomen tab</i> .....	579
<i>Modifying VOIs</i> .....	580
<i>Tissue tab</i> .....	582
<i>Muscles tab</i> .....	583
<i>VOI tab</i> .....	584
<i>VOI tab warnings</i> .....	584
<i>Where VOIs are stored</i> .....	586
<i>Changing color for VOIs</i> .....	587
<i>Manual segmentation</i> .....	588
<i>MIPAV automatic segmentation</i> .....	590
<i>Using the MIPAV VOI statistics generator</i> .....	591
<i>Analysis tab</i> .....	592
<i>Saving the statistics as a PDF file</i> .....	593
<i>To show and hide LUT</i> .....	595
<b>Optimized Automatic Registration 3D</b> .....	596
<i>Background</i> .....	596
<i>Image types</i> .....	597
<i>Outline of the method</i> .....	598
Pre-optimization steps.....	598
Optimization steps.....	598
Post optimization.....	599
<i>Pre Optimization</i> .....	599
Blurring.....	599



Resampling.....	599
<i>Calculating the center of mass</i> .....	600
Process .....	601
levelEight optimization .....	601
Optimization .....	601
levelFour optimization .....	603
levelTwo .....	604
levelOne.....	604
What is saved .....	604
Additional registration parameters .....	605
<i>Resampling Interpolation</i> .....	605
<i>Cost Functions</i> .....	605
Powell algorithm.....	606
Degrees of freedom .....	606
References.....	606
<i>Applying the AlgorithmOAR 3D</i> .....	607
To run the algorithm, complete the following steps: .....	607
<i>Applying the Constrained AlgorithmOAR 3D</i> .....	608
To run the algorithm, complete the following steps: .....	608
<i>Optimized Automatic Registration dialog box options</i> .....	609
<i>Advanced OAR settings for Constrained Optimized Automatic Registration 3D</i> .....	612
Registration: Patient Position (DICOM) .....	614
<i>Background</i> .....	614
<i>Outline of the method</i> .....	615
<i>Image types</i> .....	615
<i>Special notes</i> .....	615
References.....	616
<i>Applying the Patient Position (DICOM) algorithm</i> .....	616
Registration: Landmark—Least Squares .....	617
<i>Background</i> .....	617
<i>Image types</i> .....	619
<i>Special notes</i> .....	620
Reference.....	620
<i>Applying the Landmark—Least Squares Registration algorithm</i>	621
Registration: Landmark—TPSpline .....	623
<i>Background</i> .....	623
<i>Image types</i> .....	624
<i>Special notes</i> .....	624
References.....	624
<i>Applying the Registration: Landmark—TPSpline algorithm</i> .	625
Registration: Manual 2D Series .....	627
<i>Background</i> .....	627

---

<i>Image types</i> .....	629
<i>Special notes</i> .....	629
<i>References</i> .....	629
<i>Applying the Manual 2D Series algorithm on 2D images</i> ....	630
<i>Making manual adjustments to the image to be registered</i>	632
<i>Moving the adjusted image in any direction</i> .....	635
<i>Moving the adjusted image with the direction arrows</i> .....	636
<i>Rotating the adjusted image</i> .....	636
<i>Applying the Manual 2D Series algorithm on 2.5 images</i> ...	643
<b>Mosaic Registration</b> .....	<b>645</b>
<i>Background</i> .....	645
<i>Outline of the method</i> .....	645
Pre-registration steps.....	645
Manual mouse based alignment.....	645
Optimized Automatic Registration 2D .....	646
Forming the mosaic image.....	646
<i>Opening two images</i> .....	646
<i>Manual mouse-based alignment</i> .....	646
Rotation .....	647
Translation .....	647
Scale .....	647
Storing the transformation parameters .....	648
<i>Registering the reference and tile images</i> .....	648
<i>Backup Mosaic</i> .....	649
<i>Blending the tile image with the reference image</i> .....	649
<i>Brute Force</i> .....	650
<i>Using the Optimized Automatic Registration 2D algorithm</i> .	650
<i>Background</i> .....	650
<i>Outline of the method</i> .....	652
Pre-Optimization .....	652
Blurring.....	652
Calculating the center of mass .....	652
Resampling.....	653
Optimization .....	653
Post optimization.....	653
<i>AlgorithmOAR2D Optimization Steps</i> .....	654
What is saved .....	656
<i>Resampling Interpolation</i> .....	657
<i>Cost functions</i> .....	657
Powell method .....	657
<i>Output</i> .....	658
<i>Image types</i> .....	658
<i>Running the Mosaic Registration method</i> .....	658
<i>The Mosaic Registration dialog box options</i> .....	660
<i>Optimized Automatic Registration 2D dialog box options</i> .....	661
<i>Brute Force Settings</i> .....	663
<b>Registration: Display Pixel Similarity Cost Functions</b> .....	<b>664</b>

---

<i>Background</i> .....	664
<i>Image types</i> .....	664
<i>Applying Display Pixel Similarity Cost Function</i> .....	665
<b>Registration: Time Series Optimized Automatic Registration...</b>	<b>667</b>
<i>Background</i> .....	667
<i>Registering volumes</i> .....	667
<i>Choosing a cost function</i> .....	669
Normalized cross correlation .....	669
Correlation ratio .....	670
Least squares .....	671
Normalized mutual information.....	671
<i>Completing the dialog box</i> .....	672
<i>Image types</i> .....	672
<i>Special notes</i> .....	673
<i>Reference</i> .....	673
<i>Applying the Time Series Optimized Automatic Registration algorithm</i> .....	673
<b>Reslice—Isotropic Voxels .....</b>	<b>676</b>
<i>Background</i> .....	676
<i>Linear interpolation</i> .....	676
<i>Cubic convolution interpolation</i> .....	676
<i>Cubic B-spline interpolation</i> .....	677
<i>References</i> .....	677
<i>Image types</i> .....	678
<i>Applying the Reslice—Isotropic Voxels algorithm</i> .....	678
<b>Registration: VOI Landmark .....</b>	<b>679</b>
<i>Background</i> .....	679
<i>Template matching registration technique</i> .....	679
<i>As a similarity measure, the method uses Gradient Magnitude</i> ..	680
<i>Techniques for searching a global minimum</i> .....	680
<i>References</i> .....	682
<i>Image types</i> .....	683
<i>Applying the VOI Landmark algorithm</i> .....	683
<b>Shading Correction: Inhomogeneity N3 Correction.....</b>	<b>687</b>
<i>Background</i> .....	687
<i>How the N3 method works</i> .....	688
Mapping from v to f .....	689
Algorithm flow chart .....	689
<i>References</i> .....	696
<i>Image Types</i> .....	696
<i>Applying the Inhomogeneity N3 algorithm</i> .....	696
Choosing good parameters for N3 Algorithm .....	697
<b>Standard Deviation Threshold .....</b>	<b>700</b>

<i>Background</i> .....	700
<i>Image types</i> .....	701
<i>Applying the algorithm to grayscale images</i> .....	701
<i>The Threshold Using Standard Deviation dialog box options for grayscale images:</i> .....	702
<i>Applying the algorithm to RGB images</i> .....	703
<i>The Threshold Using Standard Deviation dialog box options for RGB images:</i> .....	706
<b>Subsampling images</b> .....	<b>708</b>
<b>Threshold</b> .....	<b>711</b>
<i>Background</i> .....	711
<i>Image types</i> .....	711
<i>Applying the Threshold algorithm</i> .....	711
<i>The Threshold dialog box options:</i> .....	712
<b>Transform</b> .....	<b>714</b>
<i>Background</i> .....	714
<i>Transforming images</i> .....	714
The Transform tab options.....	714
<i>Resampling images</i> .....	718
The Resample tab options.....	718
<i>Image types</i> .....	720
<i>References</i> .....	720
<i>Output</i> .....	720
<i>Applying the Transform/Resample algorithm</i> .....	720
To transform an image: .....	720
To resample an image: .....	721
<b>Transform: Conformal Mapping Algorithms</b> .....	<b>723</b>
<i>Background</i> .....	723
<i>Circular Sector to Rectangle</i> .....	723
<i>Transformation: Circle to Rectangle</i> .....	728
<i>Transformation: Ellipse to Circle</i> .....	729
<i>Transformation: Nearly Circular Region to Circle</i> .....	732
<i>References</i> .....	733
<i>Applying the Circular Sector to Rectangle algorithm</i> .....	733
<i>Image types</i> .....	733
<i>Running the algorithm</i> .....	734
<i>Applying the Circle to Rectangle algorithm</i> .....	735
<i>Image types</i> .....	735
<i>Running the algorithm</i> .....	735
<i>Applying the Ellipse to Circle algorithm</i> .....	736
<i>Image types</i> .....	736
<i>Running the algorithm</i> .....	736
<i>Applying the Nearly Circle to Circle algorithm</i> .....	737

---

Image types .....	737
Running the algorithm.....	737
<b>Transform Nonlinear .....</b>	<b>738</b>
Background .....	738
Image types .....	738
Applying the Transform Nonlinear algorithm.....	738
To run the algorithm, complete the following steps: .....	738
<b>Transform to power of 2.....</b>	<b>741</b>
Background .....	741
Image types .....	741
Applying the Transform to Power of 2 Algorithm .....	742
To run the algorithm, complete the following steps: .....	742
<b>Watershed .....</b>	<b>743</b>
Background .....	743
References.....	745
Image types .....	745
Applying the Watershed algorithm .....	746
<b>Glossary</b>	<b>748</b>



# List of Figures

Figure 1. The plots of the Correlation Ratio cost function versus some of the individual parameter values. In each plot (A, B, C, and D), a single parameter is varied while the other are kept constant..... 64

Figure 1. The red dots show the known pixels and the green dot is the pixel we wish to interpolate ..... 74

Figure 2. Trilinear interpolation ..... 76

Figure 3. B-spline interpolation ..... 78

Figure 4. Lagrange third-order interpolation,  $N = 4$ : (A) - Kernel, (B) - Magnitude of Fourier transform, (C) - Logarithmic plot of magnitude. Lagrange fourth-order interpolation,  $N = 5$ , (D) -Kernel, (E) - Magnitude of Fourier transform, (F) - Logarithmic plot of magnitude..... 80

Figure 5. 1-D Gaussian distribution with mean 0 and sigma=1 ..... 81

Figure 6. Discrete approximation to Gaussian function with sigma=1.0..... 82

Figure 7.  $T y = \text{sinc}(a)$ , where  $a = px$ ,  $x \in [-30, 30]$ ; a sample Hann function and its frequency response. Here,  $N$  represents the width, in samples, of a discrete-time window function. Typically, it is an integer power-of-2, such as  $2^{10} = 1024$ . ..... 84

Figure 1. Visualization of ACF ..... 86

Figure 2. Calculating autocorrelation for 2D grayscale image ..... 87

Figure 3. AutoCorrelation Coefficients dialog box..... 87

Figure 1. Calculating auto-covariance for a 2D grayscale image ..... 91

---

Figure 2. AutoCovariance Coefficients dialog box .....	92
Figure 1. Illustration of barrel distortion .....	93
Figure 2. Barrel distortion correction .....	94
Figure 3. Barrel/pincushion distortion correction dialog box.....	96
Figure 1. The input source image (A), the input target image (B) and the output registered image (C). Note that images (A) and (B) are both the images of the same brain. That's why it shows a very good overall fitting. Results may vary, if you take different brain images. ....	113
Figure 2. The deformation image and its lookup table. The dark areas on the deformation image are related to the areas on images (A) and (B) which did not perform a deformation during the registration or the deformation was relatively small. The light areas are related to the areas on images (A) and (B) which perform a bigger deformation. ....	114
Figure 3. The choice of 2D/3D or 2.5D B-spline registration is made by selecting the appropriate menu item from the Algorithms: Registration submenu .....	115
Figure 4. B-Spline Single - Pass Automatic Registration 2D/3D dialog box .....	116
Figure 5. B-Spline Two-Pass Automatic Registration 2D/3D dialog box .....	118
Figure 6. B-Spline Single-Pass Automatic Registration 2.5D dialog box.....	120
Figure 7. B-Spline Automatic Two-Pass Registration 2.5D dialog box .....	122
Figure 1. The outline of the method .....	128
Figure 2. A sample list file for a DICOM image .....	131
Figure 3. A sample list file for a PAR/REC image.....	132
Figure 4. A sample path file for a Phillips PAR/REC image.....	133
Figure 5. The DTI Create List File dialog box with the Registration option activated for DICOM images.....	135
Figure 6. The b-matrix, list and path files and the default directory for the PAR/REC dataset if the Register First option is activated in the dialog box .....	136
Figure 7. The DTI Create List File dialog box opened for a DICOM image.....	137
Figure 8. Running the DTI Create List File plug-in for PAR/REC datasets.....	138
Figure 1. Definition of the anatomical reference frame for brain images. The yz plane corresponds to the sagittal plane aligned with the interhemispheric fissure;	

the y axis corresponds to the anterior posterior intercommissural line. When no anisotropy filter is used, a particular color representation can be shown on a unit color representation sphere. The color representation sphere shown in this picture is for the no symmetry scheme (described in Section “No Symmetry” on page 143). The sphere is anchored to the anatomic frame of reference; thus, for a given scheme, an anisotropic structure has the same color independent of the view (axial, coronal, or sagittal). The figure also shows how Vmax for a generic direction is mapped on the color sphere. .... 142

Figure 2. a: Anatomic reference frame. b: Color representation sphere for the absolute value scheme. c: A unit sphere for your reference. In (b) the projection of the color representation sphere onto a color circle for axial and coronal views is shown, assuming that axial and coronal slices are exactly perpendicular to the z and y axes of the anatomical reference frame, respectively. Note that the grid for coronal view is not the projection of the constant theta and phi lines but rather the grid of constant thetav and phiv as defined in the text. .... 145

Figure 3. Applying the Adjust Exp. slider. The DTI Color Display dialog box shows two representations of the same image. (a): the Truncate option activated, thus the image intensity values, which are lower than the Anisotropy min threshold, are set to zero. (b): the Multiply option is activated, that makes the brightness of a given color to be modulated by the degree of anisotropy..... 148

Figure 4. An eigenvector (left) and anisotropy (right) files opened in MIPAV. .... 150

Figure 5. DTI Color Display dialog box. .... 152

Figure 6. Four types of color wheels which are used in the algorithm. .... 153

Figure 1. Edge detection by derivative operations – for given two images (light stripe on a dark background and dark stripe on a light background) the first and second derivatives were taken. Note that the second derivative has a zero crossing at the location of each edge .....155

Figure 2. The picture shows a cross section of the circularly symmetric function defined by Equation 4. The function crosses the zero point at r equals + or -sigma .. 156

Figure 3. The original image and its unsigned byte mask with detected edges .... 158

Figure 4. The EdgeLap dialog box options..... 158

Figure 1. The original image and its unsigned byte mask with detected edges..... 163

Figure 2. The EdgeNMSuppression dialog box options ..... 164

Figure 3. The original image (a) and its unsigned byte mask with detected edges depending on the standard deviation: (b) – standard deviation is set to 1; (c) – standard deviation is set to 4; (d) – standard deviation is set to 6; (e) – standard



---

deviation is set to 8; (f) – standard deviation is set to 6, but the original image has been smoothed before applying the algorithm. ....	165
Figure 1. The initial ellipsoids intersected with the middle slices. ....	169
Figure 2. Subdivision of a triangle. ....	170
Figure 3. An initial ellipsoid: (A) solid and (B) wireframe. ....	171
Figure 4. Two views of a brain surface mesh. ....	174
Figure 5. MRI slices with colored brain voxels. ....	175
Figure 6. Extract Brain dialog box ....	177
Figure 1. Examples of Extract Brain Surface (BSE) image processing.....	182
Figure 2. Extract Brain Surface (BSE) algorithm dialog box.....	183
Figure 3. The Extract Brain to Paint option: on your left is the original image and on your right is the result image with the brain extracted to paint. ....	185
Figure 1. An example of the Face Anonymizer (BET) image processing. The image has been processed using default parameters. ....	189
Figure 2. Face Anonymizer dialog box options.....	190
Figure 1. FFT algorithm processing ....	192
Figure 2. FFT dialog box.....	200
Figure 1. The ideal lowpass filter in frequency and spatial domains.....	205
Figure 2. Gaussian low pass filter: (a) a plot of Gaussian function, (b) the inverse Fourier transform of Gaussian, (c) Frequency response of Gaussian with different FO .....	206
Figure 3. Applying the Frequency Filter algorithm with different parameters.....	211
Figure 4. Extract Object dialog box.....	212
Figure 1. Adaptive Noise Reduction processing.....	216
Figure 2. Adaptive Noise Reduction dialog box.....	218
Figure 1. Example of adaptive path smooth processing .....	222
Figure 2. Adaptive Path Smooth dialog box for grayscale images.....	224
Figure 3. Adaptive Path Smooth dialog box for 3D color images.....	225

---

Figure 1. A 1D example of anisotropic diffusion .....	226
Figure 2. An image before and after applying the Anisotropic Diffusion algorithm .....	228
Figure 3. Anisotropic Diffusion dialog box .....	230
Figure 1. Example of coherence-enhancing diffusion.....	234
Figure 2. Coherence Enhancing Diffusion dialog box .....	235
Figure 3. Status message that appear when the Coherence-Enhancing Diffusion algorithm is running.....	236
Figure 1. Gaussian Blur algorithm processing .....	239
Figure 2. Gaussian function .....	240
Figure 3. Gaussian Blur dialog box .....	242
Figure 1. Edge detection using derivative operators.....	243
Figure 2. Gradient Magnitude algorithm processing.....	244
Figure 3. Gradient Magnitude dialog box .....	246
Figure 1. Construction of the east-west co-occurrence matrix for pixel distance equals 1. (a) – the original image with pixel image intensities showing as numbers; (b) shows the incremental stage that occurs when the outlined neighboring pixels in (a) are examined; (c) shows the final result of the co-occurrence matrix and (d) represents the matrix calculation.....	249
Figure 2. Applying the Haralick Texture algorithm: (a) – the original image, (b)–(i) result images.....	250
Figure 3. Haralick texture dialog box.....	251
Figure 1. Edge detection using laplacian operators .....	257
Figure 2. (A) Original MR image; (B) laplacian results; and (C) extraction of the zero crossing of the laplacian (object edges).....	258
Figure 3. Laplacian dialog box .....	260
Figure 1. Local Normalization algorithm processing.....	261
Figure 1. Slice averaging .....	263
Figure 2. Image processing using the Mean algorithm .....	264

---

Figure 3. Mean Filter dialog box .....	266
Figure 1. An example of median filter processing using a square 3 x 3 kernel .....	268
Figure 2. Removing shot noise with the Median Filter algorithm .....	269
Figure 3. Median Filter dialog box .....	272
Figure 1. Examples of (A) a segmented image and (B) the results of applying the Mode algorithm to the image .....	275
Figure 2. Mode Filter dialog box .....	277
Figure 1. Image processing using the Nonlinear Noise Reduction algorithm.....	281
Figure 2. Nonlinear Noise Reduction dialog box.....	283
Figure 1. Examples of Nonmaximum Suppression processing .....	289
Figure 2. Nonmaximum Suppression dialog box .....	291
Figure 1. Diffusion constant for .....	294
Figure 2. Examples of regularized isotropic (nonlinear) diffusion .....	295
Figure 3. Regularized Isotropic Diffusion dialog box .....	296
Figure 1. Slice Averaging algorithm processing .....	300
Figure 2. Slice Averaging dialog box .....	301
Figure 1. Unsharp Mask processing .....	304
Figure 2. Unsharp Mask dialog box .....	306
Figure 1. Application of the Filters wavelet BLS GSM algorithm to the noisy image .	309
Figure 2. Left: an example of decomposition of an image of a white disk on a black background; Right: the block diagram for that decomposition .....	310
Figure 3. Wavelet Thresholding dialog box .....	311
Figure 1. Meyer wavelet .....	315
Figure 2. The sample images after applying the Filers Wavelet Threshold algorithm. All options were applied with the default dialog box settings which are Number of terms =4, Wavelet threshold=1.0E-4 .....	321
Figure 3. Wavelet Thresholding dialog box .....	323

---

---

Figure 4. Applying the Wavelet Thresholding algorithm .....	324
Figure 1. Fuzzy C-means algorithm processing .....	330
Figure 2. Fuzzy C-Means dialog box for a single image.....	331
Figure 3. Fuzzy C-Means dialog box for multispectral images.....	333
Figure 1. Two input images, output 2D histogram image and its lookup table.....	336
Figure 2. The Histogram Two Dimensional dialog box for grayscale images .....	337
Figure 3. The Histogram Two Dimensional dialog box for color images.....	339
Figure 1. A sample cumulative histogram .....	340
Figure 2. A 3D grayscale image (a) and its cumulative histogram (b). .....	341
Figure 3. A 2D RGB image and its cumulative histograms for each channel.....	341
Figure 4. A 2D RGB image and its cumulative histograms . .....	342
Figure 1. Source image and histogram and reference images and histograms for reference images that were separated into 1 x 1, 2 x 2, and 3 x 3 regions.....	346
Figure 2. Examples of no clipping, 45-percent clipping, and 75-percent clipping	349
Figure 3. The Regional Adaptive Histogram Equalization dialog box .....	352
Figure 1. Sample of kernel created by algorithm .....	355
Figure 2. A source image and its histogram (A, B) compared to reference images using a 21 x 21 (C, D), 45 x 45 (E,F), or 65 x 65 (G,H) square kernels and 65 x 65 (I,J) cross kernels .....	356
Figure 3. The effect of thresholding on images.....	360
Figure 4. A reference image and histogram with no clipping and with 75-percent clipping .....	361
Figure 5. A reference image and histogram with 45-percent clipping .....	362
Figure 6. Local Adaptive Histogram Equalization dialog box.....	364
Figure 1. Histograms and cumulative histograms: (A) histogram of 8x8 pixel image; (B) cumulative histogram derived from (A); (C) desired histogram of the same 8x8 pixel image; (D) cumulative histogram derived from (C); (E) gray scale transformation function that approximately transforms the	

histogram (A) to desired histogram (C);  
 (F) histogram of gray-scale-transformed image obtained by applying the transformation function in (E) to the image with the histogram shown in (A). ... 367

Figure 2. The input image and its histogram  $p(f)$  ..... 368

Figure 3. The specified image and its histogram  $p_d(g)$  ..... 368

Figure 4. The result image and its histogram ..... 368

Figure 5. Histogram Matching dialog box ..... 370

Figure 1. Histogram summary in the Output window ..... 372

Figure 2. The Histogram summary dialog box options ..... 373

Figure 1. The Load dialog box ..... 376

Figure 2. The Blended tab and the image slider ..... 377

Figure 3. The Blended tab options ..... 377

Figure 4. The Blended tab: activating moving and rotation modes ..... 380

Figure 5. Rotating the adjusted image. .... 382

Figure 6. The Dual tab options ..... 383

Figure 7. The Dual tab shows both: the reference image and adjusted image side by side. Three corresponding landmark points were delineated in the reference and adjusted image. .... 385

Figure 8. The Window Region box is filled with the adjusted image ..... 388

Figure 9. The adjusted image and its Lookup table ..... 389

Figure 10. The adjusted image which appears in the Blended tab and also in the Window Region. .... 390

Figure 11. Applying the Manual 2D Series Registration algorithm on a 2.5D image. Slice 3 (adjusted slice) is registered to slice 1 (reference slice). To differentiate the slices, we use LUT options to color the reference slice to pale green and adjusted slice to pale blue. .... 392

Figure 12. Registering the 3D image. .... 394

Figure 1. The indexing convention for vertices and edges. Here, edge indexes are shown in blue and vertex indexes are shown in magenta. .... 395

Figure 2. Vertex 3 is inside the volume ..... 396

---

Figure 3. How 256 cases can be reduced to 15 cases .....	397
Figure 4. The Extract Surface dialog box options .....	400
Figure 5. The Resample dialog .....	401
Figure 6. The view of the surface extracted using the VOI Region option .....	402
Figure 7. The view of the surface extracted using the Mask Image option .....	403
Figure 8. The intensity level is shown on the main MIPAV toolbar .....	404
Figure 9. Enter the intensity level value here .....	405
Figure 10. The view of the surface extracted using the Intensity Level option .....	406
Figure 1. Mask algorithm processing .....	407
Figure 2. Mask Image dialog box .....	409
Figure 1. Orthogonal Regression Colocalization dialog boxes for (A) color and (B) grayscale images .....	421
Figure 2. 2D Histogram window .....	424
Figure 3. Colocalization Histogram window showing a histogram for a color image in which red and green color localization was done .....	426
Figure 4. The effect on the image when dragging the mouse-movable point on the histogram.....	427
Figure 5. The effect on the image when dragging the mouse-movable point on the histogram.....	428
Figure 1. The original color image (A), the intermediate grayscale image (B), and the final estimate color image after 10 iterations (C). .....	432
Figure 2. The original image of Jupiter in grayscale (A), the estimate image after five iterations (B), and the final estimate after 10 iterations (C).....	432
Figure 3. The Maximum Likelihood Iterated Blind Deconvolution dialog box ....	434
Figure 4. A status pop-up window. ....	435
Figure 5. The PSF warning. ....	436
Figure 6. The Blind Deconvolution algorithm was applied on the color image (first slide) with the default parameters values (shown on the second slide). The intermediate grayscale images are also shown. ....	437

---

Figure 1. Fluorescence Recovery after Photobleaching dialog box.....	454
Figure 2. An image with a required photobleached VOI in red and a whole organ VOI in green.....	456
Figure 3. Graph windows: Photobleaching Recovery Curve, Uncorrected Photobleaching Recovery Curve, and Uncorrected Whole Organ windows .....	458
Figure 4. Print dialog box .....	460
Figure 5. Modify Graph Features command on the Views menu in a graph window.....	461
Figure 6. Modify Graph dialog box showing the Graph page .....	462
Figure 7. Pick Background Color dialog box .....	463
Figure 8. Legend page in the Modify Graph dialog box.....	465
Figure 9. Functions page in the Modify Graph dialog box .....	466
Figure 10. Fitted Functions page in the Modify Graph window.....	467
Figure 1. Load Bleached ROI message .....	471
Figure 2. Fluorescence Resonance Energy Transfer (FRET) dialog box.....	473
Figure 3. Example of (A) prebleached image and (B) postbleached image after FRET algorithm was applied.....	474
Figure 4. FRET data in the Data page of the Output window .....	476
Figure 1. Jablonski diagram for FRET method.....	479
Figure 2. Nine example images used to calculate FRET efficiency .....	482
Figure 3. Algorithms > Microscopy > FRET Bleed Through command.....	486
Figure 4. FRET Bleed Through dialog box for (A) color images and (B) grayscale images .....	487
Figure 5. Acceptor only run: FP1 image with a background VOI in blue and an active VOI in orange (example image: R3_543F.tif) .....	488
Figure 6. Open Image dialog box .....	490
Figure 7. Results in the Output window from the Acceptor dye only run of the FRET Bleed Through algorithm .....	491
Figure 8. Donor dye only selection.....	491

---

Figure 9. Add background VOI selection .....	492
Figure 10. Add active VOI selection .....	492
Figure 11. Donor only run: FP1 image with a background VOI in blue and an active VOI in orange (example image: Y5_488Y.tif) .....	493
Figure 12. Open Image dialog box.....	494
Figure 13. The loaded images in the FRET Bleed Through dialog box .....	495
Figure 14. Results in the Output window from the Donor dye only run of the FRET Bleed Through algorithm .....	495
Figure 15. FRET Bleed Through dialog box .....	496
Figure 16. FRET Efficiency dialog box (A) color images and (B) grayscale images....	499
Figure 17. FRET Efficiency dialog box .....	501
Figure 1. Image formation and recording .....	505
Figure 2. The shifted version of the transfer function conventionally, but incorrectly, called the point spread function (PSF) .....	506
Figure 3. The observed bead used in this documentation. The red lines correspond to the bounding box of the image.....	507
Figure 4. Observed and PSF images: (a) observed image, (b) PSF image as it appears initially, (c) restored image, and (d) PSF image after applying the Quick LUT icon .....	510
Figure 5. XCOSM Expectation Maximum Restoration dialog box.....	511
Figure 1. The ideal coordinate system attached to the head in which the fissure is close to the plane $Z = 0$ (red) and the coordinate system of the image (blue) differ from each other by way of three angles (yaw, roll and pitch) and a 3D translation. ..	513
Figure 2. The original image (A) and the image after applying the midsagittal line alignment algorithm (B).....	516
Figure 3. The Midsagittal Line Alignment algorithm is running. ....	517
Figure 1. (a) Image A; (b) image A translated; (c) image B; (d) reflection of image B; (e) image A and its complement; (f) the difference of A and B. The origin of the image is shown as a black dot.....	519



Figure 2. Applying the Background Distance map operation to the image. You can see 1) the original image (left), 2) the dialog box with the default settings (center) and 3) the background distance map (right) ..... 521

Figure 3. Illustration of closing (a) the original image A, (b) the dilation of image A by kernel B, (c) the erosion of result of (b) by the same kernel B..... 522

Figure 4. Close dialog box..... 523

Figure 5. The original image (a) and the result image after the closing that includes 15 dilations followed by 15 erosions applied with the “3x3 -4 connected” kernel. 524

Figure 6. Applying the Delete Objects operation; (a) the original image, (b) the result image. The parameters are shown in the dialog box ..... 525

Figure 7. (a) An original image A, a square structural element B and its reflection, and the dilation of A by B; (b) an original image A, an elongated structural element B and its reflection, and the dilation of A by B ..... 525

Figure 8. Dilation dialog box..... 527

Figure 9. The original image (a) and the result image after 5 dilations applied with the “3x3–4 connected” kernel ..... 528

Figure 10. Applying the Background Distance map operation to the image. You can see 1) the original image (left), 2) the dialog box with the default settings (center) and 3) the background distance map (right) ..... 529

Figure 11. Erosion dialog box ..... 530

Figure 12. The original image (a) and the result image after 5 erosions applied with the “3x3 -4 connected” kernel ..... 531

Figure 13. The Evaluate Segmentation algorithm : (a) – a gold standard segmented image, (b) – its unsigned byte mask, (c) – a segmented image for evaluation, (d) – its unsigned byte mask, and (d) – the Evaluate Segmentation dialog box and the Output window ..... 532

Figure 14. The Fill Holes algorithm: (a) the original image; (b) the Fill Objects dialog box; (c) the result image ..... 534

Figure 15. The Find Edges algorithm: (a) the original image; (b) the Find Edges dialog box; (c) the result image ..... 535

Figure 16. ID Objects algorithm ..... 536

Figure 17. a: the original image; b: the same image after applying the noise reduction algorithm; c: the image from (b) after applying Morphological filter with parameters shown in Figure 18 ..... 539

Figure 18. Morphological Filter dialog box.....540

Figure 19. Illustration of opening (a) the original image A, (b) the erosion of image A by kernel B, (c) the dilation of result of (b) by the same kernel B, (d) image A after opening ..... 541

Figure 20. Open dialog box ..... 542

Figure 21. The original image (a) and the result image after the opening that includes 19 erosions followed by 19 dilations applied with the “3x3 -4 connected” kernel ..... 543

Figure 22. Particle Analysis: (a) – an original binary image. The following steps were performed: (b) erosion; (c) dilation; (d) skeletonize; (e) prune; (f) watershed; (g) – the result image; (h) – particle statistics as it appears in the Output window... 544

Figure 23. Particle Analysis New dialog box..... 546

Figure 24. Skeletonize operation: (a) the original image, (b) the result after applying the operation with the parameters shown in the dialog box ..... 548

Figure 25. (a) - the original image, (b) - the skeleton ..... 550

Figure 26. (a): the original image; (b): the Pad dialog box; (c): the padded image; (d): the Output window shows the skeletonize 3D statistics; (d): the Level 1 skeleton obtained using the default parameter values, see Figure 27 on page 554. .... 552

Figure 27. Skeletonize 3D dialog box ..... 554

Figure 28. Ultimate Erode – (a) the original binary image, (b) the result after applying the operation with the parameters shown in the dialog box ..... 557

Figure 1. A block diagram of the thigh muscle segmentation method ..... 559

Figure 2. A typical image processed by the Muscle Segmentation plug-in. Visible in the figure are the right and left thigh as well as the water phantom, which appears as the gray rectangular block at the bottom center of the image..... 560

Figure 3. LiveWire produced VOI (Orange), LevelSet produced VOI (Red); Plot of intensity values along green line; Magnitude of gradient along the green line perpendicular to thigh edge; Intensity values when all the pixels inside the livewire detected VOI have been set to 1..... 561

Figure 4. Each pixel of the matrix is a vertex of the graph and has edges going to the 8 pixels around it, as up, down, left, right, upper-right, upper-left, down-right, down-left..... 562

Figure 5. Segmented image that results from the Muscle Segmentation plug-in. The different muscles and fat regions are assigned different values..... 568

---

Figure 6. Plug-in output .....	568
Figure 7. The plug-in user interface with, first, the Thigh tab selected, then the Bone tab selected, and finally, the Muscle tab selected. All VOIs are available for editing .	570
Figure 8. The catalogue where segmentation VOIs are stored .....	573
Figure 9. Fat/Lean ratio for the thigh and thigh muscles in the Output window. Use the scroll bar to view the whole statistics.....	574
Figure 10. Show and Hide LUT options .....	575
Figure 1. Determination of subcutaneous fat layer: (a) - radii drawn from the outer contour towards the body center at a fixed angle, (b) - a line VOI covering the radius, (c) - an intensity profile along the line VOI .....	577
Figure 2. Segmentation: (a) - an abdomen (green) and subcutaneous area (blue) and (b) - a visceral cavity (yellow).....	578
Figure 3. The Abdomen tab shows abdomen VOIs.....	579
Figure 4. Calculating the fat/lean ratio for the abdomen and subcutaneous area	581
Figure 5. The Tissue tab .....	582
Figure 6. The Muscles tab.....	583
Figure 7. The “Too many curves” warning.....	585
Figure 8. The “Any curves made must be closed “warning.....	585
Figure 9. The “MIPAV created VOI” warning .....	586
Figure 10. The plug-in looks for VOIs in the subdirectory of the directory where the image dataset is stored .....	587
Figure 11. VOI color selection.....	588
Figure 12. Manual segmentation steps.....	589
Figure 13. The MIPAV automatic segmentation of the abdomen and subcutaneous area.....	590
Figure 14. Using the MIPAV Statistics Generator.....	591
Figure 15. The Analysis tab.....	593
Figure 16. Saving the statistics as a PDF file .....	594

---

Figure 17. The MIPAV Segmentation PDF file.....	594
Figure 18. Showing the LUT for selected VOIs .....	595
Figure 1. The plots of the Correlation Ratio cost function versus some of the individual parameter values. In each plot (A, B, C, and D), a single parameter is varied while the other are kept constant.....	597
Figure 2. Calculating the center of mass (COM) .....	600
Figure 3. levelEight optimization .....	603
Figure 4. The input image (A), reference image (B) and result image (C). Registration was performed using the default algorithm settings. ....	608
Figure 5. The Optimized Automatic Registration dialog box options .....	609
Figure 6. The Advanced Optimized Automatic Registration (constrained) dialog box options .....	612
Figure 1. Patient Position (DICOM) algorithm processing.....	615
Figure 1. Landmark—Least Squares processing .....	620
Figure 2. Least Squares Registration dialog box.....	622
Figure 1. Example of Registration: Landmark—TPSpline algorithm.....	624
Figure 2. Thin Plate Spline Registration dialog box .....	626
Figure 1. Registration: Manual 2D Series window.....	627
Figure 2. The Load dialog box.....	630
Figure 3. The Blended page of the Manual 2D Series window showing the adjusted image moved down and to the right.....	635
Figure 4. Manual 2D Series window showing center of rotation .....	637
Figure 5. Manual 2D Series window showing the Blended page.....	638
Figure 6. Manual 2D Series window showing the Dual page .....	641
Figure 7. Manual 2D Series window.....	644
Figure 1. Calculating the rotation angle .....	647
Figure 2. The translation is calculated based on the mouse's X and Y coordinates before and after translation.....	648

---

Figure 3. The plots of the Correlation Ratio cost function versus some of the individual parameter values. In each plot (A, B, C, and D), a single parameter is varied while the other are kept constant.....	651
Figure 4. AlgorithmOAR2D the levelEight optimization steps .....	654
Figure 5. Running Mosaic Registration .....	659
Figure 6. Mosaic Registration dialog box.....	660
Figure 7. The Optimized Automatic Registration 2D dialog box options .....	661
Figure 8. The Brute Force algorithm settings .....	663
Figure 1. The Display Pixel Similarity Cost Functions method: (a) the first image, (b) the second image, and (c) the output of the method. ....	665
Figure 2. Show Pixel Similarity Cost Function Values dialog box.....	666
Figure 1. The Time Series Registration dialog box .....	674
Figure 1. Reslice—Isotropic Voxels algorithm applied to an image .....	677
Figure 2. Reslice dialog box.....	678
Figure 1. Applying the VOI landmark algorithm: (a) the first image slice with delineated VOI, (b) the result image when using the Exhaustive search for a global minimum, (c) the first slice of another image with delineated VOI, (d) the result image when using the Downhill Simplex search for a global minimum .....	682
Figure 2. Applying the Registration- VOI Landmark algorithm using the Exhaustive search option .....	684
Figure 3. Applying the Registration- VOI Landmark algorithm using the Downhill Simplex search option .....	685
Figure 4. Register (VOI Landmark) dialog box .....	686
Figure 1. Flow chart describing the N3 method.....	692
Figure 2. The N3 method showing the equations.....	693
Figure 3. The effect of N3 on an image: (A) original image and (B) image after N3 correction.....	694
Figure 4. The effects of different settings for the Inhomogeneity N3 algorithm on an image.....	695
Figure 5. N3 Inhomogeneity Correction dialog box .....	699

Figure 1. (a): an original image with delineated single VOI, (b): the image thresholded using the default parameters opened in a new image frame, (c) the thresholded image painted over the original image, (d) VOI statistics for the original image .....700

Figure 2. Threshold Using Standard Deviation dialog box options for grayscale images ..... 702

Figure 3. (a): an original RGB image with delineated single VOI, (b): the same image thresholded using the default parameters opened in a new image frame, (c) statistics for the original image appears in the Output MIPAV window, (d) the Threshold Using Standard Deviation dialog box for RGB images with the default settings..... 705

Figure 4. Threshold Using Standard Deviation dialog box options for RGB images.. 706

Figure 1. Original image before subsampling .....708

Figure 2. Subsample dialog box ..... 709

Figure 3. An image subsampled by 2, by 4, and then by 8 ..... 710

Figure 1. The Threshold dialog box. ....712

Figure 1. Transformation options.....715

Figure 2. The options available for defining a transformation matrix: Tx, Ty, Tz in mm - translations in the X, Y, and Z directions; Rx, Ry, Rz in degrees - rotations along the X, Y, and Z axis; Sx, Sy, Sz - scale in the X, Y, and Z directions; SKx, SKy, and SKz – skew in the X, Y, and Z directions.....715

Figure 3. The rotation choices. ....716

Figure 4. Two examples of using the Retain original image size option for 2D images: (b) with padding and (c) with cropping. An example of using the Pad image option to include entire original image option is (d). (a) – the original 2D image, (b) – the image scaled to 0.5 in X direction, (c) – the image scaled to 2 in X direction, (d) – the image scaled to 0.5 in X direction and padded with pixels with intensities equal to 140..... 717

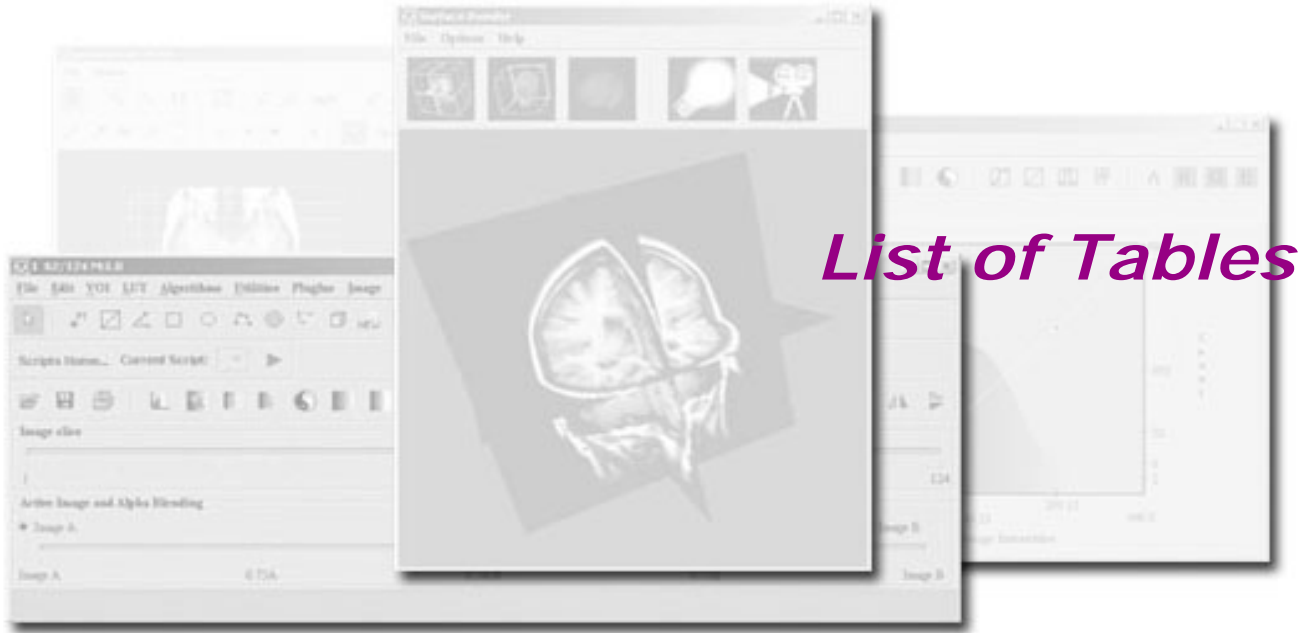
Figure 5. The Resample tab .....719

Figure 6. Resampling images: (a) the original image, (b) the image resampled by 0.5 in X direction while the XY aspect ratio has been preserved, (c) the image resampled by 2 in X direction while the XY aspect ratio has been preserved..... 722

Figure 1. Calculating the center of the circle to which the circular segment belongs . 724

---

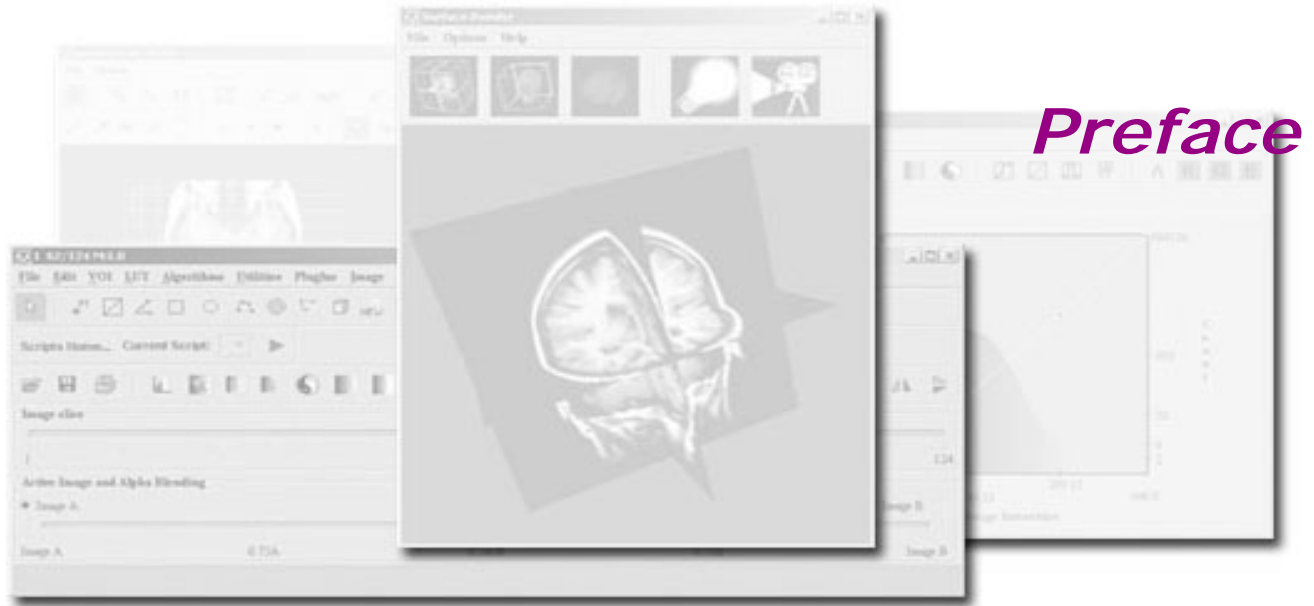
Figure 2. Calculating the angle of the circular segment .....	724
Figure 3. The region inside a circular sector is mapped to a rectangle: z1 is the upper right point on rmax, z2 is the upper left point on rmax, z3 is the lower left point on rmin, and z4 is the lower right point on rmin .....	727
Figure 4. The user inputs 2 point VOIs. The first point is located at the center of the circle. The second point can be any point located on the actual circle.....	728
Figure 5. Applying the Circular Segment to Rectangle algorithm.....	734
Figure 6. Applying the Circle to Rectangle algorithm.....	735
Figure 7. Applying the Ellipse to Circle algorithm: (a)-the eye image, (b)-the distorted eye image with an elliptical VOI, (c)-the part of the image under the VOI is transformed to a circle using the dialog box settings.....	736
Figure 8. Applying the Nearly Circle to Circle algorithm: (a)-the eye image with a nearly circular VOI, (b)-the part of the image under the VOI is transformed to a circle using the dialog box settings.....	737
Figure 1. Applying the Transform Nonlinear algorithm.....	739
Figure 2. A sample .nlt file.....	740
Figure 1. The Resample dialog box options .....	742
Figure 1. 1D example of watershed segmentation: (a) grey level profile of the image data, (b) watershed segmentation.....	743
Figure 2. Segmenting images using the interactive Watershed algorithm .....	745
Figure 3. Watershed dialog box.....	746



# List of Tables

- Table 1. MIPAV algorithms and their effects on images.....51
- Table 1. Possible responses of the SUSAN filter ..... 279
- Table 1. edgeTable..... 397
- Table 1. Setup of nine example images..... 480
- Table 2. Six images used in applying the FRET Bleed Through algorithm.....483
- Table 64.1 A determinant of X depending on the image type..... 619





The purpose of the MIPAV software program is to allow medical researchers to extract quantitative information from image datasets of various medical image modalities.

---

## Scope of this guide

The primary purpose of the *MIPAV User's Guide* is to provide medical researchers the information they need to use the MIPAV software program.

A secondary goal is to give researchers the information needed to extend, if desired, the software's capabilities through the development of new functions in plug-in applications via use of the software's application program interface (API).

---

## Who should use this guide

Medical researchers, medical technicians, and other people who are involved in analyzing medical images or maintaining and supporting the equipment used to produce images form the audience for the *User's Guide*.

---

## Skills you need to perform tasks in MIPAV

Depending on the platform—Windows, MacIntosh, or Unix—that is running your workstation, it is recommended that you are familiar with installing and using software programs for that platform. If you plan to create plug-in applications for MIPAV to add new functionality, you must have software programming skills and be familiar with Java.

---

## How this guide is organized

The *MIPAV User's Guide* is divided into two volumes:

- Volume 1, *Basics*, explains how to use the basic features and functions of MIPAV and how to incorporate plug-in applications.
- Volume 2, *Algorithms*, presents detailed information about the purpose, background, and function of the algorithms packaged with MIPAV and gives instructions for using them.

## Volume 1, Basics

The *MIPAV User's Guide*, Volume 1, *Basics*, includes the following:

- Chapter 1, “Introducing MIPAV,” presents an overview of the MIPAV software program.
- Chapter 2, “Installing MIPAV,” explains how to install, remove, and upgrade the MIPAV software program. It also explains how to subscribe to the MIPAV mail list and how to search the MIPAV archive.
- Chapter 3, “Getting Started Quickly with MIPAV,” explains how to use MIPAV to perform common functions, such as opening an image file, saving the file, and printing a log file.
- Chapter 4, “Understanding Image Basics,” provides background information on image file formats. It also provides information on how to view and adjust image file attributes.
- Chapter 5, “Working with DICOM Images,” explains how to access DICOM databases, perform queries, and retrieve image files. In addition, it explains how to send files to a database.

- Chapter 6 “Get Connected to SRB BIRN” explains how to connect to SRB BIRN.
- Chapter 7 “NDAR Imaging Import,” provides the help for the NDAR Imaging Import tool, which is designed to assist users in moving data from the MIPAV XML output to the NDAR shared data repository for the benefit of the greater autism community.
- Chapter 8, “Visualizing Images,” provides instruction on how to customize the way image files are displayed, how to magnify and minify images, how to view images together, how to view a portion of the image, and how to change image brightness and contrast by generating histograms and adjusting color look-up tables (LUTs).
- Chapter 9, “Segmenting Images Using Contours and Masks,” explains how to create, group, rearrange, and modify volumes of interest (VOIs); how to create masks; and how to use paint to further identify VOIs.
- Chapter 10, “Analyzing Images,” discusses how to calculate statistics for VOIs and masks and how to generate intensity profiles, or graphs, for images.
- Chapter 11, “Changing Image Datasets Using MIPAV Utilities . . .,” explains how to use the utilities included in the software.
- Chapter 12, “Using Scripts (Macros) in MIPAV,” describes how to develop scripts, which you can use to customize the program.
- Chapter 13, “Developing Plug-in Programs,” and Appendix D explain how to incorporate plug-in programs into MIPAV.
- Chapter 14, “Technical support” on page 557 provides information about MIPAV technical support and also explains how to use the debug mode.
- Appendix A, “References,” provides a list of references that can be used to learn more about MIPAV functions.
- Appendix B, “DICOM Conformance Statement,” provides a copy of the formal DICOM Conformance Statement, which specifies MIPAV’s service classes, information objects, communications protocols, and media storage application profiles.
- Appendix C, “Supported formats” on page 562 lists graphical and file formats supported by MIPAV. It also provides examples of MIPAV

system files (such as the preference file) and explains how the user can interpret them; provides limited instruction on how to modify specific files.

- Appendix D, “PluginAlgorithmMedian” gives an example of MIPAV plug in.

The guide also includes a glossary of terms and acronyms.

## Volume 2, Algorithms

Volume 2, *Algorithms*, includes two chapters:

- Chapter 1, “Understanding MIPAV capabilities,” which discusses the tools and application programming interface provided with MIPAV
- Chapter 2, “Using MIPAV Algorithms,” provides detailed information about the algorithms packaged in MIPAV

In addition, the book includes the glossary of terms and acronyms.

---

## Where to find more information

Both volumes 1 and 2 of the *MIPAV User’s Guide* are available as Acrobat PDF files, which you can view, download, and print. You can either print each volume, or you can print individual chapters separately. For PDFs of this guide, go to the MIPAV web site: <http://mipav.cit.nih.gov>

## Conventions used in this guide

This guide uses the following conventions:

This convention . . .	Stands for . . .
<i>Italics</i>	Names of books, guides, or manuals as references New terms or emphasis Names of executable files
<b>Bold</b>	User input Names of programming commands
All caps	File types, such as TIFF, GIF, or JPG
Upper- and lowercase	Names of keys
name@address.com	E-mail address format
<u>Hyperlink</u>	An internet link (position the cursor on this word and click the left mouse button)*
Monospace	Code sample, including keywords and variables within text and as separate paragraphs, and user-defined program elements within text

\*All figure and table citations, such as Figure 1 or Table 1, are hyperlinks although they are not underscored. Clicking the citation displays the appropriate figure or table.

Both volumes of the *MIPAV User's Guide* include special information that briefly highlights particular features or functions or that provide clarification. Based on the type of information they convey, these notes are labeled “note,” “tip,” “example,” “recommendation,” “remember,” “reference,” “caution,” and “disclaimer.” The following examples indicate how these notes appear and the type of information they include.



**Note:** Notes provide additional information that is related to the subject at hand. They tend to be “by the way” types of information or asides.



**Tip:** Tip paragraphs point out application shortcuts or briefly describe special features of the software.



---

**Example:** An example paragraph provides an example of a task or incident in which something of note could occur.

---



---

**Recommendation:** Paragraphs that are labeled “Recommendation” suggest methods of good practice or advice.

---



---

**Definition:** The definitions of specific words or phrases of note appear in “definition” paragraphs.

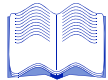
---



---

**Remember:** Notes labeled “Remember” present information that was previously discussed and that is pertinent in the current topic.

---



---

**Reference:** A reference note highlights one or more references that contain information on the current topic.

---



---

**Caution:** A paragraph labeled “Caution,” alerts you to be very careful about avoiding some action that could harm your equipment or data.

---

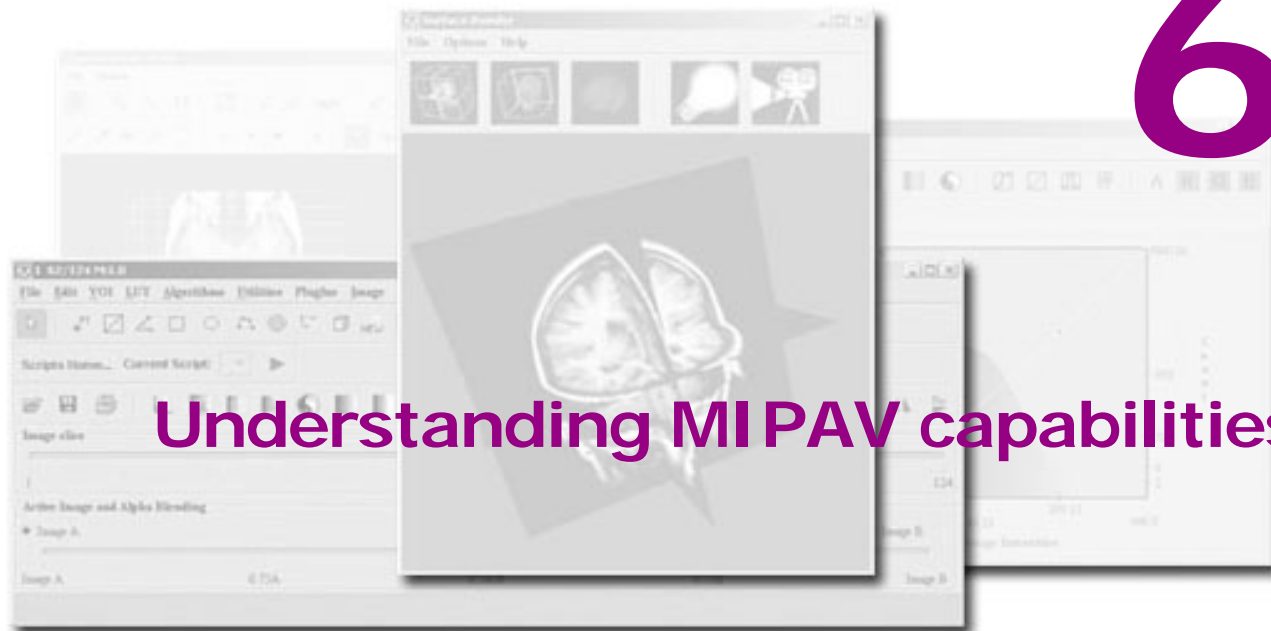


---

**Disclaimer:** A disclaimer indicates the possible limitations or ramifications of a topic.

---

# 6



## Understanding MIPAV capabilities

---

### MIPAV Algorithms Overview

- “Introducing MIPAV”
- “Understanding MIPAV capabilities”
- “Developing new tools using the API”

---

### Introducing MIPAV

Imaging is an essential component in many fields of medical research and clinical practice. Biologists study cells and generate three-dimensional (3D) confocal microscopy datasets, virologists generate 3D reconstructions of viruses from micrographs, radiologists identify and quantify tumors from Magnetic Resonance Imaging (MRI) and Computerized Tomography (CT) scans, and neuroscientists detect regional metabolic brain activity from Positron Emission Tomography (PET) and functional MRI scans. Analysis of these diverse types of images requires sophisticated computerized quantification and visualization tools. Until recently, 3D visualization of images and quantitative analyses could only be performed using expensive UNIX workstations and customized software. Because of technological advancements, much of the today's visualization and analysis can be performed on an inexpensive desktop computer equipped with the appropriate graphics hardware and software. The Medical Image

Processing, Analysis, and Visualization (MIPAV) software application takes advantage of the recent technological advancements, while offering functionality specifically designed for the medical community.

MIPAV is an extensible image processing, analysis, and visualization software. Because MIPAV is Java-based, it can run on virtually any platform. It can also accommodate  $n$ -dimensional images in over 20 different file formats, including DICOM, Analyze, TIFF, JPG, and RAW.

MIPAV is designed to be both a software application and an application programming interface (API). As a software application, MIPAV provides an easy-to-use graphical user interface (GUI) that allows researchers to extract quantitative information from various medical imaging modalities.

---

## Understanding MIPAV capabilities

MIPAV provides ready-made, general-purpose tools that meet the majority of requirements of many researchers. Researchers can use these tools to perform a variety of tasks:

- View image files, juxtapose images, and adjust opacity level so that overlapping areas can be studied
- Create image files and view and modify the attributes of an image, including DICOM and volume of interest (VOI) information
- Adjust the display of an image file and adjust magnification settings
- View DICOM overlays and sanitize DICOM information and send and receive image files to and from databases residing on DICOM-compliant servers (on computer or imaging device)
- Manually, semiautomatically, and automatically delineate or segment volumes of interest (VOI) and generate graphs and run statistics on VOIs
- Generate and adjust histograms and lookup tables (LUT) using customized or preset options
- Run sophisticated, predefined algorithms and generate a log of each algorithm run for each image
- Create new plug-ins to further customize the analysis of data



- Save transformation, look-up table (LUT), and VOI data and apply them to other images
- Print image files, graphs, statistical data, algorithmic logs, and debugging log data

---

## Developing new tools using the API

Researchers who have programming resources can use MIPAV as an application programming interface (API) to develop and incorporate new and innovative image processing, registration, and visualizations. MIPAV provides researchers the ability to quickly build a specific program, which can be installed in MIPAV as a plug-in, to quantify and visualize their data.

The API provides much flexibility while the software application provides a variety of commonly used features, thus allowing researchers to use MIPAV to meet a variety of research needs.

For more information of how to develop MIPAV plug-ins, refer to MIPAV Users Guide Volume 1, Chapter “Developing Plug-in Programs”.



---

### In this chapter . .

“Recording algorithms usage with the history feature” on page 50

“Introducing MIPAV algorithms” on page 51

---

MIPAV supports a wide range of image-processing algorithms to facilitate the quantification of data from medical images. Although MIPAV provides storage for all types of 3D volumes, most of the algorithms are designed for application to 3D datasets where all three dimensions are spatial. However, MIPAV’s storage techniques do not preclude developing algorithms or visualization for datasets of any dimensionality.

---

## Recording algorithms usage with the history feature

MIPAV provides a way for you to record the actions, whether with algorithms or utilities, that you perform on images. To turn on this feature, use the MIPAV Options dialog box. See also “Debugging MIPAV” on page 76.

## Introducing MIPAV algorithms

Table 1 lists both the current and planned MIPAV algorithms (planned algorithms marked as TBD). This table also explains what effect the algorithms have on images.

**Table 1. MIPAV algorithms and their effects on images**

Algorithm	Effect	Image types
"Interpolation methods used in MIPAV"	Bilinear, Trilinear, B-spline 3-rd order, B-spline 4-th order, Cubic Lagrangian, Quintic Lagrangian, Heptic Lagrangian, and Windowed sinc	
"Cost functions used in MIPAV algorithms"		
"Autocorrelation Coefficients"	The algorithm calculates autocorrelation coefficients for images	Color and black and white 3D and 4D images
"Autocovariance Coefficients"	The algorithm produces autocovariance coefficients which help to provide a description of the texture or a nature of the noise structure.	Color and black and white 3D and 4D images.
"Barrel Distortion Correction"	The algorithm is a cost-effective alternative to an expensive lens system	RGB and grayscale 2D images
<b>DTI</b>		
"DTI Color Display"	This plug-in introduces a novel technique to visualize nerve fiber tracts in diffusion-weighted magnetic resonance imaging data	The algorithm works with any image types supported by MIPAV

**Table 1. MIPAV algorithms and their effects on images (continued)**

Algorithm	Effect	Image types
"DTI Create List File"	This utility takes as an input a DICOM study directory or a PAR/REC file extracts the information and creates the following files: <ul style="list-style-type: none"> <li>– A list file with metadata information</li> <li>– A sorted relative path file that contains the relative paths to the slice files</li> <li>– B-matrix file</li> </ul>	DICOM study directory or a PAR/REC file
<b>Edge Detection</b>		
"Edge Detection: Zero X Laplacian"	The algorithm finds edges that form a closed contour, which completely bound an object	2D and 3D grayscale images
"Edge Detection: Zero X Non-Maximum Suppression"	The method produces an edge map of the zero-crossings of the non-maximum suppression for 2D and 3D images	2D and 3D grayscale images
Extract Surface: Adaptive Climbing	TBD	TBD
Extract Surface: Tetrahedron	TBD	TBD
<b>Brain tools</b>		
"Extract Surface (Marching Cubes)"	Extracts surface information from a 3D array of values	2D and 3D MRI images
"Extract Brain: Extract Brain Surface (BET)"	Extracts the surface of the brain from a T1-weighted MRI	3D MRI images
"Extract Brain: Extract Brain Surface (BSE)"	This algorithm strips areas outside the brain from a T1-weighted magnetic resonance image (MRI)	3D MRI images
"Face Anonymizer (BET)"	The algorithm extracts an approximated face from a T1-weighted MRI, eliminating facial areas outside of a certain buffer from the brain	You can apply this algorithm only to 3D T1-weighted MRI images
<b>Fourier Transform</b>		

**Table 1. MIPAV algorithms and their effects on images (continued)**

Algorithm	Effect	Image types
"Fast Fourier Transformation (FFT)"	Processes images by filtering in the frequency domain	<ul style="list-style-type: none"> <li>• Gray-scale 2D and 3D images</li> <li>• Conjugate, symmetric, and complex data (frequency filter or inverse FFT only)</li> </ul>
<b>Filters</b>		
"Filters (Frequency)"	Frequency filters process an image in the frequency domain	2D and 3D grayscale images
"Filters (Spatial): Adaptive Noise Reduction"	Reduces noise without blurring the edges	<ul style="list-style-type: none"> <li>• Color images</li> <li>• Black-and-white images</li> </ul>
"Filters (Spatial): Adaptive Path Smooth"	The algorithm reduces noise without blurring edges. It replaces a pixel value with a weighted sum of the local pixels on the low-cost path	Color and grayscale 2D and 3D images
"Filters (Spatial): Anisotropic Diffusion"	Blurs image except across the boundaries	Grayscale images.
"Filters (Spatial): Coherence-Enhancing Diffusion"	Useful for filtering relatively thin, linear structures such as blood vessels, elongated cells, and muscle fibers	<ul style="list-style-type: none"> <li>• All data types except complex</li> <li>• 2D, 2.5D, and 3D images</li> </ul>
"Filters (Spatial): Gaussian Blur"	Blurs an image. This algorithm produces a similar result as applying a low-pass or smoothing filter	<ul style="list-style-type: none"> <li>• All image data types except Complex</li> <li>• 2D, 2.5D, 3D, and 4D images</li> </ul>
"Filters (Spatial): Gradient Magnitude"	Generates a strong response at the edges of an objects	<ul style="list-style-type: none"> <li>• All image data types except Complex</li> <li>• 2D, 2.5D, 3D, and 4D images</li> </ul>
"Filters (Spatial): Haralick Texture"	The Haralick texture features are used for image classification. These features capture information about the patterns that emerge in the image texture	Any noncomplex 2D image
"Filters (Spatial): Laplacian"	Detects edges in an image by identifying meaningful discontinuities in a gray level or color	<ul style="list-style-type: none"> <li>• All image data types except Complex</li> <li>• 2D, 2.5D, and 3D images</li> </ul>
Filters (Spatial): Local Normalization	TBD	TBD

**Table 1. MIPAV algorithms and their effects on images (continued)**

Algorithm	Effect	Image types
"Filters (Spatial): Mean"	Provides a simple way of reducing noise, either in an entire image, or in a delineated VOI in the image	<ul style="list-style-type: none"> <li>All image data types except COMPLEX</li> <li>2D and 3D images</li> </ul>
"Filters (Spatial): Median"	Removes shot noise by replacing a target pixel's value with the median value of neighboring pixels	<ul style="list-style-type: none"> <li>Gray-scale 2D, 2.5D, and 3D images</li> <li>Color 2D, 2.5D, and 3D images</li> </ul>
"Filters (Spatial): Mode"	Uses mode filtering (a nonlinear spatial filtering technique) to replace a target pixel with the mode of the neighboring pixels	2D, 2.5D, and 3D images of the following data types: <ul style="list-style-type: none"> <li>BYTE and UBBYTE</li> <li>SHORT and USHORT</li> <li>INTEGER and UINTEGER</li> </ul>
"Filters (Spatial): Nonlinear Noise Reduction"	Reduces noise in an image while preserving both the underlying structure and the edges and corners	Black-and-white 2D, 2.5D, and 3D images
"Filters (Spatial): Nonmaximum Suppression"	Defines the edges of an image	<ul style="list-style-type: none"> <li>Black-and-white 2D, 2.5D, and 3D images</li> <li>Edge processing only applicable to black-and-white 2D images</li> </ul>
"Filters (Spatial): Regularized Isotropic (Nonlinear) Diffusion"	Regularized isotropic nonlinear diffusion. Diffusion filtering, which models the diffusion process, is an iterative approach of spatial filtering	<ul style="list-style-type: none"> <li>All data types except complex</li> <li>2D, 2.5D, and 3D images</li> </ul>
"Filters (Spatial): Slice Averaging"	Reduces image noise	<ul style="list-style-type: none"> <li>All image data types</li> <li>All 2D, 2.5D, and 3D images</li> </ul>
"Filters (Spatial): Unsharp Mask"	Produces a sharpened version of the image or a VOI of the image	<ul style="list-style-type: none"> <li>All image data types, except complex and RGB images</li> <li>2D, 2.5D, 3D, and 4D images</li> </ul>

**Table 1. MIPAV algorithms and their effects on images (continued)**

Algorithm	Effect	Image types
<p>“Filters (Wavelet): De-noising BLS GSM”</p> <p>“Filters (Wavelet): Thresholding”</p>	<p>The modulated window is shifted along the signal, and for every position, the spectrum is calculated. This process is repeated many times with a slightly shorter (or longer) window for every new cycle. The result appears as a collection of time-frequency representations of the signal, all with different resolutions. This solves the signal-cutting problem which arises in the Fourier transform</p>	<ul style="list-style-type: none"> <li>• 2D and 3D datasets</li> <li>• RGB datasets</li> </ul>
<b>Fuzzy C-Means</b>		
<p>“Fuzzy C-Means: Multispectral and Single Channel Algorithms”</p>	<p>Performs both hard and soft segmentation on multiple images</p>	<ul style="list-style-type: none"> <li>• 2D and 3D datasets</li> <li>• RGB datasets</li> </ul>
<b>Histogram tools</b>		
<p>“2D Histogram”</p>	<p>The algorithm takes as an input two grayscale images or one color image, and then creates a 2D histogram image based on the data from two input images</p>	<p>Color and black-and-white images 2D and 3D images</p>
<p>“Cumulative Histogram”</p>	<p>Calculates the cumulative histogram for a chosen image.</p>	<p>2D, 2.5D and 3D grayscale and color (RGB) images. For RGB images the algorithm will display a separate cumulative histogram for each channel (R, G, and B).</p>
<p>“Histogram Equalization: Neighborhood Adaptive”</p>	<p>Enhances the contrast in an image by reevaluating the grayscale or intensity value of each pixel based on a region of nearby pixels</p>	<ul style="list-style-type: none"> <li>• Color image</li> <li>• Black-and-white images</li> </ul>
<p>“Histogram Equalization: Regional Adaptive”</p>	<p>Enhances the contrast in an image</p>	<ul style="list-style-type: none"> <li>• Color images</li> <li>• Black-and-white images</li> <li>• Whole images, not VOIs</li> </ul>
<p>“Histogram Matching”</p>	<p>The algorithm generates an output image based upon a specified histogram</p>	<p>Color and black-and-white 2D and 3D images</p>
<p>“Histogram summary”</p>	<p>Displays frequency distribution information for a chosen image</p>	<p>All image types</p>

**Table 1. MIPAV algorithms and their effects on images (continued)**

Algorithm	Effect	Image types
<b>Image calculator</b>		
"Image Calculator" See also MIPAV Volume1 User Guide	Adds, subtracts, multiplies, and divides, etc. the pixel values of one image by the pixel values of another image. Two images can also be ANDed, ORed or XORed together. More advanced math operators available via the dialog text field	2D and 3D color and grayscale images
"Image Math" See also MIPAV Volume1 User Guide	The algorithm adds, subtracts, multiplies, or divides an image by a user specified value. The square root, absolute value, or log of an image also can be calculated	2D and 3D grayscale images
<b>InsightToolkit (ITK)</b>		
"ITK"	This technical guide explains how to integrate Kitware's InsightToolkit (ITK) with MIPAV	
<b>Levelset tools</b>		
Levelset	TBD	TBD
<b>Mask tools</b>		
"Mask"	Generates a mask, with a specified intensity, of the region inside or outside the contoured VOIs that are delineated on an image	2D, 2.5D, 3D, and 4D images, not RGB images
"Quantify Mask" See also MIPAV Volume1 User Guide	There are two similar algorithms Quantify Mask(s) and Quantify Using Mask that calculate Center of Mass, area (in resolutions), and number of pixels for a selected mask(s)	Algorithms work with Boolean, byte, unsigned byte, and short masks
<b>Microscopy</b>		
"Microscopy: Blind Deconvolution"	Recovers a target object from a set of <i>blurred</i> images in the presence or a poorly determined or unknown Point Spread Function (PSF)	2D and 3D images, color and grayscale
"Microscopy: Colocalization Orthogonal Regression"	Provides an automatic method of quantifying the amount of colocalization in images	Color and black-and-white 2D or 3D images



**Table 1. MIPAV algorithms and their effects on images (continued)**

Algorithm	Effect	Image types
“Microscopy: FRAP (Fluorescence Recovery After Photobleaching)”	Determines an association rate, dissociation rate, and diffusion transfer coefficient in images	Color and black-and-white 3D images
“Microscopy: Colocalization Orthogonal Regression (FRET)—Acceptor Photobleaching”	This algorithm uses acceptor photo bleaching to compare the proximity of fluorescent-labeled molecules in two 2D images over distance	You can apply this algorithm to two 2D images or to a single 2-slice 3D image
“Microscopy: Fluorescent Resonance Energy Transfer (FRET) Bleed Through and Efficiency”	This section provides information on and discusses how to use the following two FRET algorithms: <ul style="list-style-type: none"> <li>• FRET Bleed Through algorithm</li> <li>• FRET Efficiency algorithm</li> </ul>	Three 2D images
“Microscopy (Restoration): Computational Optical Sectional Microscopy (COSM)”	This algorithm removes out-of-focus light in 3D volumes collected plane by plane using either widefield or confocal fluorescence microscopes	All types of 3D microscopy images that can be opened in MIPAV
<b>Morphology:</b>		
Morphology: “Background Distance map”	This operation converts a binary image into an image where every foreground pixel has a value corresponding to the minimum distance from the background	<b>All Morphology operations</b> can be applied to the images of the following types: <ul style="list-style-type: none"> <li>• Boolean—1 bit per pixel/voxel (1 on, 0 off)</li> <li>• Unsigned byte—1 byte per pixel/voxel (0, 255)</li> <li>• Unsigned short—2 bytes per pixel/voxel (0, 65535)</li> </ul>
Morphology: “Close”	Performs Morphology closing for a selected image	
Morphology: “Delete Objects”	Deletes objects larger and or smaller than the indicated maximum/minimum size	
Morphology: “Dilate”	Dilates the image using the user specified structural element	
Morphology: “Distance Map”	The algorithm uses the Euclidean distance metric to calculate a distance map for a selected image or image region	
Morphology: “Erode”	Erodes the image using the user specified structural element	
Morphology: “Evaluate Segmentation”	Compares segmentation results of a test image to segmentation results of an ideal gold standard true image	

**Table 1. MIPAV algorithms and their effects on images (continued)**

Algorithm	Effect	Image types
Morphology: "Fill holes"	Fills holes in a selected image	
Morphology: "Find Edges"	Finds the edges of the objects in an image using combinations of the following Morphology operations: dilation, erosion and XOR	
Morphology: "ID objects"	The algorithm labels each object in a selected image with a different integer value and also deletes objects which are outside of the user defined threshold	
Morphology: "Morphological filter"	It corrects a selected image for non-uniform illumination and non-uniform camera sensitivity	
Morphology: "Open"	Perform Morphology opening of the selected image using the user specified structural element	
Morphology: Particle Analysis	Generates the information on the particle composition for the binary images	
Morphology: "Skeletonize"	Skeletonizes the image by means of a lookup table, which is used to repeatedly remove pixels from the edges of objects in a binary image, reducing them to single pixel wide skeletons	
Morphology: "Skeletonize 3D pot field"	The algorithm uses an iterative approach to simultaneously produce a hierarchical shape decomposition of a selected image and create a corresponding set of multi-resolution skeletons.	
Morphology: "Ultimate erode"	Generates the ultimate eroded points (UEPs) of an Euclidian distance map for an image	
<b>End of Morphology</b>		
<b>Muscle segmentation</b>		
"Muscle Segmentation" "Abdomen Segmentation"	This is a semi automatic tool for segmenting different muscles and muscles and fat in Computed Tomography (CT) images of the thigh and abdomen	Any CT image of the thighs and abdomen that can currently be opened by MIPAV

**Table 1. MIPAV algorithms and their effects on images (continued)**

Algorithm	Effect	Image types
Noise	TBD	TBD
Plot Surface	TBD	TBD
Principal Component	TBD	TBD
“Randomizing image (slice) order” see Volume 1, Utilities	Randomizes the order of slices in the image dataset	3D images
<b>Registration</b>		
Registration: AFNI-Shear	TBD	TBD
Registration: AIR Linear	TBD	TBD
Registration: AIR Nonlinear	TBD	TBD
Registration: “B-Spline Automatic Registration”	B-spline based registration of images in two and three dimensions	2D and 3D color (RGB) and grayscale images
“Registration: Patient Position (DICOM)” TBD	The method uses the image origin coordinates and image orientations to align DICOM images based on a patient position	DICOM images
“Registration: Display Pixel Similarity Cost Functions”	The algorithm calculates costs for various voxel similarity cost functions that are used in registration and output them to the Output window	Color, grayscale, and black and white 2D and 3D images
“Registration: Landmark—Least Squares”	Registers an image to the reference image by using corresponding points placed in both images	2D and 3D gray-scale and color images
“Registration: Landmark—TPSpline”	Registers two images in a nonlinear manner using the corresponding landmark points that you delineate in both of the images	All 2D and 3D images
“Registration: Manual 2D Series”	Registers manually or semimanually two images by placing corresponding points on both images and then, applying either the Landmark—Thin Plate Spline or Landmark—Least Squares algorithm	All 2D gray-scale and color images

**Table 1. MIPAV algorithms and their effects on images (continued)**

Algorithm	Effect	Image types
Registration: "Midsagittal Line Alignment"	Used for the automatic detection of the midsagittal line in 2D and midsagittal plane in 3D brain images	Color and black-and-white 3D images
Registration: "Mosaic Registration"	This is a user interface for a semi-manual image registration. It allows the user, first, manually aligns two images using a mouse, and then it calls the Optimized Automatic Registration 2D algorithm for further precise registration	Color, grayscale, and black and white 2D images
Registration: "NEI Build MP maps plug in"	The method includes the measurements of MP that are performed by analysis of autofluorescence (AF) images and building the MP maps	2D retinal images
Registration: "Optimized Automatic Registration 3D" and 2.5 D	Determines a transformation that minimizes a cost function, which represents the quality of alignment between two images	Color, grayscale, and black and white 3D images
"Registration: Time Series Optimized Automatic Registration"	Registers individual 3D volumes within a 4D time series to a reference volume	Both 4D color and gray-scale images
Registration: "Reslice—Isotropic Voxels"	Applies image resampling	All image data types except RGB and complex
<b>Shading correction</b>		
"Shading Correction: Inhomogeneity N3 Correction"	Corrects for shading artifacts often seen in MRI	2D and 3D MRI images
<b>Talairach Space</b>		
"Labeling and Measuring Brain Components in Talairach Space"	How to use the TalairachTransformation wizard and the FANTASM (Fuzzy and Noise Tolerant Adaptive Segmentation Method) plug-in programs, which were developed by the Johns Hopkins University, with MIPAV	
<b>Threshold</b>		

**Table 1. MIPAV algorithms and their effects on images (continued)**

Algorithm	Effect	Image types
“Standard Deviation Threshold”	Standard Deviation Threshold works by, first, examining an active VOI for which the standard deviation (st.dev) of pixel intensities and other statistics are calculated. The algorithm, then, thresholds an image using the user defined parameters, such as a number of standard deviations and/or values outside of the range	2D, 3D, and 4D color (RGB) and grayscale images
“Threshold”	The algorithm replaces the image pixel values with the fill values specified by a user. The pixel values change or remain unchanged depending on whether the original color value is within the threshold range	All 2D, 3D, and 4D color (RGB) and grayscale images
Stereo Depth	TBD	TBD
“Subsampling images”	Reduces an image in size by a chosen factor of 2, 4, or 8 times	All image types
<b>Subtract</b>		
Subtract VOI Background	TBD	TBD
<b>Transformation</b>		
“Transform to power of 2”	The algorithm resamples the original image to dimensions that are powers of 2	Color and black-and-white 2D and 3D images
“Transform”	Offers multiple options that help a user to define the transformation matrix, and then execute the transformation and (or) resampling	Color and black-and-white 2D, 2.5D, 3D, and 4D images

**Table 1. MIPAV algorithms and their effects on images (continued)**

Algorithm	Effect	Image types
"Transform: Conformal Mapping Algorithms" : <ul style="list-style-type: none"> <li>• "Circular Sector to Rectangle"</li> <li>• "Transformation: Circle to Rectangle"</li> <li>• "Transformation: Ellipse to Circle"</li> <li>• "Transformation: Nearly Circular Region to Circle"</li> </ul>	The methods described in this document use conformal mapping to transform points in a circular sector, circle, ellipse, or nearly circular region to points in a circle or rectangle	All ultrasound images that can be opened in MIPAV and also 2D color and black and white images
Transform Nonlinear	The algorithm takes a source image and uses information read in from a nonlinear transformation (.nlt) file to perform a nonlinear B-Spline transformation on the image	Color (RGB) and grayscale 2D and 3D images
<b>Watershed</b>		
Watershed	This is an interactive algorithm that allows automatically segment the image regions of interest using the topographical approach	2D and 3D grayscale images

---

## Cost functions used in MIPAV algorithms

A *similarity* or *cost function* measures the similarity between two images. During the registration the adjusted image  $V$  is transformed using the various transformation functions. And the similarity  $S(U;V^t)$  between the reference image  $U$  and transformed image  $V^t$  is then calculated.

We assume that the transformation that gives the smallest value of the chosen inverse cost function is the transformation that also gives the best alignment.

---

In this MIPAV User Guide we use the term *cost function* to refer to the negative cost function.

---

## Background

For the registration algorithms, such as OAR, the main approach for determining an optimal transformation is to

- Calculate a *cost function*,
- Determine how it changes as individual transformation parameters are varied,
- And find the parameters that minimize the value of the cost function.

For instance, Figure 1 shows a plot of a sample cost function for a selection of transformation parameters; in each plot, a single parameter is varied while all other are kept constant.

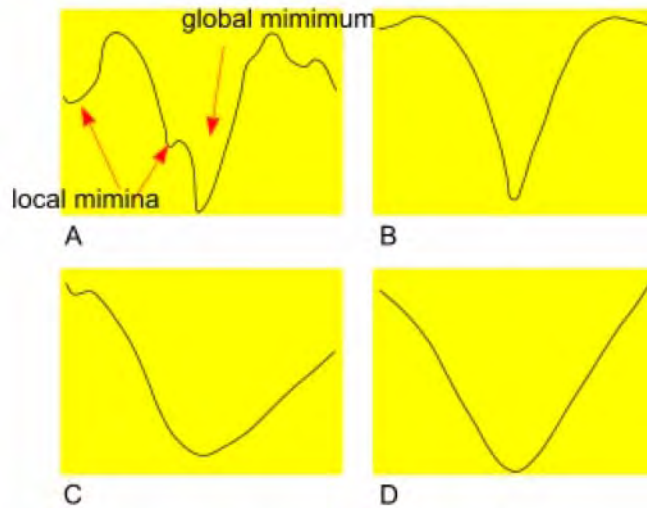


Figure 1. The plots of the Correlation Ratio cost function versus some of the individual parameter values. In each plot (A, B, C, and D), a single parameter is varied while the other are kept constant

The cost functions implemented in MIPAV:

- Correlation ratio, refer to page 64
- Least Squares, refer to page 66
- Normalized Cross Correlation, on page 67
- Normalized Mutual Information, see page 70

## POWELL ALGORITHM

In MIPAV, the most of the methods and algorithms use the Powell algorithm to find the global minimum of the chosen cost function. For more information about the Powell algorithm, refer to <http://math.fullerton.edu/mathews/n2003/PowellMethodMod.html>

## CORRELATION RATIO

Given two images  $I$  and  $R$ , the basic principle of the Correlation Ratio method is to search for a spatial transformation  $T$  and an intensity mapping



$f$  such that, by displacing  $R$  and remapping its intensities, the resulting image  $f(R^* T)$  be as similar as possible to  $I$ . This could be achieved by minimizing the following correlation ratio function:

EQUATION 1

$$\min(T, f) \text{ of } \sum_k \{I(x_k) - f(R(T(x_k)))\}$$

which integrates over the voxel positions  $x_k$  in the image  $I$ .

### Image types

Correlation Ratio can be used in multimodal image registration, e.g. that involves Magnetic Resonance (MR), Computed Tomography (CT), and Positron Emission Tomography (PET) images.

---

## LAPLACIAN ZERO-CROSSING

The laplacian zero-crossing is a binary edge feature used for edge detection, see also “Edge Detection: Zero X Laplacian” . Convolution of an image with a laplacian kernel approximates the 2-nd partial derivative of the image. The laplacian image zero-crossing corresponds to points of maximal (or minimal) gradient magnitude. Thus, laplacian zero-crossings represent “good” edge properties and should, therefore, have a low local cost meaning edge detection. If  $I_L(q)$  is the laplacian of an image  $I$  at pixel  $q$ , then

EQUATION 2

$$fz(q) = \begin{cases} 0 & ; \text{ if } I_L(q) = 0 \\ 1 & ; \text{ if } I_L(q) \neq 0 \end{cases}$$

Though, application of a discrete laplacian kernel to a digital image produces very few zero-valued pixels. Rather, a zero-crossing is represented by two neighboring pixels that change from positive to negative values. Of these two neighboring pixels, the one closest to zero is used to represent the zero-crossing. The resulting feature cost contains single-pixel wide cost “canyons” used for boundary localization.

## GRADIENT MAGNITUDE

Gradient Magnitude provides a direct correlation between the edge strength and local cost. If  $I_x$  and  $I_y$  represent the partials of an image  $I$  in  $X$  and  $Y$  directions respectively, then the gradient magnitude  $G$  is approximated with

EQUATION 3

$$G = \sqrt{I_x^2 + I_y^2}$$

The gradient is then scaled and inverted, so high gradients produce low costs and vice-versa. Thus, the gradient component function is

EQUATION 4

$$f_G = \frac{\max(G) - G}{\max(G)} = 1 - \frac{G}{\max(G)}$$

Finally, gradient magnitude costs can be scaled by Euclidean distance. To keep the resulting maximum gradient at unity,  $f_G(q)$  can be scaled by 1 if  $q$  is a diagonal neighbor to  $p$  and by  $1/\sqrt{2}$  if  $q$  is a horizontal or vertical neighbor.

## LEAST SQUARES

Least squares measures the average of the squared difference in image intensities.

EQUATION 5

$$\frac{\sum_{i=1}^N \{R(p_i) - I(p_i)\}^2}{N}$$

Where,

$R(p_i)$  = reference image( $p_i$ ) - minimum reference image value

$I(p_i)$  = input image( $p_i$ ) - minimum input image value

$N$  = the number of values over which the sum is performed

It can be divided by count to normalize the cost function or make it invariant to the number of voxels in the region of overlap. If two images show large differences in image intensity, then we recommend using Scaled Least Squares with a global scaling term added, refer to Equation 6:

EQUATION 6

$$\frac{\sum_{i=1}^N \{R(p_i) - sI(p_i)\}^2}{N}$$

Where  $s$  is a global scaling factor.

### Image types

It can be shown that Least Squares is the optimum cost function when two images only differ by Gaussian noise. Images in two different modalities, such as PET and MRI, will never differ by only Gaussian noise. Even two images in the same modality, such as two PET images, will seldom only differ by Gaussian noise as medical scan noise is frequently not Gaussian and because the object often changes between the image acquisitions. The effectiveness of this cost function will be greatly diminished by small numbers of voxels having large intensity differences.

---

## NORMALIZED CROSS CORRELATION

The Cross-Correlation function can be described as

EQUATION 7

$$CrossCorr(s, t) = \sum_x \sum_y R(x, y) I(x - s, y - t)$$

Where,

R—reference image intensity

I—input image intensity

And the summation is taken over the region (s,t) where R and I overlap. For any value of (s,t) inside R(x,y) the cross-correlation function yields one value of CrossCorr. The maximum value of CrossCorr(s,t) indicates the position where I(x,y) best matches R(x,y).

The correlation ratio ranges from 0 for very good registrations to 1 for very bad registrations.

### Normalized cross correlation

Normalized Cross Correlation – (NCC)=  $var(RI)/\sqrt{varR * varI}$ . Where,

*var* is the variance

$$varRI = \frac{\sum(RI)}{(count-1)} - \frac{(\sum R * \sum I)}{((count-1)*count)}$$

$$varR = \frac{\sum(R^2)}{(count-1)} - \frac{(\sum R * \sum R)}{((count-1)*count)}$$

$$varI = \frac{\sum(I^2)}{(count-1)} - \frac{(\sum I * \sum I)}{((count-1)*count)}$$

R = (the reference image value - reference image minimum value);

I = (the input image value - input image minimum value);

count is a total number of values.

For correlation ratio:

*First, go over all pixels forming numY, sumY, and sumY2:*

```
indexValue = (int)( (reference image[index] - minimum
reference image value) * constant);
```

```
numY[indexValue] = numY[indexValue] + 1
```

```
sumY[indexValue] = sumY[indexValue] + input image
value - minimum input image value
```

```
sumY2[indexValue] = sumY2[indexValue] + (input
image value - minimum input image value)2
```

*Now calculate the individual variances*

For each iso-set weighting them by the number of pixels from the input image that contribute.

```
for (int b=0; b < nBins; b++)
```

```
{
```

```
    numTotY = numTotY + numY[b]
```

```
totSumY = totSumY + sumY[b]
```

```
totSumY2 = totSumY2 + sumY2[b]
```

*The following should be the variance of the b-th iso-subset*

```
variance = (sumY2[b] - sumY[b]*sumY[b]/
numY[b]) / ( numY[b]-1)
```

```
corrRatio = corrRatio + variance * numY[b]
```

```
}
```

*Normalize the weighting of numY[]*

```
corrRatio = corrRatio/numTotY
```

*Calculate the total variance of input image and then normalize by this*

```
variance = ( totSumY2 - totSumY * totSumY /
numTotY ) / (numTotY - 1)
```

```
corrRatio = corrRatio/variance
```

More simply the above steps can be expressed as

EQUATION 8

$$NCC = \frac{1}{var(Y)} \sum_b \frac{numY(b)}{numTotY} varY(b)$$

## Image types

The Cross-Correlation function can be used for aligning the images that have a linear relationship between the intensity values in the images. E.g. images acquired using the same modality. However, the Normalized Cross-Correlation may not work very well in two images that are identical except for a global scaling factor.

The Normalized Cross Correlation is 1 for identical images and approach 0 for images that are very different.

---

## NORMALIZED MUTUAL INFORMATION

Mutual information (MI) measures how well one image explains the other. In medical image processing, it is often used as 1) a similarity measure for image registration for images that were acquired at different times or by different modalities and 2) also for combining multiple images to build 3D models. The mutual information (MI) of random variables A and B is defined as:

EQUATION 9

$$MI(A, B) = \sum_{ab} p(a, b) \log \frac{p(a, b)}{p(a)p(b)}$$

Where,  $p(a,b)$  is the joint probability distribution function of A and B, and  $p(a)$  and  $p(b)$  are the marginal probability distribution functions of A and B respectively.

Equation 9 can be also rewritten as  $MI(A, B, C) = H(A) + H(B) - H(A, B)$ ,

where  $H(A)$  and  $H(B)$  are the Shannon entropies of image A and B respectively computed on the probability distributions of their grey values. And  $H(A, B)$  denotes the conditional entropy, which is based on the conditional probabilities  $p(a | b)$  – the chance of grey value a in image A given that the corresponding voxel in B has grey value b. MI measures the distance between the joint distributions of the images' gray values  $p(a, b)$  and the distribution when assume that images are independent from each other.

**Normalized mutual information** can be calculated as  $NMI(A, B) = (H(A) + H(B)) / H(A, B)$ .

### Image types

The normalized mutual information has been shown to work very well for registering multi-modality images and also time series images. The normalized mutual information approaches 0 for identical images and approaches 1 for images that are very different.

## DEGREES OF FREEDOM

The number of independent pieces of information that go into the estimate of a parameter is called the degrees of freedom (DOF). In general, the degrees of freedom of an estimate is equal to the number of independent scores that go into the estimate minus the number of parameters estimated as intermediate steps in the estimation of the parameter itself. In image registration, a transformation matrix establishes geometrical correspondence between coordinate systems of different images. It is used to transform one image into the space of the other. The following transformations are generally used in biomedical imaging:

- <sup>1</sup>Rigid-body transformations include translations and rotations. They preserve all lengths and angles. These are 6 DOF transformation, and the transformation matrix is as follows:

EQUATION 10

$$\begin{bmatrix} R_x & R_y & R_z \\ T_x & T_y & T_z \end{bmatrix}$$

Where  $R_x$ ,  $R_y$ , and  $R_z$  represent rotations and  $T_x$ ,  $T_y$ ,  $T_z$  – translations.

- Global rescale transformations include translations, rotations, and a single scale parameter  $S=S_x=S_y=S_z$ . They preserve all angles and relative lengths. These are 7DOF transformations and the transformation matrix is as follows:

EQUATION 11

$$\begin{bmatrix} R_x & R_y & R_z \\ T_x & T_y & T_z \\ S \end{bmatrix}$$

- Affine transformations include translations, rotations, scales, and/or skewing parameters. They preserve straight lines but necessarily not

1. Here, DOF are given for 3D images.

angles or lengths. Transformation matrixes for affine transformations are as follows:

9 DOF transformation matrix which includes scale parameters  $S_x$ ,  $S_y$  and  $S_z$ :

EQUATION 12

$$\begin{bmatrix} R_x & R_y & R_z \\ T_x & T_y & T_z \\ S_x & S_y & S_z \end{bmatrix}$$

12 DOF transformation matrix which includes both scale and skew parameters. In the matrix, skewing parameters are presented as  $Sk_x$ ,  $Sk_y$ , and  $Sk_z$ :

EQUATION 13

$$\begin{bmatrix} R_x & R_y & R_z \\ T_x & T_y & T_z \\ S_x & S_y & S_z \\ Sk_x & Sk_y & Sk_z \end{bmatrix}$$

---

## REFERENCES

Björn Hamre "Three-dimensional image registration of magnetic resonance (MRI) head volumes" Section for Medical Image Analysis and Informatics Department of Physiology & Department of Informatics University of Bergen, Norway.

Chapter 33 "Within-Modality Registration Using Intensity-Based Cost Functions" by Roger P. Woods in *Handbook of Medical Image Processing and Analysis*, Editor, Isaac N. Bankman, Academic Press, 2000, pp. 529-553.

Mortensen E., Barrett W., "Intelligent scissors for image composition", International Conference on Computer Graphics and Interactive Techniques archive. Proceedings of the 22-nd annual conference on Computer graphics and interactive techniques, pp. 191 - 198, 1995, ISBN:0-89791-701-4.

Josien P. W. Pluim, J. B. Antoine Maintz and Max A. Viergever. *Mutual information based registration of medical images: a survey*. IEEE TRANSACTIONS ON MEDICAL IMAGING, VOL. XX, NO. Y, MONTH 2003.

Powell method to find the global minimum: <http://math.fullerton.edu/mathews/n2003/PowellMethodMod.html>





## Interpolation methods used in MIPAV

*The registration and transformation algorithms implemented in MIPAV resamples images (if needed) using an interpolation scheme, where anisotropic voxels (or pixels for 2D images) are resampled into isotropic 1 mm cubic voxels. New voxel values are computed using a weighted combination of existing voxel values within a defined neighborhood of the new voxel location. The following interpolation methods are available in this implementation of the registration technique, including Bilinear, Trilinear, B-spline 3-rd order, B-spline 4-th order, Cubic Lagrangian, Quintic Lagrangian, Heptic Lagrangian, and Windowed sinc.*

### Bilinear Interpolation

Bilinear interpolation considers the closest 2x2 neighborhood of known pixel values surrounding the unknown pixel. It then takes a weighted average of these 4 pixels to arrive at its final interpolated value. This results in smoother looking images.

Figure 1 is for a case when the known pixel distances are not equal, therefore the interpolated value is calculated using the Equation 1.– Equation 4.

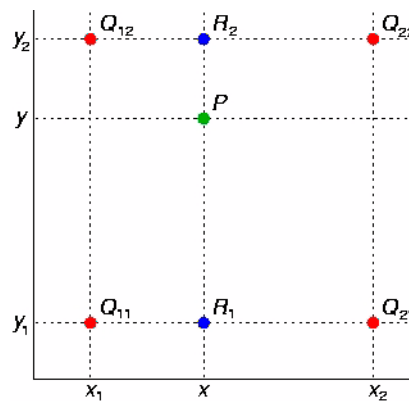


Figure 1. The red dots show the known pixels and the green dot is the pixel we wish to interpolate

First, we will do a linear interpolation in X direction and calculate values for blue pixels R1 and R2:

EQUATION 1

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \quad \text{where } R_1 = (x, y_1),$$

EQUATION 2

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \quad \text{where } R_2 = (x, y_2).$$

Then, we will do interpolation in Y direction:

EQUATION 3

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2).$$

This gives us the desired interpolated value for  $P=f(x,y)$

EQUATION 4

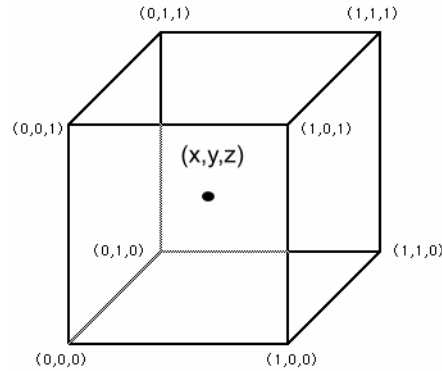
$$\begin{aligned} f(x, y) \approx & \frac{f(Q_{11})}{(x_2 - x_1)(y_2 - y_1)} (x_2 - x)(y_2 - y) \\ & + \frac{f(Q_{21})}{(x_2 - x_1)(y_2 - y_1)} (x - x_1)(y_2 - y) \\ & + \frac{f(Q_{12})}{(x_2 - x_1)(y_2 - y_1)} (x_2 - x)(y - y_1) \\ & + \frac{f(Q_{22})}{(x_2 - x_1)(y_2 - y_1)} (x - x_1)(y - y_1). \end{aligned}$$

## Trilinear Interpolation

Trilinear interpolation computes values located between existing voxel values by linearly weighting the eight closest neighboring values. Specifically, Figure 2 depicts the situation where it is desired to determine an intensity value at the point labeled (x, y, z), given the values at the corner of the cube. The unknown value is computed by combining the known corner values weight by their distance from the location of the point (x, y, z) according to the following formula:

EQUATION 5

$$V_{xyz} = V_{000} (1 - x) (1 - y) (1 - z) + V_{100} x (1 - y) (1 - z) + V_{010} (1 - x) y (1 - z) + V_{001} (1 - x) (1 - y) z + V_{101} x (1 - y) z + V_{011} (1 - x) y z + V_{110} x y (1 - z) + V_{111} x y z$$



**Figure 2. Trilinear interpolation**

Trilinear interpolation operates in a parameter space with dimension  $D = 3$  of order  $n = 1$ , thus requiring  $(1 + n)^3 = 8$  adjacent pre-defined values surrounding the interpolation point.

---

Trilinear interpolation is the default resampling interpolation method used in this registration technique.

---

## B-spline basis interpolations

B-spline interpolation utilizes weighted voxel values in a larger neighborhood compared to trilinear interpolation. The larger number of values are weighted based on a 3-rd and 4-th order of polynomial, instead of a second order polynomial as in the case of trilinear interpolation. B-spline interpolation refers to the locations of the neighboring points as control points and combines the intensity values at these locations using a set of polynomial basis according to Equation 6.

The equation for k-order B-spline with  $n+1$  control points ( $P_0, P_1, \dots, P_n$ ) is

EQUATION 6

$$P(t) = \sum_{i=1}^{n+1} N_{i,k}(t) P_i, \quad t_{min} \leq t < t_{max}$$

In Equation 6,  $N_{i,k}$  are the polynomial functions of degree  $k-1$ , and  $n$  is the number of control points. Note, that the degree of the weighting polynomial is independent of the number of control points,  $n$ .

The weighting polynomial is recursively defined as

EQUATION 7

$$N_{i,1}(t) = \begin{cases} 1, & t_i \leq t < t_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

$$N_{i,k} = \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t)$$

Here,  $t(i)$  represents an index that refer to the control points and  $t(i)$  are generally referred as *knot points*, see Figure 3. The sequence of knot points is also called a *knot vector*. This indexing scheme allows one to weight different control points more than other control points by using it more than once during the computation. Typically, the first and last control points are weighed more heavily than the internal points to provide a smooth interpolating curve. The Optimized Automatic Registration method uses *the open uniform knot vectors* for calculating B-splines. Open uniform knot vectors are vectors that have  $k$  equal knot values at each end, such as described in Equation 8.

EQUATION 8

$$\begin{aligned} t_i &= t_1, & i &\leq k \\ t_{i+1} - t_i &= \text{constant}, & k &\leq i < n + 2 \\ t_i &= t_{k+(n+1)}, & i &\geq n + 2 \end{aligned}$$

For example: [0,0,0,0,1,2,3,4,4,4,4,] for  $k=4$ ; [1, 1, 1, 2, 3, 4, 5, 6, 6, 6] for  $k=3$ ; [0.1, 0.1, 0.1, 0.1, 0.1, 0.3, 0.5, 0.7, 0.7, 0.7, 0.7, 0.7] for  $k=5$ .

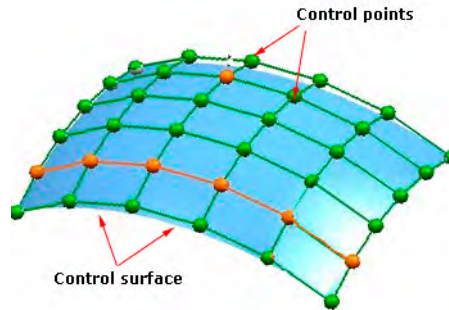


Figure 3. B-spline interpolation

## Lagrange interpolations

To obtain higher-order interpolations, the Optimized Automatic Registration algorithm uses Lagrange interpolations. The Lagrange kernel of degree  $N-1$  for  $N \times N$  region can be defined by Equation 9.

**Note:** Options provided by this method include the following Lagrange interpolations: Cubic ( $N = 3$ ), quintic ( $N = 5$ ), and heptic ( $N = 7$ ).

EQUATION 9

$$Lagra_{h_N}(x) = \begin{cases} \prod_{j=0, j-N/2+1 \neq n}^{N-1} \frac{n-i-x}{n-i}, & n-1 \leq x < n \\ = 0, & elsewhere \end{cases}$$

Where  $i = j - N/2 + 1$ , and  $n \in \{-N/2 + 1, -N/2 + 2, N/2\}$

The Lagrange kernel for  $N=1$  is nearest neighbor interpolation. In the case  $N=2$  it is linear interpolation. The Lagrange kernel for  $N=4$  supporting points results in cubic polynomials. See Equation 10:

EQUATION 10

$$\text{Lagra}_{h_4}(x) = \begin{cases} \frac{1}{2}x^3 - x^2 - \frac{1}{2}x + 1, & \text{for } 0 \leq x < 1 \\ -\frac{1}{6}x^3 + x^2 - \frac{11}{6}x + 1, & \text{for } 1 \leq x < 2 \\ 0, & \text{elsewhere} \end{cases}$$

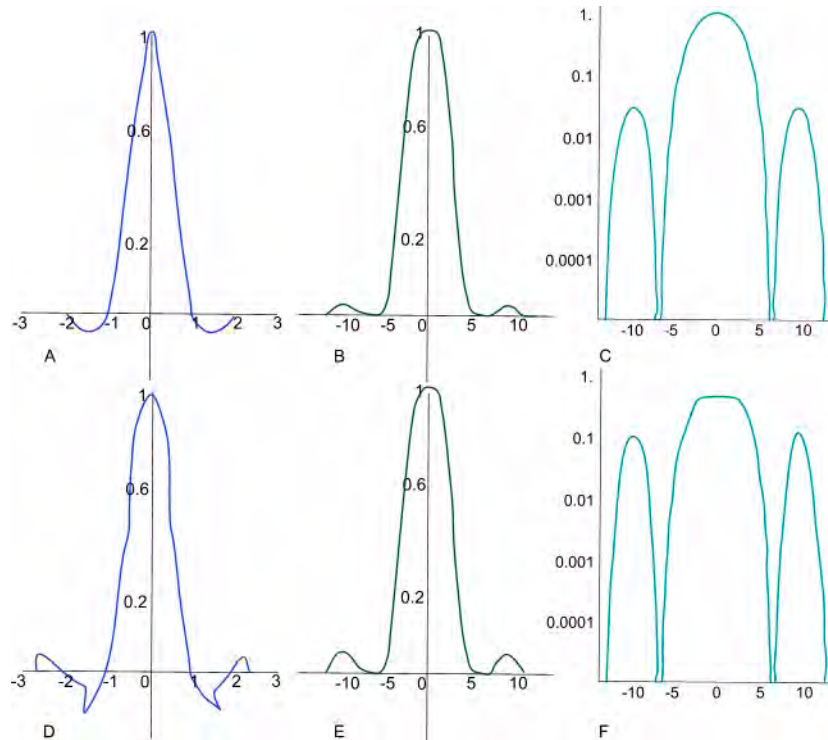
The Lagrange kernel for N=5 supporting points results in quadratic polynomials as shown in Equation 11 below:

EQUATION 11

$$\text{Lagra}_{h_5}(x) = \begin{cases} \frac{1}{4}x^4 - 54x^2 + 1, & \text{for } 0 \leq x < 1/2 \\ -\frac{1}{6}x^4 + \frac{5}{6}x^3 - \frac{5}{6}x^2 - \frac{5}{6}x + 1, & \text{for } 1/2 \leq x < 3/2 \\ \frac{1}{24}x^4 - \frac{5}{12}x^3 + \frac{35}{24}x^2 - \frac{25}{12}x + 1, & \text{for } 3/2 \leq x < 5/2 \\ 0, & \text{elsewhere} \end{cases}$$

All Lagrange kernels are direct current (DC) constant. In other words, the mean brightness of the image is not affected if the image is interpolated or re-sampled. However, the even Lagrange interpolators do not fit C1-continuously at the connecting points causing significant sidelobes within their Fourier transforms. Figure 4 below shows the Lagrange kernels for N=4 and N=5, respectively.

- $\text{Lagra}_{h_4}(x)$  shows a sharp edge at the center of the mask in the spatial domain. The amplitude of the sidelobe in the Fourier domain is about 4%.
- The odd Lagrange kernels, such as  $\text{Lagra}_{h_5}(x)$ , is not C0-continuous.



**Figure 4.** Lagrange third-order interpolation,  $N = 4$ : (A) - Kernel, (B) - Magnitude of Fourier transform, (C) - Logarithmic plot of magnitude. Lagrange fourth-order interpolation,  $N = 5$ , (D) -Kernel, (E) - Magnitude of Fourier transform, (F) - Logarithmic plot of magnitude

Therefore, the amplitude of the sidelobe is raised up to 10% as shown in Figure 4-F. However, the plateau in the passband is wider causing the major lobe to approximate more closely the ideal rectangular shape. In other words, the passband characteristic is improved by raising the order of the Lagrange kernel. Therefore, odd Lagrange kernels, such as cubic ( $N = 3$ ), quintic ( $N = 5$ ), and heptic ( $N = 7$ ), should be used for images with high contrasts.

## Gaussian

The Gaussian function is as follows:

EQUATION 12

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} * \exp -\frac{x^2}{2\sigma^2}$$



Where ***sigma*** is the standard deviation of the distribution. We assume that the distribution has a mean of zero (i.e. it is centered on the line  $x=0$ ). The distribution is illustrated in Figure below.

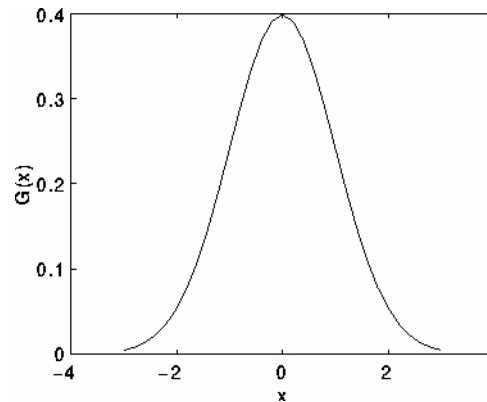


Figure 5. 1-D Gaussian distribution with mean 0 and  $\sigma=1$

In 2-D, an isotropic (i.e. circularly symmetric) Gaussian has the form:

EQUATION 13

$$G(x) = \frac{1}{\sqrt{2\pi\sigma}} * \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

The idea of Gaussian interpolation is to use the 2-D distribution as a point-spread function, and this is achieved by convolution. Since the image is stored as a collection of discrete pixels the algorithm, first, produces a discrete approximation to the Gaussian function before it performs the convolution.

In theory, the Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution kernel, but in practice it is effectively zero more than about three standard deviations from the mean, and so the algorithm truncates the kernel at this point. Figure 6 shows a suitable integer-valued convolution kernel that approximates a Gaussian with a  $\sigma$  of 1.0.

$$\frac{1}{273}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

**Figure 6. Discrete approximation to Gaussian function with sigma=1.0**

Once a suitable kernel has been calculated, then the Gaussian interpolation can be performed using standard convolution methods. The convolution can be performed fairly quickly since Equation 13 for the 2-D isotropic Gaussian shown above is separable into X and Y components. Therefore, the 2-D convolution can be performed by, first, convolving with a 1-D Gaussian in the Y direction, and then, convolving with another 1-D Gaussian in the X direction. (The Gaussian is the only completely circularly symmetric operator which can be decomposed in such a way.) A further way to compute a Gaussian interpolation with a large standard deviation is to convolve an image several times with a smaller Gaussian.

---

The Gaussian function is a recurrent with respect to operations such as derivation and Fourier transform.

---

## Windowed sinc

The Windowed *sinc* function is a product of the *sinc* function with a window function of a limited spatial support. The *sinc* function can be defined as shown in Equation 14:

EQUATION 14

$$\text{sinc}(x) = \begin{cases} \frac{\sin(x)}{x}, & \forall x \neq 0 \\ 1, & x = 0 \end{cases}$$

Consider an image data set consisting of a 3D matrix of voxels with intensities  $I(X; Y; Z)$  identified by integer position coordinates  $(X; Y; Z)$ . If one

wants to calculate the intensity value at an interior point defined by non-integer coordinates (x; y; z), this can be achieved by forming the following triple sum, see Equation 15:

EQUATION 15

$$I(x, y, z) = \sum_X \sum_Y \sum_Z I(X, Y, Z) \text{sinc}(\pi(x - X)) \text{sinc}(\pi(y - Y)) \text{sinc}(\pi(z - Z))$$

There are two limiting conditions that the function I(x, y, z) must satisfy:

- I(x, y, z) must be bandlimited. In other words, the function must have a *Fourier transform*  $F\{I(x, y, z)\} = I(f) = 0$  for  $|f| > B$  for some maximum frequency  $B > 0$ .
- The sampling rate  $-f_s$ , must exceed twice the bandwidth, e.g.  $f_s > 2B$ .

The interpolation formula reconstructs the original signal, I(interior), as long as these two conditions are met.

Equation 15 implies a sum over all voxels for each new intensity value calculated. Since this is computationally very demanding and since distant voxels (for which *sinc* weighting is small) make a little contribution, it is convenient in practical situations to truncate the interpolation series. This can be done by limiting the calculation to a cubic sub-volume containing  $(2R+2)^3$  voxels centered on the target point (x; y; z) for which an intensity is calculated. Here, R determines the size of the cubic box, and when R is zero only the eight nearest neighbors will be used in the calculation.

Because of the oscillating nature of the *sinc* function, some truncation errors and negative values may result from this procedure. To reduce the truncation error, each *sinc* function is multiplied by an apodizing the Hann window function, refer to Equation 16:

EQUATION 16

$$H(\Delta; R) = \frac{1}{2} \left[ 1 + \cos \left( \frac{\pi \Delta}{R+1} \right) \right]$$

Where,  $\Delta$  can be replaced by (x-X), (y-Y) or (z-Z) as appropriate.

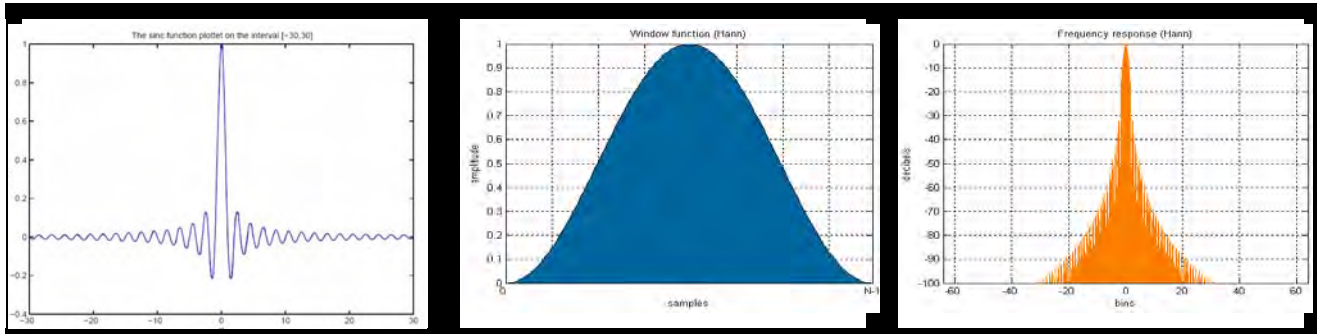


Figure 7.  $T y = \text{sinc}(a)$ , where  $a = px, x \in [-30, 30]$ ; a sample Hann function and its frequency response. Here,  $N$  represents the width, in samples, of a discrete-time window function. Typically, it is an integer power-of-2, such as  $2^{10} = 1024$ .

## REFERENCES

Bourke Paul "Interpolation methods" <http://local.wasp.uwa.edu.au/~pbourke/other/interpolation/index.html>

B-splines: <http://www.cl.cam.ac.uk/teaching/1999/AGraphHCI/SMAG/node4.html>

Chapter 3: "Registration Methodology: Concepts and Algorithms" by Derek L.G.Hill and Philippe Batchelor in *Medical Image Registration* edited by Joseph V. Hajnal, Derek L.G.Hill, and David J. Hawkes, CRC Press, 2001, pp. 40-70.

Chapter 33 "Within-Modality Registration Using Intensity-Based Cost Functions" by Roger P. Woods in *Handbook of Medical Image Processing and Analysis*, Editor, Isaac N. Bankman, Academic Press, 2000, pp. 529-553.

Thomas M. Lehmann,\* Member, IEEE, Claudia Gonner, and Klaus Spitzer. *Survey: Interpolation Methods in Medical Image Processing*. IEEE TRANSACTIONS ON MEDICAL IMAGING, VOL. 18, NO. 11, NOVEMBER 1999 1049.

---

## Autocorrelation Coefficients

The algorithm calculates autocorrelation coefficients for images.

### Background

In any time series containing non-random patterns of behavior, it is likely that any particular item in the series is related in some way to other items in the same series. This can be described by the autocorrelation function and/or autocorrelation coefficient.

The correlation of two continuous functions  $f(x)$  and  $g(x)$  can be defined as

EQUATION 1

$$f(x) \bullet g(x) = \int (f(a) \cdot g(x+a)) dx$$

where, “ $\cdot$ ” represents the complex conjugate. If  $f(x)$  and  $g(x)$  are the same functions,  $f(x) \bullet g(x)$  is called autocorrelation function.

For a 2D image, its autocorrelation function (ACF) can be calculated as

EQUATION 2

$$f(x, y) \bullet g(x, y) = \int (f(a, b) \cdot g(x+a, y+b)) dx$$

where  $f(x,y)$  is the two-dimensional brightness function that defines the image, and  $a$  and  $b$  are the variables of integration. Like the original image, the ACF is a 2D function. Although the dimensions of the ACF and the original image are exactly the same, they have different meaning. In the original image, a given coordinate point  $(x,y)$  denotes a pixel position, while in the ACF, a given coordinate point  $(a,b)$  denotes the endpoint of a neighborhood vector.

The ACF describes how well an image correlates with itself under conditions where the image is displaced with respect to itself in all possible directions.

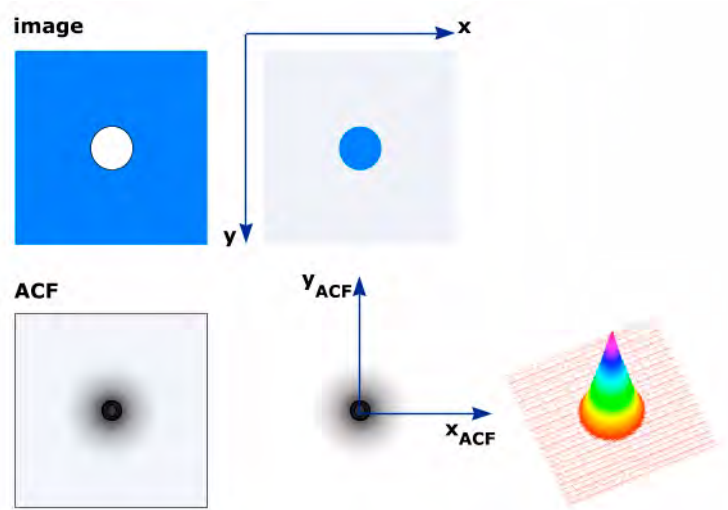


Figure 1. Visualization of ACF

For a given series of points  $\{x_i\}$ , the autocorrelation coefficient can be described as follows:

EQUATION 3

*<a measure of covariance among the  $\{x_i\}$ >/<signal variance>*

Or, for a given autocorrelation function  $f(x, y)$  autocorrelation coefficient A can be described as

EQUATION 4

$$\frac{A(dx, dy)}{A(0, 0)} = \frac{\sum_{dx=0}^{xDim-1-dx} \sum_{dy=0}^{yDim-1-dy} f(x + dx, y + dy)}{(xDim - dx) * (yDim - dy)}$$

## IMAGE TYPES

Color and black and white 3D and 4D images.

## REFERENCES

Digital Image Processing, Second Edition by Rafael C. Gonzalez and Richard C. Woods, Prentice-Hall, Inc., 2002, pp. 205 - 208 and pp. 414-417.

## Applying the AutoCorrelation Coefficients algorithm

To use this algorithm, do the following:

- 1 Open an image of interest.
- 2 Select Algorithms > AutoCorrelation Coefficients in the MIPAV window. The AutoCorrelation Coefficients dialog box (Figure 3) appears.
- 3 Click OK. The algorithm begins to run, and a status bar appears with the status. When the algorithm finishes running, the progress bar disappears, and the result image replaces the original one.

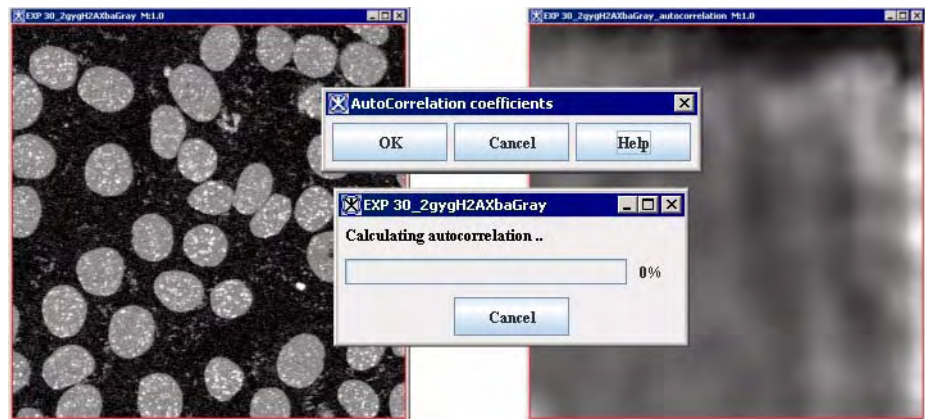


Figure 2. Calculating autocorrelation for 2D grayscale image

<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.	
<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes this dialog box.	
<b>Help</b>	Displays online help for this dialog box.	

Figure 3. AutoCorrelation Coefficients dialog box

## Autocovariance Coefficients

Autocovariance is a measure of the degree to which the outcome of the function  $f(T + t)$  at coordinates  $(T + t)$  depends upon the outcome of  $f(T)$  at coordinates  $t$ . It provides a description of the texture or a nature of the noise structure.

### Background

In statistics, given a real stochastic process  $f(t)$ , the auto-covariance is simply the covariance of the signal against a time-shifted version of itself. If each state of the series has a mean:

EQUATION 1

$$E[f(T)] = \mu_T$$

then the auto-covariance is given by

EQUATION 2

$$G_{xx}(T, S) = E[(f_T - \mu_T) \cdot (f_S - \mu_S)] = (E[f_T \cdot f_S] - (\mu_T \cdot \mu_S))$$

If  $f(t)$  is wide sense stationary function, then

EQUATION 3

$$\mu_T = \mu_S = \mu$$

for all  $T$  and  $S$ , therefore

EQUATION 4

$$G_{xx}(T, S) = G_{xx}(S - T) = G_{xx}(t)$$

Where  $t = S - T$  is the amount of time by which the signal has been shifted.

When normalized by the variance  $\sigma^2$  the auto-covariance function becomes the auto-covariance coefficient

EQUATION 5

$$\rho = G_{xx}(t) / \sigma^2$$



Calculation of the auto-covariance function  $G(t)$  of a randomly varying function  $i(T)$  may be done in the time domain using the following equation:

EQUATION 6

$$G(t) = \langle i(T) \cdot i(T+t) \rangle$$

Correspondingly, the auto-covariance function may be calculated in the spatial domain, as

EQUATION 7

$$g(\xi) = \partial_i(x) \cdot \partial_i(x + \xi)$$

Where, the  $\langle \rangle$  brackets indicate integration over time. And

EQUATION 8

$$\partial_i(T) = (i(T) - \langle i(T) \rangle) / \langle i(T) \rangle$$

which gives us the normalized auto-covariance function or auto-covariance coefficient

EQUATION 9

$$g(t) = \langle \sigma_i(T) \cdot \sigma_i(T+t) \rangle = [i(T) \cdot i(T+t) - \langle i(T) \rangle^2] / \langle i(T) \rangle^2 = [G(t) / \langle i(t) \rangle^2] - 1$$

When the data consist of a set of  $N$  discrete points, the averaging is performed as sums, then in spatial domain the 1D auto-covariance function is calculated by

EQUATION 10

$$g(\xi) = \frac{(1/N) \sum_{k=1}^N i(k)i(k + \xi)}{[(1/N) \sum_{k=1}^N i(k)]^2} - 1$$

If the random intensity variable,  $i$ , is a function of two independent variables,  $x$  and  $y$ , then, it is possible to define a corresponding two-dimensional auto-covariance function:

EQUATION 11

$$g(\xi, \eta) = \langle \delta i(x, y) \delta i(x + \xi, y + \eta) \rangle$$

For a discrete set of data this becomes

EQUATION 12

$$g(\xi, \eta) = \frac{(1/NM) \sum_{k=1}^N \sum_{l=1}^M i(k, l) i(k + \xi, l + \eta)}{[(1/NM) \sum_{k=1}^N \sum_{l=1}^M i(k, l)]^2}$$

---

Note that while some medical literature refers to this as an autocorrelation, it is actually an auto-covariance since means are subtracted.

---



---

## NOTES

An auto-covariance function, which falls rapidly as a function of  $t$ , indicates that the resultant noise is in actual fact independent except at short separated distances, providing an appearance of 'sharp' noise.

An auto-covariance function, which falls off slowly and smoothly as a function of  $t$ , indicates that the noise is highly correlated. At a given point, a positive value of the auto-covariance means that the noise tends to be of the same sign, while a negative value indicates that the noise tends to be of the opposite sign.

In short, the amplitude of the auto-covariance at zero displacement provides a measure of the noise magnitude, while the shape the auto-covariance can be used to describe the nature of noise.

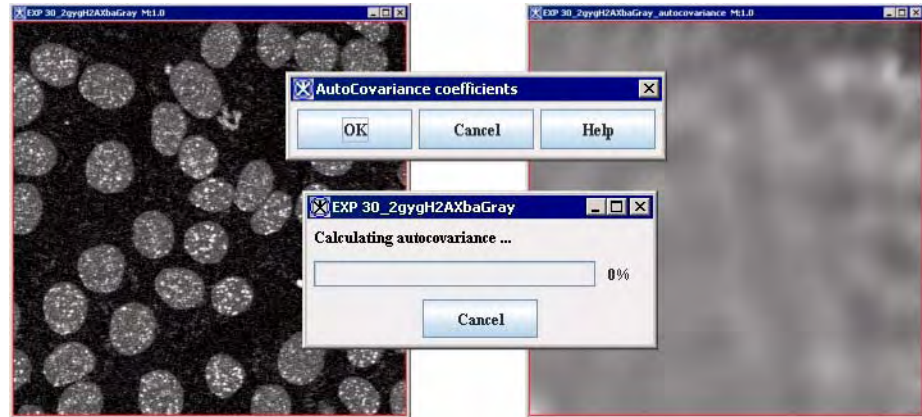


Figure 1. Calculating auto-covariance for a 2D grayscale image

---

## IMAGE TYPES

Color and black and white 3D and 4D images.

---

## REFERENCES

Digital Image Processing, Second Edition by Rafael C. Gonzalez and Richard C. Woods, Prentice-Hall, Inc., 2002, pp. 205 - 208 and pp. 414-417.

"Two-photon image correlation spectroscopy and image cross-correlation spectroscopy" by P. W. Wiseman, J.A. Squier, M.H. Ellisman, and K.R. Wilson, Journal of Microscopy, Vol. 200. Pt. 1, October 2000, pp. 14-25.

"Quantitation of Membrane Receptor Distributions by Image Correlation Spectroscopy: Concept and Application" by Nils O. Petersen, Pia L. Hoddelius, Paul W. Wiseman, Olle Seger, and Karl-Eric Magnusson, Biophysical Journal, Volume 65, September, 1993, pp. 1135-1146.

"Image cross-correlation spectroscopy: A new experimental biophysical approach to measurement of slow diffusion of fluorescent molecules" by Mamta Srivastava & Nils O. Petersen, Methods in Cell Science, Vol. 18, March, 1996, pp. 47-54.

"Analysis of recorded image noise in nuclear medicine" by Benjamin M. W. Tsui, Robert N. Beck, Kunio Doi and Charles E. Metz, Phys. Med. Biol., 1981, Vol. 26. No. 5, 883-902. Printed in Great Britain.

## Applying the AutoCovariance Coefficients algorithm

To use this algorithm, do the following:

- 1** Select Algorithms > AutoCovariance Coefficients in the MIPAV window. The AutoCovariance Coefficients dialog box (Figure 2) appears.
- 2** Click OK. The algorithm begins to run, and a status bar appears with the status. When the algorithm finishes running, the progress bar disappears, and the results replace the original image.


<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.	
<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes this dialog box.	
<b>Help</b>	Displays online help for this dialog box.	

Figure 2. AutoCovariance Coefficients dialog box

## Barrel Distortion Correction

Barrel distortion occurs when the magnification at the center of the lens is greater than at the edges. Fisheye lenses, which take hemispherical views, produce this type of distortion as a result of a hemispherical scene being projected onto a flat surface. A higher quality lens can be used to correct for distortion; however, this will greatly rise the cost of the image capture system. A cost effective alternative to an expensive lens system is to algorithmically correct for barrel distortion using the presented method.

Barrel distortion is primarily radial in nature, therefore it can be corrected using a relatively simple model compensating for the most of the distortion.

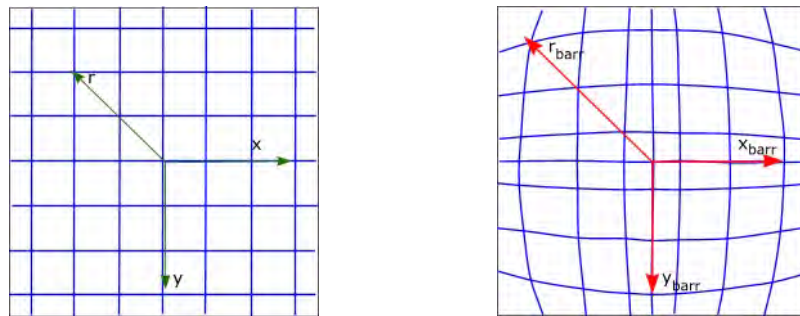


Figure 1. Illustration of barrel distortion

## Background

The correcting function  $M(a,b,c,d,r_{corr})$  is a third order polynomial. It relates the distance of a pixel from the center of the source image ( $r_{src}$ ) to the corresponding distance in the corrected image ( $r_{corr}$ ):

EQUATION 1

$$M = a * r_{3corr} + b * r_{2corr} + c * r_{corr} + d$$

and

$$r_{src} = (a * r_{3corr} + b * r_{2corr} + c * r_{corr} + d) * r_{corr}$$

Where  $r_{src}$  and  $r_{corr}$  are specified in units of the  $min((xDim-1)/2, (yDim-1)/2)$ .

Parameters in Eq. 1 are as follows:

- $a$ ,  $b$  and  $c$  describe distortion of the image
- $d$  describes the linear scaling of the image

Correcting using  $a$  affects only the outermost pixels of the image, while  $b$  correction is more uniform. Using negative values for  $a$ ,  $b$ , and  $c$  shifts distant points away from the center. This counteracts barrel distortion, and forms the basis for the above corrections.

Using positive values for  $a$ ,  $b$ , and  $c$  shifts distant points towards the center. This counteracts *pincushion distortion* which is opposite to barrel distortion. See Figure 2.

Using  $d=1$ , and  $a=b=c=0$  leaves the image as it is. Choosing other  $d$  values scales the image by that amount. See Figure 2.

### Correcting pincushion and/or barrel distortion

Finally, you may correct pincushion and barrel distortions simultaneously in the same image: if the outer regions exhibit barrel distortion, and the inner parts pincushion, you should use negative  $a$  and positive  $b$  values. If you do not want to scale the image, you should set  $d$  so that  $a + b + c + d = 1$ .

### Examples

In most cases, you will get quite satisfactory results by using just one parameter, such as the parameter  $b$  from the examples shown in Figure 2.

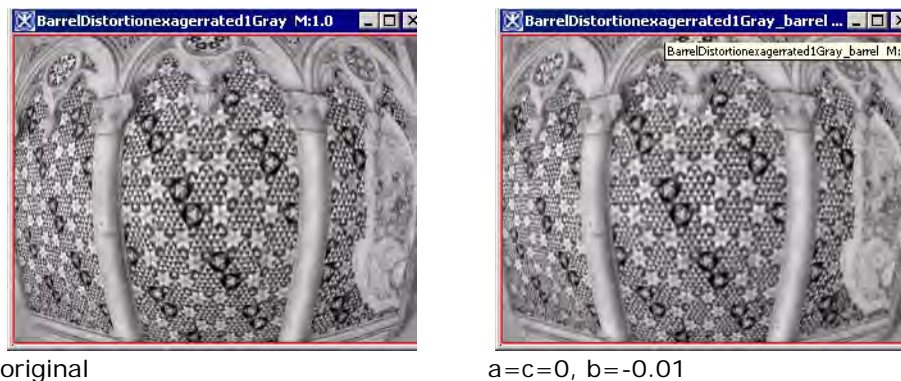
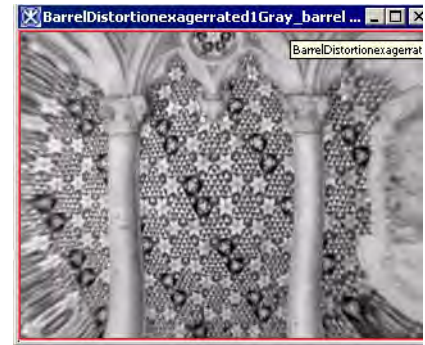


Figure 2. Barrel distortion correction



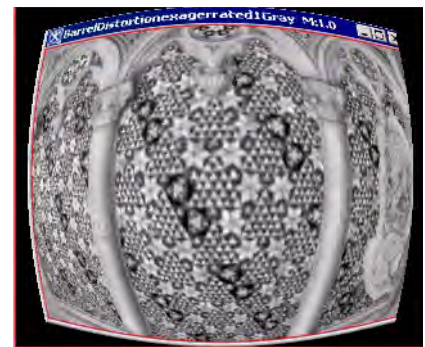
$a=c=0, b=-0.1$



$a=c=0, b=-0.2$



$a=b=c=0, d=2$



$a=c=0.05, b=0.2$

Barrel distortion correction – the parameters for each image are shown below the image. As  $b$  becomes more negative (e.g. changes from  $-0.1$  to  $-0.2$ ), the amount of barrel distortion correction increases, while positive coefficients make barrel distortion worse. If  $d$  parameter is not shown, that means that  $d$  is set so that  $a + b + c + d = 1$ .

**Figure 2. Barrel distortion correction (continued)**

Examples from Figure 2 may also serve as a guide of how large parameter values should be, i.e. select the parameters around 0.1 (in absolute value) if barrel distortion is quite visible, or close to 0.01 if it is very small. Simply optimize the initial parameters until you like your image.

## REFERENCES

Helmut Dersch, "Correcting Barrel Distortion" available at <http://www.all-in-one.ee/~dersch/barrel/barrel.html>

PanoTools wiki: *Lens correction model* available at [http://wiki.panotools.org/Lens\\_correction\\_model](http://wiki.panotools.org/Lens_correction_model)

## IMAGE TYPES

The algorithm can be applied to RGB and gray 2D images.

### Applying barrel distortion correction algorithm

- 1 Open an image of interest.
- 2 Select Algorithms > Transformation tools > Barrel Distortion Correction. The **Barrel/pincushion distortion correction** dialog box opens.
- 3 Complete the dialog box and click OK.

The corrected image appears in a new image frame. See Figure 2.

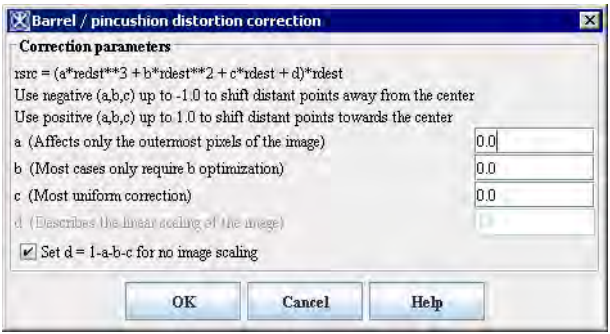
a	Correcting using $a$ affects only the outermost pixels of the image.	
b		
c		
d	$b$ and $c$ parameters should be used for more uniform correction.	
	<p><b>Using negative values</b> for <math>a</math>, <math>b</math>, and <math>c</math> shifts distant points away from the center of the image. This negates barrel distortions.</p> <p><b>Using positive values</b> for <math>a</math>, <math>b</math>, and <math>c</math> shifts distant points towards the center. This negates pincushion distortions.</p> <p>Using <math>d=1</math>, and <math>a=b=c=0</math> leaves the image as it is. Choosing other <math>d</math> values scales the image by that amount.</p>	
<b>Set d=1-a-b-c</b>	Activate this option if you do not want to scale the image, e.g. you set $d$ so that $a + b + c + d = 1$ .	
<b>OK</b>	Applies the parameters that you specified to the image.	
<b>Cancel</b>	Disregards any changes you made in this dialog box and closes the dialog box.	
<b>Help</b>	Displays online help for this dialog box.	

Figure 3. Barrel/pincushion distortion correction dialog box





## B-Spline Automatic Registration

This section describes the B-spline based registration of images in three dimensions (3D). The description specializes to two dimensions (2D) in the obvious manner.

### Background

Let the target image have dimensions  $p_0 \times p_0 \times p_0$  with image indices satisfying  $0 \leq j_0 < p_0$ ,  $0 \leq j_1 < p_1$ , and  $0 \leq j_2 \leq p_2$ . Let the source image have dimensions  $q_0 \times q_1 \times q_2$  with image indices satisfying  $0 \leq k_0 < q_0$ ,  $0 \leq k_1 < q_1$ , and  $0 \leq k_2 \leq q_2$ . The B-spline volume function maps indices  $j_0, j_1, j_2$  via Equation 1.

EQUATION 1

$$M_{j_0 j_1 j_2} = \sum_{i_0=0}^{n_0} \sum_{i_1=0}^{n_1} \sum_{i_2=0}^{n_2} N_{i_0, d_0} \left( \frac{j_0}{p_0-1} \right) N_{i_1, d_1} \left( \frac{j_1}{p_1-1} \right) N_{i_2, d_2} \left( \frac{j_2}{p_2-1} \right) B_{i_0, i_1, i_2}$$

where the  $N_{i_e, d_e}(u)$  functions are the B-spline basis functions, and the  $\mathbf{B}$  values are the control points. The numbers  $n_0, n_1$ , and  $n_2$  are user selectable, the degrees  $d_0, d_1$ , and  $d_2$  for the basis functions are user selectable, and the B-spline basis functions are open and uniform.

The control points have the following characteristics:

- Each control point has coordinates from the unit cube. In two dimensions, each control point has coordinates from the unit square.
- The initial placement of the control points depends on the B-spline basis and must yield the identity function; that is,

$$M_{j_0 j_1 j_2} = \left( \frac{j_0}{p_0-1}, \frac{j_1}{p_1-1}, \frac{j_2}{p_2+1} \right). \text{ The initial placement of the control}$$

points is described in Section “Improving performance ” on page 105.

- A control point is said to be in the outer layer if one of its indexes is either the minimum index (zero) or the maximum index (one less than the number of control points in a given dimension). The position of the control points in the outer layer are fixed as follows:

EQUATION 2

$$\begin{aligned}
 B_{0, i_1, i_2} &= (0, y_{i_1}, z_{i_2}) \\
 B_{i_0, 0, i_2} &= (x_{i_0}, 0, z_{i_2}) \\
 B_{i_0, i_1, 0} &= (x_{i_0}, y_{i_1}, 0) \\
 B_{n_0 - 1, i_1, i_2} &= (1, y_{i_1}, z_{i_2}) \\
 B_{i_0, n_1 - 1, i_2} &= (x_{i_0}, 1, z_{i_2}) \\
 B_{i_0, i_1, n_2 - 1} &= (x_{i_0}, y_{i_1}, 1)
 \end{aligned}$$

This constraint implies that the  $0 \leq k'_w < q_w$  for all  $w$ .

- A control point not in the outer layer is said to be an interior control point. The position of the interior control points is constrained such that its movement must not cause any folding in the B-spline transformation. *Folding* is defined to occur when there is a negative value anywhere in the computed deformation for the current B-spline transformation. The computation of the deformation is described in Section “Deformation” on page 109. The method by which folding is detected is described in Section “Detect folding” on page 110.

Each  $M_{j_0 j_1 j_2}$  contains normalized 3D coordinates (i.e., in the unit cube) that identify a location in the source image corresponding to the sample at  $(j_0, j_1, j_2)$  in the target image. The  $M_{j_0 j_1 j_2}$  coordinates are related to sample

coordinates in the source image by  $M_{j_0 j_1 j_2} = \left( \frac{k'_0}{q_0 - 1}, \frac{k'_1}{q_1 - 1}, \frac{k'_2}{q_2 - 1} \right)$  where

the output indices  $(k'_0, k'_1, k'_2)$  are considered to be continuous values; that is, they are not necessarily integer valued. If the source image's samples are  $S_{k_0, k_1, k_2}$ , where  $0 \leq k'_w < q_w$  for all  $w$ , we need to evaluate the image at the nonintegral values  $(k'_0, k'_1, k'_2)$ . We can use trilinear interpolation to do this (use bilinear interpolation in the case of two-dimensional (2D))

registration). Let  $k_w = \lfloor k'_w \rfloor$ , the largest integer smaller than  $k'_w$ . Define  $\Delta_w = k'_w - k_w$ . The interpolated image value is:

EQUATION 3

$$\begin{aligned}
 U_0 &= (1 - \Delta_0) \cdot S_{k_0, k_1, k_2} + \Delta_0 \cdot S_{k_0 + 1, k_1, k_2} \\
 U_1 &= (1 - \Delta_0) \cdot S_{k_0, k_1 + 1, k_2} + \Delta_0 \cdot S_{k_0 + 1, k_1 + 1, k_2} \\
 U_2 &= (1 - \Delta_0) \cdot S_{k_0, k_1, k_2 + 1} + \Delta_0 \cdot S_{k_0 + 1, k_1, k_2 + 1} \\
 U_3 &= (1 - \Delta_0) \cdot S_{k_0, k_1 + 1, k_2 + 1} + \Delta_0 \cdot S_{k_0 + 1, k_1 + 1, k_2 + 1} \\
 V_0 &= (1 - \Delta_1) \cdot U_0 + \Delta_1 \cdot U_1 \\
 V_1 &= (1 - \Delta_1) \cdot U_2 + \Delta_1 \cdot U_3 \\
 S(k'_0, k'_1, k'_2) &= (1 - \Delta_2) \cdot V_0 + \Delta_2 \cdot V_1
 \end{aligned}$$

If the target image samples are  $T_{j_0, j_1, j_2}$ , where  $0 \leq j_w < p_w$  for all  $w$ , then the source image  $S$  is trilinearly interpolated at  $(k'_0, k'_1, k'_2)$ , which are the coordinates of  $M_{j_0, j_1, j_2}$ . This means that the currently registered source image, denoted  $S'$ , has the same dimensions as the target image and can be defined in terms of the target image samples as  $S'(j_0, j_1, j_2) = S(k'_0, k'_1, k'_2)$ . A measure of error between the two images  $S_{j_0, j_1, j_2}$  and  $T_{j_0, j_1, j_2}$  can be computed. A number of possible error measure functions are defined in Section 1.2. The goal is to select the control point positions to make the error measure as small as possible. This is done by moving a single control point at a time using gradient descent.

The gradient of the error function is estimated for the control point  $B_{i_0, i_1, i_2}$  by means of finite differences, where changes in the error function are measured for small steps of the control point from its current position in each of the coordinate axis directions. Let  $\Delta_x$ ,  $\Delta_y$  and  $\Delta_z$  be small step

sizes equivalent to one sample increment in the target image,

EQUATION 4

$$\left(\Delta x = \frac{1}{p_0}\right), \left(\Delta y = \frac{1}{p_1}\right), \left(\Delta z = \frac{1}{p_2}\right)$$

The gradient of the error function is estimated at each control point  $B_{i_0, i_1, i_2}$ . The gradient at each control point is denoted by

EQUATION 5

$$\nabla E(i_0, i_1, i_2) = \left( \frac{E(i_0 + \Delta x, i_1, i_2) - E(i_0 - \Delta x, i_1, i_2)}{2 \cdot \Delta x}, \frac{E(i_0, i_1 + \Delta y, i_2) - E(i_0, i_1 - \Delta y, i_2)}{2 \cdot \Delta y}, \frac{E(i_0, i_1, i_2 + \Delta z) - E(i_0, i_1, i_2 - \Delta z)}{2 \cdot \Delta z} \right)$$

The gradient descent step is as follows. The current control point under consideration is  $B_{i_0, i_1, i_2}$ . The ray originating at this point and having direction of greatest decrease in E is

EQUATION 6

$$B'_{i_0, i_1, i_2}(t) = B_{i_0, i_1, i_2} + tD$$

where the direction is the unit length vector

$$D = \frac{\nabla E(i_0, i_1, i_2)}{|\nabla E(i_0, i_1, i_2)|}$$

and where  $\nabla E$  is computed using Equation 5. The idea is to choose  $t > 0$  to minimize

$$e(t) = E(B'_{i_0, i_1, i_2}(t))$$

A calculus approach which sets the first derivative to zero and solves for  $t$  yields an expression for  $e(t)$  that is really complicated. Instead, you should choose a set of  $t$ -values,  $t^l$  through  $t_m$ , evaluate  $e(t_i)$  and find the minimum value. Suppose this occurs at  $t_k$ . Replace this in Equation 6 to obtain the new location for the control point.

Only those  $t$ -values which do not result in the occurrence of folding in the B-spline transformation are considered. The method described in Section “Detect folding” on page 110 is used to determine if folding occurs for a given set of control point positions. Note that sample  $t_0$  (i.e.,  $t=0$ ) is equivalent to the location of the control point before any movement attempt, and by definition, sample  $t_0$  does not cause folding. A test is made to determine whether folding occurs for  $t_m$ , the extreme sample in the set of chosen  $t$ -values. If folding does not occur at  $t_m$ , then folding does not occur at any of the other  $t$ -values between  $t_0$  and  $t_m$ . If folding does occur at  $t_m$ , then a bisection method is employed to find the  $t_{old}$  value between  $t_0$  and  $t_m$  where folding begins to occur. The search for a minimum value of  $e(t)$  by sampling must be limited to the  $t < t_{old}$ .

## Error measures

All of the error measures between two images defined in this section assume that the two images have the same dimensions. This is the case for the target image  $T(j_0, j_1, j_2)$  and registered source image  $S'(j_0, j_1, j_2) = S(k'_0, k'_1, k'_2)$ . The particular error measures defined here are only concerned with corresponding samples from the two images or histograms of the values of the samples from the two images, and are not concerned with whether the images have two or three dimensions. For simplicity, each image measure will be defined with a single index to access its samples.

---

## LEAST SQUARES MEASURE

The least squares measure is defined by

EQUATION 7

$$E_{LS} = \sum_j |S'(j) - T(j)|^2$$

where the summation is over all the indices for the target image voxels.

Here,  $S'(j)$  is a registered source image and  $T(j)$  is a target image. The minimum value of this measure is zero; the maximum value depends on the range of values in the target and registered source images.

---

## CORRELATION RATIO MEASURE

The correlation ratio measure is defined by

EQUATION 8

$$E_{CR} = \frac{\sum \frac{n_k}{N} \text{Var}(S'_k)}{\text{Var}(S)}$$

where

- $\text{Var}(S)$  is the variance of the registered source image  $S'$ .
- $k$  is the number of intensity histogram bins equally distributed over the range of intensity values of the target image  $T$ .
- $S'_k$  is the  $k$ -th isoset defined as the set of intensities in registered source image  $S'$  at positions where the intensity in target image  $T$  is in the  $k$ -th intensity bin.
- $n_k$  is the number of elements in the set  $S'_k$ , such that  $N = \sum_k n_k$ .
- $N$  is the number of samples in the target image  $T$  which is the same number of samples in the registered source image  $S'$ .
- $\text{Var}(S'_k)$  is the variance of the values in the set  $S'_k$ .

The range of values for this measure is  $[0,1]$ .

## NORMALIZED MUTUAL INFORMATION MEASURE

The normalized mutual information measure is defined by

EQUATION 9

$$E_{NMI} = \frac{H(S, T)}{H(S) + H(T)}$$

where

$$H(S, T) = -\sum_{i,j} p_{i,j} \log p_{i,j}$$

is the standard entropy definition for probability  $p_{ij}$  estimated using the  $(i,j)$  joint histogram bins, and similarly for the marginals  $H(S)$  and  $H(T)$ .

The standard entropy calculation

$$H(S, T) = -\sum_k p_k \log p_k$$

for an image  $X$  of  $N$  samples and user-selectable  $k$  histogram intensity bins equally distributed over the range of samples in image  $X$  can be rewritten as

$$H(X) = -\sum_k \frac{n_k}{N} \log \left( \frac{n_k}{N} \right)$$

where  $n_k$  is the histogram value for the  $k$ -th intensity bin and  $N = \sum_k n_k$ . The computational effort can be reduced by factoring to get



$$H(X) = \left(-\frac{1}{N}\right) \sum_k n_k \cdot (\log n_k - \log N)$$

where  $\log n_k$  can be a precomputed lookup table of logarithms of integer values.

The range of values for this measure is [0,1].

## Improving performance

The gradient descent applied to a control point at a time can be quite slow. Performance can be improved by taking advantage of the local control aspect of using B-splines. Since only one control point is being moved at a time, the local control aspect of using B-splines means that only some of the registered source image values will be different. Note that for an image of a given size, the number of image values, under local control, affected by moving a control point decreases as the number of control points increases.

One performance improvement can be achieved by precomputing the B-spline basis function values once for each possible sample for each dimension. Since the registration occurs by transforming the source image into the space of the target image, all samples for B-spline basis function evaluation always occur at the discrete sample coordinates for the target image. Since the number of target images samples for a dimension is fixed, the B-spline basis function will be evaluated on the [0, 1] interval at equally spaced intervals (we are using open, uniform B-splines). For each dimension, a 2D array is created which stores B-splines basis function evaluations for each sample with the weight associated with each control point. Because of local control, many of these weights will be zero indicating that a sample is not affected by a particular control point. In the process of creating this 2D array of precomputed basis function evaluations, the index range of samples affected by each control point can be determined. The algorithm to determine this range does not examine the weights as they are computed; instead, the range of control points affecting a single sample are determined from the computed knot index and the known B-spline degree.

Another performance improvement involves maintaining intermediate error measure calculations. For example, in the case of the least squares error measure, an image is maintained of the current error calculations, that is, the target image subtracted from the current registered source

image. A cumulative total error value is also maintained, which is the sum of the squared error values for all current values in the error image. When a control point is moved, the old values that are being replaced are first subtracted from the total error, and then the new values to be stored are added to the total error. Similar approaches are performed for the correlation ratio and normalized mutual information error measures such as storing cumulative histogram values and images which contain predetermined histogram bin indexes.

## Initial placement of control points for identity map

Referring to Equation 1 for the definition of the B-spline mapping function, the identity map is the one which yields

$$M_{j_0 j_1 j_2} = \left( \frac{j_0}{p_0 - 1}, \frac{j_1}{p_1 - 1}, \frac{j_2}{p_2 + 1} \right).$$

This section describes how to initially place the control points to yield this identity map. Since a separate set of B-spline basis functions is defined for each dimension, it happens that the placement of the control points to provide this identity map is computed independently for each dimension. This means that the  $x$ -coordinate for each control point initially is determined by the B-spline basis defined for the  $X$ -axis, and similarly for other dimensions.

Let  $k$  be the coordinate for which the control point identity map is being computed for the associated B-spline basis. Let  $n_k$  be the number of control points defined for that B-spline basis. Let  $i_k$  be the index of the control point coordinate such that  $0 \leq i_k \leq n_k$ . The goal is to compute the coordinate values  $B_{i_k}$  for all  $i_k$  where each coordinate value is in the  $[0,1]$  range.

For a B-spline basis of degree 1, the initial placement of the control points for the identity map is such that the control point coordinates are uniformly spaced as follows:

EQUATION 10

$$B_{i_k} = \frac{i_k}{n_k}$$

for coordinate  $k$ . If the problem is to register images of three dimensions, and the B-spline bases for all three dimensions are selected to have degree 1, then

EQUATION 11

$$B_{i_0, i_1, i_2} = \left( \frac{i_0}{n_0}, \frac{i_1}{n_1}, \frac{i_2}{n_2} \right)$$

For a B-spline basis of degree 2 or larger, the initial placement of the control point coordinates is computed by the method of least squares. Again referring to Equation 1 for the definition of the B-spline mapping function, the identity map for coordinate  $k$  is

$$\frac{j_k}{p_{k-1}} = \sum_{i_k=0}^{n_k} N_{i_k, d_k} \left( \frac{j_k}{p_{k-1}} \right) B_{i_k}$$

for any target image coordinate  $j_k$  satisfying  $0 \leq j_k < p_k$ .

A system of  $p_k$  equations above for  $n_k + 1$  unknowns (i.e., the  $B_{i_k}$  coordinate values) is set up in a matrix form  $\mathbf{AX} = \mathbf{C}$ , where

$$\left( X_i = B_{i_k} \right), \left( A_{ji} = N_{i_k, d_k} \cdot \left( \frac{j_k}{p_{k-1}} \right) \right), \left( C_j = \frac{j_k}{p_{k-1}} \right)$$

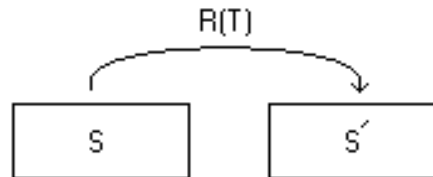
The solution is computed as

$$X = (A^T A)^{-1} A^T C$$

The repositioning of the control points to yield this identity map is guaranteed not have create folding in the resulting transformation.

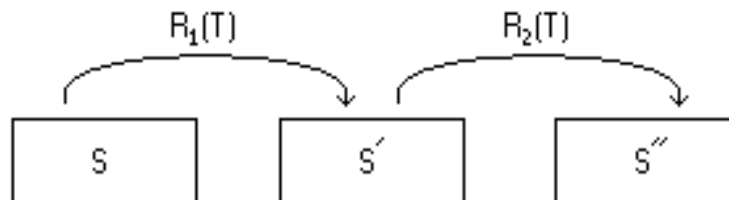
## COMPOSITION FOR TWO-PASS REGISTRATION

Let  $S$  be a source image and  $T$  be the target image. If the image  $S$  is registered to image  $T$ , then image  $S$  becomes the registered source image where  $R(T)$  is the registration mapping. This is graphically shown as:



For B-spline registration, the  $R(T)$  mapping is based on the selected B-spline basis and control points for each axis.

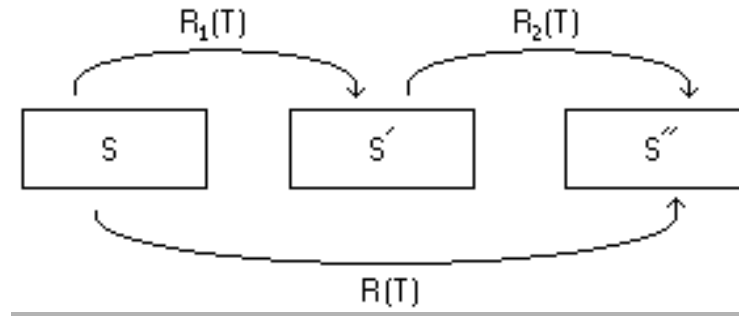
In the case of two-pass registration, the registered source image  $S'$  is then registered to the same target image  $T$ , but this time using different mapping parameters, i.e., a different set of B-spline basis and control points for the axes. The output of this second registration is a new registered source image  $S''$ . Let  $R_1(T)$  be the mapping for the first pass registration and  $R_2(T)$  be the mapping for the second pass registration. This is graphically shown as:



The usefulness of a two-pass registration is that the first pass would apply a coarse-level computation to quickly arrive at an approximate registration, and then the second pass would apply a finer-level computation to continue from the point where the first pass registration finished. The first pass registration may use B-spline bases with fewer control points and a smaller degree.

Although the two-pass registration can be performed in the manner just described, it is desirable to have a final registration between the original source image  $S$  and the target image  $T$ .

This means defining a mapping  $R(T)$  having the same B-spline bases as the second pass registration mapping  $R_2(T)$ , but which maps the original source image  $S$  into the final registered source image  $S$ . This is graphically shown as:



The  $R(T)$  mapping for the second pass in a two-pass registration is different in that the initial placement of the control points does not yield the identity map to the original source image  $S$ .

Instead, the initial placement of the control points for  $R(T)$  yields the output of the  $R_1(T)$  mapping. This is achieved by initializing the  $R_2(T)$  registration mapping in the normal manner which provides an initial placement of its control points to yield the identity map from the registered source image  $S$ . The initial placement of the control points for the  $R(T)$  mapping then are the initial placement of the control points for  $R_2(T)$  mapped through  $R_1(T)$ .

## Deformation

The deformation from the nonlinearly registered source image  $S$  to the target image  $T$  can be measured at each sample in the target image by computing the determinant of the Jacobian matrix for the nonlinear transformation. For three dimensions, the nonlinear registration transformation is defined by Equation 1. Since each  $M_{j_0j_1j_2}$  contains normalized 3D coordinates of the position in the source image mapped for each  $(j_0, j_1, j_2)$  in the target image, a map image of 3D coordinates can be created that contains these  $M_{j_0j_1j_2}$  source images coordinates (this map image has the same dimensions as the target image). An estimate of the gradient at each sample in this map image is computed using central finite differences at interior samples and forward/backward finite differences at “outer layer” samples. Since each sample in the map image contains 3D coordinates, the estimated

gradient at each sample then is a 3x3 matrix of values representing the Jacobian matrix. A single-valued deformation image with the same dimensions as the target image is created which contains the computed determinants of the Jacobian matrices for each sample.

If no deformation has occurred at a sample, the Jacobian matrix is the identity and its determinant is 1. If the source image is compressed near a sample, the determinant will be less than one. If the source image is expanded near a sample, the determinant will be greater than one. The computation of deformation is unaffected by differences in the dimensions for the source and target images. Because of the constraints on the placement of the control points, a negative determinant should not be possible, which would indicate some type of folding in the mapping.

## Detect folding

The situation of folding in a 2D or 3D B-spline transformation occurs when there is a negative value in the computed deformation. The computation of the deformation is described in Section “Deformation” on page 109. This section describes a method for detecting if such folding occurs given the current positions for the lattice of the B-spline control points. Given the lattice structure of the B-spline control points, folding can be detected by geometric means. In two dimensions, consider any interior control point and its 8 neighboring control points forming a polygon. Construct 8 triangles involving the chosen interior control point each time with the 8 pairings of adjacent control points on the polygon perimeter. The intersection of any one triangle with the other 7 triangles should each be empty in order for there to be no folding. Alternatively, the area of the polygon should be identical to the union of the areas of the 8 triangles.

The issues are similar in three dimensions by considering the 26 neighboring control points forming a polyhedron. In this case, 48 tetrahedrons are constructed involving the chosen interior control point each time along with the triples of control points from the 48 triangles forming the polyhedron mesh. The intersection of any one tetrahedron with the other 47 tetrahedrons should each be empty in order for there to be no folding. Alternatively, the volume of the polyhedron should be identical to the union of the volumes of the 48 tetrahedrons.

A sufficient but not necessary condition to conclude that folding occurs in the case of two dimensions is the premise that any interior control point is

outside the polygon formed by its 8 neighboring control points. In the case of three dimensions, the premise is that any interior control point is outside the polyhedron formed by its 26 neighboring control points.

A condition that is both necessary and sufficient to conclude that folding occurs does exist for both 2D and 3D. In the case of two dimensions, again consider the 8 triangles formed by the chosen interior control point each time with the 8 pairings of adjacent control points on the polygon perimeter. It is important that these triangles formed by control points from the lattice be consistently counterclockwise ordered. For each triangle, let  $A = (a_0, a_1)$  be the 2D coordinates of the chosen interior control point, and let  $B = (b_0, b_1)$  and  $C = (c_0, c_1)$  be the 2D coordinates of the other two triangle control points taken in counterclockwise order. The normal to the plane which contains this triangle can be computed from the (right handed) cross product of the two vectors  $\overline{AB}$  and  $\overline{AC}$ . One way to express the calculation of the cross product involves the construction of the following matrix which contains the three triangle point coordinates:

EQUATION 12

$$\begin{bmatrix} a_0 & a_1 & 1 \\ b_0 & b_1 & 1 \\ c_0 & c_1 & 1 \end{bmatrix}$$

The determinant of this matrix is twice the signed area of the triangle. Starting with consistently counter clockwise ordered triangles from the initially placed lattice of control points, the determinant of this matrix should remain positive as control point positions are moved. The determinant will become negative when the triangle *folds*.

The premise for the necessary and sufficient condition to detect when folding occurs extends to three dimensions. Consider the 48 tetrahedrons formed by the chosen interior control point each time with the triples of adjacent control points from the 48 triangles forming the polyhedron mesh. It is important that the tetrahedrons formed by the control points from the lattice be consistently counterclockwise ordered, in the 3D sense. For each tetrahedron, let  $A = (a_0, a_1, a_2)$  be the 3D coordinates of the chosen interior control point, and let  $B = (b_0, b_1, b_2)$ ,  $C = (c_0, c_1, c_2)$ , and  $D = (d_0, d_1, d_2)$  be the 3D coordinates of the other three tetrahedron control points taken in counterclockwise order. Extending the matrix determinant formulation

used for two dimensions to three dimensions, the following matrix is constructed which contains the four tetrahedron point coordinates:

$$\begin{bmatrix} a_0 & a_1 & a_2 & 1 \\ b_0 & b_1 & b_2 & 1 \\ c_0 & c_1 & c_2 & 1 \\ d_0 & d_1 & d_2 & 1 \end{bmatrix}$$

The determinant of this matrix is 6 times the signed volume of the tetrahedron. Starting with consistently counterclockwise ordered tetrahedrons from the initially placed lattice of control points, the determinant of this matrix should remain positive as control point positions are moved. The determinant will become negative when the tetrahedron *folds*.

### Support for color images

The registration of 2D and 3D color images involves first converting the three color channels of values at each sample in the source and target color images to create single channel source and target intensity images. Once the conversion has been done, the single channel intensity source and target images are registered as described in “Background” on page 98. The resulting transformation map is extracted from the registration which is then used to interpolate the original source color image in order to generate the output registered source color image.

Normally, the three color channels represent red, green, and blue intensities, and so the conversion to a single channel value usually represents overall intensity. In this manner, the conversion is simply a weighted average of the three color channels where separate weights can be provided for each channel. The current implementation uses equal weights for each channel.

### Support for 2.5D images

A 2.5D image is actually a 3D image containing a series 2D slice images. The registration of a 2.5D intensity or color image involves a 2D registration of each slice in the 2.5D image to a reference slice also in the same 2.5D image. The reference slice can either be a xed slice or the previous slice. The resulting output registered 2.5D image contains the same number of slices as the original 2.5D image where the sample values in a slice of the registered output are from the same slice of the original only “warped” by the B-spline registration transformation.



If the reference slice is fixed, then the reference slice will appear in the registered output as being unchanged. If the reference slice is the previous slice, then the first slice will be registered to the last slice.

---

## EXAMPLES OF B-SPLINE REGISTRATION

Figure 1 below shows an example of B-spline 3D image registration. Figure 2 shows the registration deformation image and its lookup table. The registration was done using the default parameter values, which are explained in Section “Image types” on page 114.




---

**Note:** that the input and the target image are both images of the same brain, that’s why the registration demonstrates a very good overall fitting.

---



Figure 1. The input source image (A), the input target image (B) and the output registered image (C). Note that images (A) and (B) are both the images of the same brain. That’s why it shows a very good overall fitting. Results may vary, if you take different brain images.

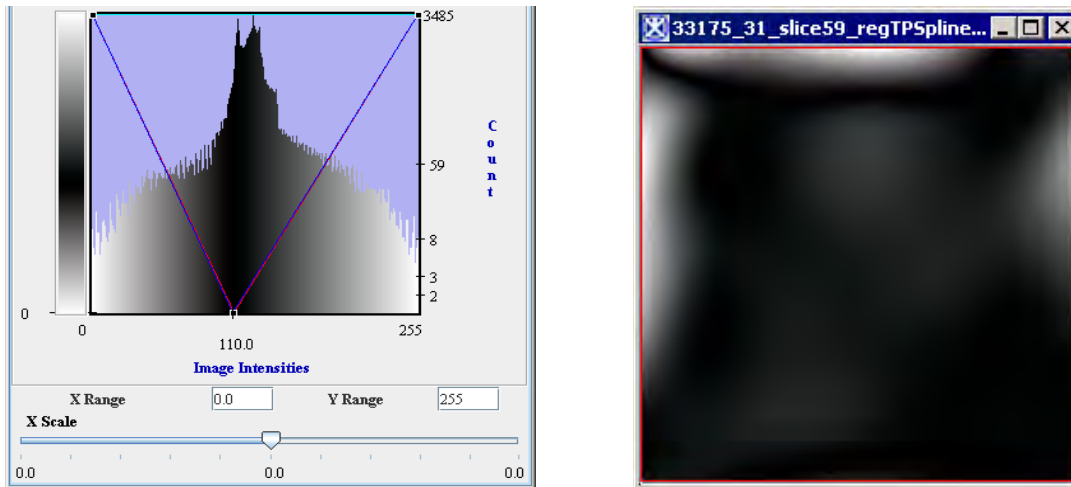


Figure 2. The deformation image and its lookup table. The dark areas on the deformation image are related to the areas on images (A) and (B) which did not perform a deformation during the registration or the deformation was relatively small. The light areas are related to the areas on images (A) and (B) which perform a bigger deformation.

## IMAGE TYPES

The algorithm can be applied to 2D and 3D color (RGB) and grayscale images.

## User Dialogs in MIPAV

Whenever B-spline registration is selected by the user, a dialog is displayed for the user to select the registration reference and the options related to how the registration is to be performed. A different dialog is displayed depending on whether the user selects 2D/3D registration between two images, or 2.5D registration of slices within a single image. The displayed dialogs differ only in how the registration reference is specified which depends on whether 2D/3D registration or 2.5D registration was selected; otherwise, options related to how the registration is to be performed are the same.

The choice of 2D/3D or 2.5D B-spline registration is made by selecting the appropriate menu item from the Algorithms: Registration submenu of MIPAV. Refer to Figure 3 for details.

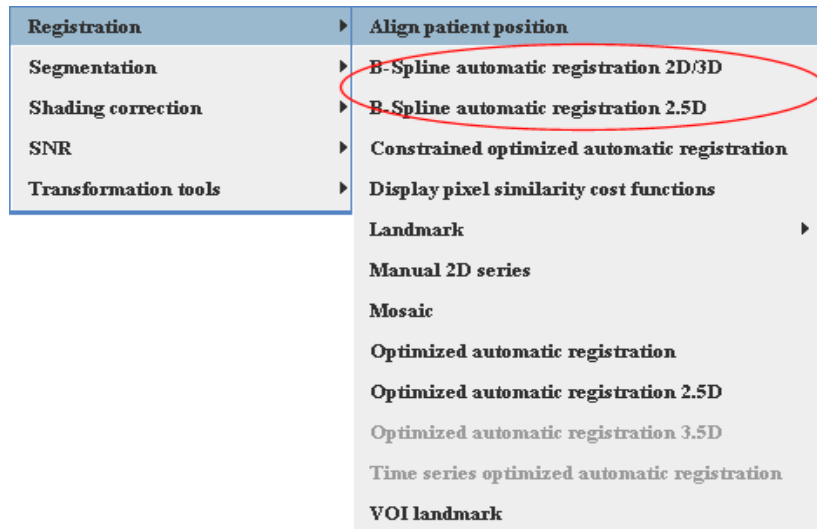


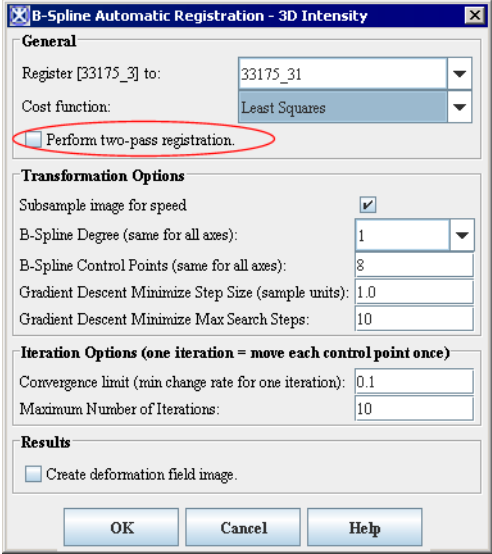
Figure 3. The choice of 2D/3D or 2.5D B-spline registration is made by selecting the appropriate menu item from the Algorithms: Registration submenu

The 2D/3D B-spline registration is available through the **B-spline automatic registration 2D/3D** menu item. This menu item is enabled when a 2D or 3D image (intensity or color) is the currently active image. The dialog is displayed only if there is at least one other loaded image in MIPAV which is compatible for registering to the currently active image. A compatible image is one that has the same number of channels of data and the same number of dimensions, but not necessarily the same sizes for each dimension. Refer to Figures 4 and 5.

The 2.5D B-spline registration is available through the **B-spline automatic registration 2.5D** menu item. This menu item is enabled only when a 3D image (intensity or color) is the currently active image. Refer to Figure 6.

Since the options related to how the registration is to be performed are identical for 2D/2.5D/3D and intensity/color images, and the only difference is how the registration reference is selected for 2D/3D vs. 2.5D, the dialog for presenting these options to the user is implemented in the same class. This class is `JDialogRegistrationBSpline` and is in the `mipav/model/view/dialogs` package. The class dynamically creates a dialog to present to the user with options that are appropriate for the type of registration.

Figure 4 below shows an example of the dialog which is displayed for 2D/3D intensity/color image registration. Note that this dialog is shown with the default options for **single-pass registration**.

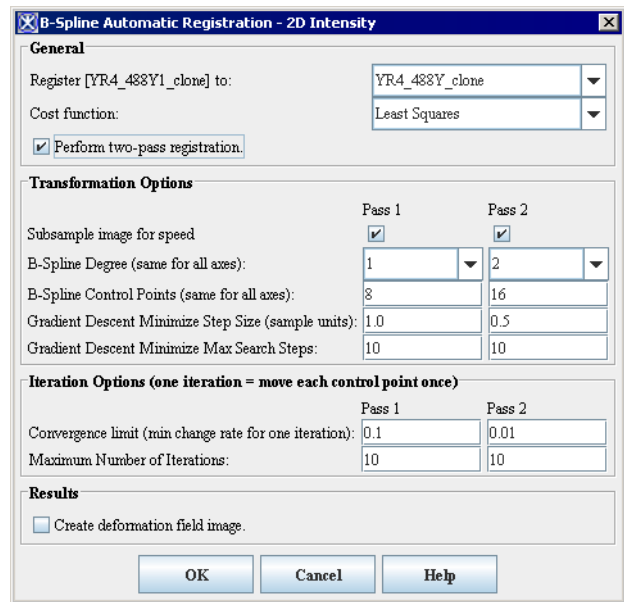
General		
<b>Register</b>	Displays a list of currently active 2D/3D images to registration. The drop down list of options show the names of available images currently loaded to which this image can be registered. This list will only contain the names of images currently loaded in MIPAV which are compatible.	
<b>Cost function</b>	This drop down list contains the following choices for cost measure to use during registration: Least Squares, Correlation Ratio, and Normalized Mutual Information.	
<b>Perform two-pass registration</b>	This dialog is shown with the default options for single-pass registration. If the <b>Perform two-pass registration</b> option is enabled, then the dialog will change to make options for <b>two-pass registration</b> available.	
Transformation Options		
<p>The following is a discussion of the dialog options which control each registration pass, one set of options per pass. For more information refer to "Transformation options" on page 124.</p>		
<b>Sub-sample image for speed</b>	If this option is enabled, the image will be subsample in power of 2 for each dimension. Subsampling increases performance of the algorithm, but it exhibits lower accuracy of registration.	
<b>B-Spline Degree (same for all axes)</b>	This is the degree of the B-Spline to use for all axes of the image. The drop list of options show the values of 1 (linear), 2 (quadratic), 3 (cubic), and 4 which are available. Although, the underlying BSpineBasisf class supports any degree, this dialog restricts the degrees to these values. Choosing a smaller degree executes faster than a larger degree, but a higher degree offers a higher order of continuity.	
<p><b>Figure 4. B-Spline Single - Pass Automatic Registration 2D/3D dialog box</b></p>		

<b>B-Spline Control Points (same for all axes)</b>	This is the number of B-spline control points to use for all axes of the image. The minimum number of control points is the larger of the number 2 and the degree of the B-Spline. This dialog limits the maximum number of control points to one-half of the number of samples in the smallest dimension.
<b>Gradient Descent Minimize Step Size (sample units)</b>	This is the size of steps each control point is moved along the gradient-based direction when searching for a minimum of the error. The registration does not take into account the size of an image sample in each dimension, so <b>this step size is specified in units of a sample</b> . E.g., a value of 0.5 will move the control point in increments of <b>half of a sample</b> while searching for the minimum error. The minimum step size is 0.1 sample and the maximum is 5 samples.
<b>Gradient Descent Minimize Max Search Steps</b>	This is the maximum number of steps the control point is moved along the gradient-based direction when searching for a minimum of the error. If the distance from the control point current position to the bounding polygon/polyhedron of its neighboring control points along the gradient-based direction is less than this specified value, then the computed distance value will be used for this control point instead. The minimum number of steps is 1 and the maximum is 100.
<b>Iteration options (one iteration=move each control point once)</b>	
<b>Convergence limit (min change rate for one iteration)</b>	This is a rate of change threshold such that if the rate the error changes between starting and ending of each iteration is smaller than this value, then the registration terminates.
<b>Maximum Number of Iterations</b>	If the convergence limit is not satisfied after this many iterations have been executed, then the registration terminates. The range of possible values is 1 to 100.
<b>Results</b>	
<b>Create deformation field image</b>	If this checkbox is enabled, then the deformation image is computed after the last pass of registration is performed. The image that is created has the same dimensions as the target and the registered source images, and has the name of the source image appended with <i>deformation</i> suffix. The deformation image will be automatically displayed along with the registered source image upon completion of registration. Refer to Figure 2 for details.
<b>OK</b>	Applies the algorithm to the input image that you chose in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

Figure 4. B-Spline Single - Pass Automatic Registration 2D/3D dialog box (continued)

If the **Perform two-pass registration** check box is selected, then the dialog will change to the following:

General	
<b>Register</b>	Displays a list of currently active 2D/3D images to registration. The drop down list of options show the names of available images currently loaded to which this image can be registered. This list will only contain the names of images currently loaded in MIPAV which are compatible.
<b>Cost function</b>	This drop down list contains the following choices for cost measure to use during registration: Least Squares, Correlation Ratio, and Normalized Mutual Information.
<b>Perform two-pass registration</b>	This dialog is shown with the default options for <b>two-pass registration</b> . If the <b>Perform two-pass registration</b> option is disabled, then the dialog will change to make options for <b>single-pass registration</b> available as shown in Figure 4.
<b>Transformation Options</b>	
<p>The following is a discussion of the dialog options which control each registration pass, one set of options per pass. For more information refer to "Transformation options" on page 124.</p>	
<b>Sub-sample image for speed</b>	If this option is enabled, the image will be subsample in power of 2 for each dimension. Subsampling increases performance of the algorithm, but it results lower accuracy of registration.
<b>B-Spline Degree (same for all axes)</b>	This is the degree of the B-Spline to use for all axes of the image. The drop list of options show the values of 1 (linear), 2 (quadratic), 3 (cubic), and 4 which are available. Although, the underlying BSpineBasisf class supports any degree, this dialog restricts the degrees to these values. Choosing a smaller degree executes faster than a larger degree, but a higher degree offers a higher order of continuity.
<b>B-Spline Control Points (same for all axes)</b>	This is the number of B-spline control points to use for all axes of the image. The minimum number of control points is the larger of the number 2 and the degree of the B-Spline. This dialog limits the maximum number of control points to one-half of the number of samples in the smallest dimension.
<p><b>Figure 5. B-Spline Two-Pass Automatic Registration 2D/3D dialog box</b></p>	



<b>Gradient Descent Minimize Step Size (sample units)</b>	This is the size of steps each control point is moved along the gradient-based direction when searching for a minimum of the error. The registration does not take into account the size of an image sample in each dimension, so <b>this step size is specified in units of a sample</b> . E.g., a value of 0.5 will move the control point in increments of <b>half of a sample</b> while searching for the minimum error. The minimum step size is 0.1 sample and the maximum is 5 samples.
<b>Gradient Descent Minimize Max Search Steps</b>	This is the maximum number of steps the control point is moved along the gradient-based direction when searching for a minimum of the error. If the distance from the control point current position to the bounding polygon/polyhedron of its neighboring control points along the gradient-based direction is less than this specified value, then the computed distance value will be used for this control point instead. The minimum number of steps is 1 and the maximum is 100.
<b>Iteration options (one iteration=move each control point once)</b>	
<b>Convergence limit (min change rate for one iteration)</b>	This is a rate of change threshold such that if the rate the error changes between starting and ending of each iteration is smaller than this value, then the registration terminates.
<b>Maximum Number of Iterations</b>	If the convergence limit is not satisfied after this many iterations have been executed, then the registration terminates. The range of possible values is 1 to 100.
<b>Results</b>	
<b>Create deformation field image</b>	If this checkbox is enabled, then the deformation image is computed after the last pass of registration is performed. The image that is created has the same dimensions as the target and the registered source images, and has the name of the source image appended with <i>deformation</i> suffix. The deformation image will be automatically displayed along with the registered source image upon completion of registration. Refer to Figure 2 for details.
<b>OK</b>	Applies the algorithm to the input image that you chose in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

**Figure 5. B-Spline Two-Pass Automatic Registration 2D/3D dialog box (continued)**

This dialog is shown with the default options for a two-pass registration. As previously mentioned and as indicated in the dialog, a single iteration involves the movement of each interior control point in a gradient descent minimization manner.

The following is an example of the dialog which is displayed for 2.5D intensity/color image registration: This dialog is show with the default options for a **single-pass registration**. If the **Perform two-pass registration** check box is selected, then the dialog will change in a manner similar to that for the 2D/3D image registration shown in Figure 5.

General	
<b>Register</b>	<p>The Register source field is already filled with the name of the currently selected 2.5D image. The reference slice is selected from among the slices in the selected image.</p> <p>Radio buttons are used to select whether the reference slice is the the <b>adjacent</b> (previous) slice or a <b>particular slice number</b> in the image. If the <b>Reference Slice</b> radio button is selected, then a drop down list of slices numbers is enabled where the middle slice is selected by default.</p>
<b>Cost function</b>	<p>This drop down list contains the following choices for cost measure to use during registration: Least Squares, Correlation Ratio, and Normalized Mutual Information.</p>
<b>Perform two-pass registration</b>	<p>This dialog is shown with the default options for single-pass registration. If the <b>Perform two-pass registration</b> option is enabled, then the dialog will change to make options for <b>two-pass registration</b> available. Refer to Figure 7.</p>
<b>Transformation Options</b>	
<p>The following is a discussion of the dialog options which control each registration pass, one set of options per pass. For more information refer to "Transformation options" on page 124.</p>	
<b>Sub-sample image for speed</b>	<p>If this option is enabled, the image will be subsample in power of 2 for each dimension. Subsampling increases performance of the algorithm, but it results lower accuracy of registration.</p>

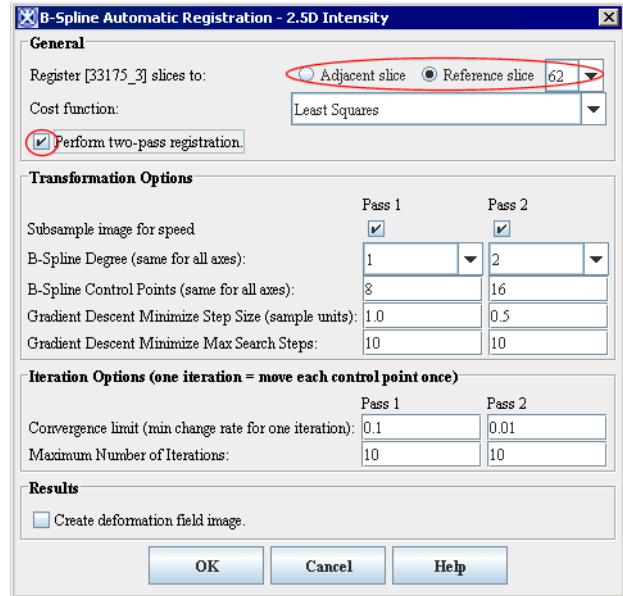
**Figure 6. B-Spline Single-Pass Automatic Registration 2.5D dialog box**



<b>B-Spline Degree (same for all axes)</b>	This is the degree of the B-Spline to use for all axes of the image. The drop list of options show the values of 1 (linear), 2 (quadratic), 3 (cubic), and 4 which are available. Although, the underlying BSpineBasisf class supports any degree, this dialog restricts the degrees to these values. Choosing a smaller degree executes faster than a larger degree, but a higher degree offers a higher order of continuity.
<b>B-Spline Control Points (same for all axes)</b>	This is the number of B-spline control points to use for all axes of the image. The minimum number of control points is the larger of the number 2 and the degree of the B-Spline. This dialog limits the maximum number of control points to one-half of the number of samples in the smallest dimension.
<b>Gradient Descent Minimize Step Size (sample units)</b>	This is the size of steps each control point is moved along the gradient-based direction when searching for a minimum of the error. The registration does not take into account the size of an image sample in each dimension, so <b>this step size is specified in units of a sample</b> . E.g., a value of 0.5 will move the control point in increments of <b>half of a sample</b> while searching for the minimum error. The minimum step size is 0.1 sample and the maximum is 5 samples.
<b>Gradient Descent Minimize Max Search Steps</b>	This is the maximum number of steps the control point is moved along the gradient-based direction when searching for a minimum of the error. If the distance from the control point current position to the bounding polygon/polyhedron of its neighboring control points along the gradient-based direction is less than this specified value, then the computed distance value will be used for this control point instead. The minimum number of steps is 1 and the maximum is 100.
<b>Iteration options (one iteration=move each control point once)</b>	
<b>Convergence limit (min change rate for one iteration)</b>	This is a rate of change threshold such that if the rate the error changes between starting and ending of each iteration is smaller than this value, then the registration terminates.
<b>Maximum Number of Iterations</b>	If the convergence limit is not satisfied after this many iterations have been executed, then the registration terminates. The range of possible values is 1 to 100.
<b>Results</b>	
<b>Create deformation field image</b>	If this checkbox is enabled, then the deformation image is computed after the last pass of registration is performed. The image that is created has the same dimensions as the target and the registered source images, and has the name of the source image appended with <i>deformation</i> suffix. The deformation image will be automatically displayed along with the registered source image upon completion of registration. Refer to Figure 2 for details.
<b>OK</b>	Applies the algorithm to the input image that you chose in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

**Figure 6. B-Spline Single-Pass Automatic Registration 2.5D dialog box (continued)**

General	
<b>Register</b>	<p>The Register source field is already filled with the name of the currently selected 2.5D image. The reference slice is selected from among the slices in the selected image.</p> <p>Radio buttons are used to select whether the reference slice is the the <b>adjacent</b> (previous) slice or a <b>particular slice number</b> in the image. If the <b>Reference Slice</b> radio button is selected, then a drop down list of slices numbers is enabled where the middle slice is selected by default.</p>
<b>Cost function</b>	<p>This drop down list contains the following choices for cost measure to use during registration: Least Squares, Correlation Ratio, and Normalized Mutual Information.</p>
<b>Perform two-pass registration</b>	<p>This dialog is shown with the default options for two-pass registration.</p>
<b>Transformation Options</b>	
<p>The following is a discussion of the dialog options which control each registration pass, one set of options per pass. For more information refer to "Transformation options" on page 124.</p>	
<b>Sub-sample image for speed</b>	<p>If this option is enabled, the image will be subsample in power of 2 for each dimension. Subsampling increases performance of the algorithm, but it results lower accuracy of registration.</p>
<b>B-Spline Degree (same for all axes)</b>	<p>This is the degree of the B-Spline to use for all axes of the image. The drop list of options show the values of 1 (linear), 2 (quadratic), 3 (cubic), and 4 which are available. Although, the underlying BSpineBasisf class supports any degree, this dialog restricts the degrees to these values. Choosing a smaller degree executes faster than a larger degree, but a higher degree offers a higher order of continuity.</p>
<p><b>Figure 7. B-Spline Automatic Two-Pass Registration 2.5D dialog box</b></p>	



<b>B-Spline Control Points (same for all axes)</b>	This is the number of B-spline control points to use for all axes of the image. The minimum number of control points is the larger of the number 2 and the degree of the B-Spline. This dialog limits the maximum number of control points to one-half of the number of samples in the smallest dimension.
<b>Gradient Descent Minimize Step Size (sample units)</b>	This is the size of steps each control point is moved along the gradient-based direction when searching for a minimum of the error. The registration does not take into account the size of an image sample in each dimension, so <b>this step size is specified in units of a sample</b> . E.g., a value of 0.5 will move the control point in increments of <b>half of a sample</b> while searching for the minimum error. The minimum step size is 0.1 sample and the maximum is 5 samples.
<b>Gradient Descent Minimize Max Search Steps</b>	This is the maximum number of steps the control point is moved along the gradient-based direction when searching for a minimum of the error. If the distance from the control point current position to the bounding polygon/polyhedron of its neighboring control points along the gradient-based direction is less than this specified value, then the computed distance value will be used for this control point instead. The minimum number of steps is 1 and the maximum is 100.
<b>Iteration options (one iteration=move each control point once)</b>	
<b>Convergence limit (min change rate for one iteration)</b>	This is a rate of change threshold such that if the rate the error changes between starting and ending of each iteration is smaller than this value, then the registration terminates.
<b>Maximum Number of Iterations</b>	If the convergence limit is not satisfied after this many iterations have been executed, then the registration terminates. The range of possible values is 1 to 100.
<b>Results</b>	
<b>Create deformation field image</b>	If this checkbox is enabled, then the deformation image is computed after the last pass of registration is performed. The image that is created has the same dimensions as the target and the registered source images, and has the name of the source image appended with <i>deformation</i> suffix. The deformation image will be automatically displayed along with the registered source image upon completion of registration. Refer to Figure 2 for details.
<b>OK</b>	Applies the algorithm to the input image that you chose in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

Figure 7. B-Spline Automatic Two-Pass Registration 2.5D dialog box (continued)

---

## Transformation options

The following is a discussion of the dialog options which control each registration pass, one set of options per pass:

**B-spline Degree (same for all axes):** This is the degree of the B-spline to use for all axes of the image. Even though the `BSplineRegistration2Df` and `BSplineRegistration3Df` classes allow for specification of separate degrees for each axis, this dialog restricts the degrees to be the same. The drop list of options show the values of 1 (linear), 2 (quadratic), 3 (cubic), and 4 which are available. Although, the underlying `BSplineBasisf` class supports any degree, this dialog restricts the degrees to these values. Choosing a smaller degree executes faster than a larger degree, but a higher degree offers a higher order of continuity.

**B-spline Control Points (same for all axes):** This is the number of B-spline control points to use for all axes of the image. Even though the `BSplineRegistration2Df` and `BSplineRegistration3Df` classes allow for specification of separate number of control points for each axis, this dialog restricts the numbers of control points to be the same. Initially, the control points are evenly spaced in each dimension, and the registration algorithm moves the interior control points to minimize the error difference between the target image and the registered source image. The minimum number of control points is the larger of the number 2 and the degree of the B-spline. This dialog option limits the maximum number of control points to one-half of the number of samples in the smallest dimension. Choosing more control points generally results in better overall fitting, but a limitation may be that the control points will be closer to each other and that will limit how much a control point can move to minimize the error (in order to satisfy the constraint that an interior control point must be within the bounding polygon/polyhedron formed by its neighboring control points). Also, since choosing more control points means that control points will be closer to each other, it may be helpful to reduce the maximum number of steps for searching for the gradient descent minimum.

Choosing more control points rather than fewer should result in a longer execution time. Even though more control points means fewer samples (for the same size input image) are affected by a single control point, the fact that there are more control points to move and each one requires a gradient descent sample search for a minimum error most likely results in the longer execution time.

Choosing more control points may require a different convergence limit. Since the convergence limit is tested once per iteration and an iteration involves moving each interior control point once, it may be that more control points could result in more (or less) overall changes in the error between starting and ending one iteration.

**Gradient Descent Minimize Step Size (sample size units):** This is the size of steps each control point is moved along the gradient-based direction when searching for a minimum of the error. The registration does not take into account the size of an image sample in each dimension, so this step size is specified in units of a sample. For instance, a value of 0.5 will move the control point in increments of half a sample while searching for the minimum error. The minimum step size is 0.1 sample and the maximum is 5 samples.

**Gradient Descent Minimize Max Search Steps:** This is the maximum number of steps the control point is moved along the gradient-based direction when searching for a minimum of the error. If the distance from the control points current position to the bounding polygon/polyhedron of its neighboring control points along the gradient-based direction is less than this specified value, then the computed distance value will be used for this control point instead. The minimum number of steps is 1 and the maximum is 100.

**Convergence limit (min change rate for one iteration):** This is a rate of change threshold such that if the rate the error changes between starting and ending of each iteration is smaller than this value, then the registration terminates. As mentioned previously, the rate of change is computed as

$$(\text{previousError} - \text{currentError}) / \text{previousError}$$

where `previousError` is the error measured before starting an iteration and `currentError` is the error measured after completing an iteration. The error being measured here depends on the particular cost function selected.

**Maximum Number of Iterations:** If the convergence limit is not satisfied after this many iterations have been executed, then the registration terminates. The range of possible values is 1 to 100.

## OUTPUT FILES

After the B-spline registration algorithm finishes running, the following files can be found in the image directory:

- the source and target image,
- the \*.xmp files with the recorded history for both source and target images,
- the NonLinear Transformation (\*.nlt) file that contains the information about the performed B-spline transformation, such as the dimensionality of the transformed image, the degree of the B-spline, the number of control points, and the values of the control points. This file is used for the Transform Nonlinear algorithm, refer to “Transform Nonlinear” ,
- the result file.

## Class implementation for MIPAV

The following classes are part of the `mipav/model/structures` package:

- B-splineBasis
- B-splineBasisDiscretef
- BSplineLattice2Df
- BSplineLattice3Df
- BSplineRegistrationBasef
- fBSplineRegistration2Df

The following classes are also part of the `mipav/model/algorithms/registration` package:

- AlgorithmRegBSpline
- AlgorithmRegBSpline2D
- AlgorithmRegBSpline3D
- AlgorithmRegBSpline25D



## DTI Create List File

This utility takes as an input a DICOM study directory or a PAR/REC file extracts the information and creates the following files:

- A list file with metadata information, such as shown in Figure 2;
- A sorted relative path file that contains the relative paths to the slice files;
- And a b-matrix file;
- If the Register option is activated, it, first, registers the image volume to the reference volume, and then creates a list file, a path file, and a b-matrix file for the result image. For more information, refer to Section “If the Registration option is activated” on page 134.

These three files are used for the diffusion tensor (**D**) computation in the MIPAV Diffusion Tensor Imaging module.

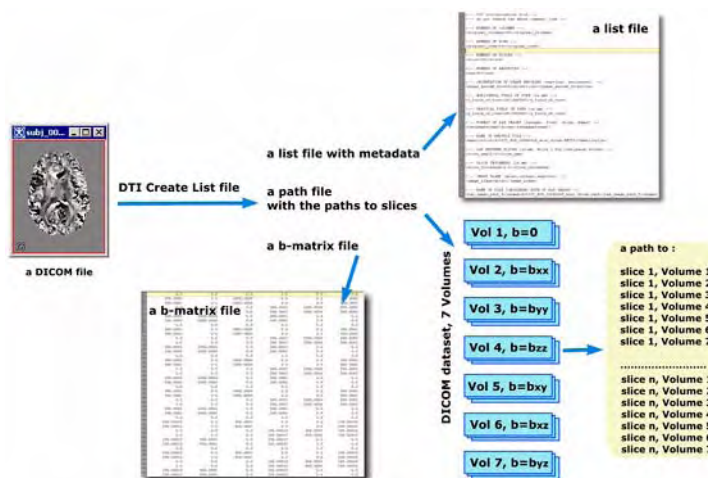


Figure 1. The outline of the method



---

## Outline of the method

---

### LIST FILE

The list file contains metadata which describe the study and experiment. It includes the information about the acquired image, such as image format, number of slices, slice thickness, image plane, and also it keeps the name of the path file.

#### DICOM

For DICOM images, this information is extracted directly from the file header. A sample list file is shown in Figure 2.

#### PAR/REC

The method extracts metadata from the PAR file and uses them to form the list file. A sample list file for PAR/REC dataset is shown in Figure 3.

---

### PATH FILE

The method forms the path file that has the relative paths to all image volumes and slices arranged in the following order (the example below is given for a 7 Volume image):

#### Example 1: a path file structure

A path to slice 1 Volume1  
A path to slice 1 Volume2  
.....  
A path to slice 1 Volume7  
A path to slice 2 Volume1  
A path to slice 2 Volume2  
.....  
A path to slice 2 Volume7  
.....  
.....  
A path to slice n Volume1

A path to slice n Volume2

.....

A path to slice n Volume7

### **DICOM**

In order to get this path information for DICOM images, the method parses the study directory and sorts files into lists using a series number that is available in the DICOM image header. Then, based on the filename, it gets a volume number, for example,

`mipav_dicom_dti_series11_volume10005.dcm`, and sorts within lists using the volume number and instance number that comes from the corresponding DICOM tag.

### **PAR/REC**

The reconstructed image (a \*.REC file) is a data file that has all DWIs stored in a single volume pixel by pixel in Integer data format (16-bit or 8 bit per pixel) defined by the header parameter.

### **For Phillips PAR/REC format, the method does the following:**

- 1** In the `study_name_proc` directory it creates a subdirectory with the name that corresponds to the name of your s PAR/REC file;
- 2** It decomposes the \*.REC file into and slices in Float format one \*.RAW +XMP file per slice and copies these files into the `your_study_name_slices` directory. The slice name includes the volume number. See Figure 4;
- 3** Finally, the method forms the path file that contains the relative paths to all slices arranged by volumes as shown in Example 1 (this example is given for a 7 Volume image). See also Figure 4.

```

<!-- DTI initialization file -->
<!-- do not remove the above comment line -->
<!-- NUMBER OF COLUMNS -->
<original_columns>64</original_columns>
<!-- NUMBER OF ROWS -->
<original_rows>64</original_rows>
<!-- NUMBER OF SLICES -->
<slice>38</slice>
<!-- NUMBER OF BMATRICES -->
<nim>42</nim>
<!-- ORIENTATION OF PHASE ENCODING (vertical, horizontal) -->
<phase_encode_direction>vertical</phase_encode_direction>
<!-- HORIZONTAL FIELD OF VIEW (in mm) -->
<x_field_of_view>200.000000</x_field_of_view>
<!-- VERTICAL FIELD OF VIEW (in mm) -->
<y_field_of_view>200.000000</y_field_of_view>
<!-- FORMAT OF RAW IMAGES (integer, float, dicom, dummy) -->
<rawimageformat>dicom</rawimageformat>
<!-- NAME OF BMATRIX FILE -->
<bmatrixfile>433672_BOS_20040628_minc_dicom.BMTXT</bmatrixfile>
<!-- GAP BETWEEN SLICES (in mm. Write 0 for contiguous slices) -->
<slice_gap>0.0</slice_gap>
<!-- SLICE THICKNESS (in mm) -->
<slice_thickness>4.0</slice_thickness>
<!-- IMAGE PLANE (axial,coronal,sagittal) -->
<image_plane>axial</image_plane>
<!-- NAME OF FILE CONTAINING PATH OF RAW IMAGES -->
<raw_image_path_filename>433672_BOS_20040628_minc_dicom.path<
/raw_image_path_filename>
  
```

**Figure 2. A sample list file for a DICOM image**

```

<!-- DTI initialization file -->
<!-- do not remove the above comment line -->
<!-- NUMBER OF COLUMNS -->
<original_columns>128</original_columns>
<!-- NUMBER OF ROWS -->
<original_rows>128</original_rows>
<!-- NUMBER OF SLICES -->
<slice>50</slice>
<!-- NUMBER OF BMATRICES -->
<nim>33</nim>
<!-- ORIENTATION OF PHASE ENCODING (vertical, horizontal) -->
<phase_encode_direction>vertical</phase_encode_direction>
<!-- HORIZONTAL FIELD OF VIEW (in mm) -->
<x_field_of_view>240.000</x_field_of_view>
<!-- VERTICAL FIELD OF VIEW (in mm) -->
<y_field_of_view>240.000</y_field_of_view>
<!-- FORMAT OF RAW IMAGES (integer, float, dicom, dummy) -->
<rawimageformat>float</rawimageformat>
<!-- NAME OF BMATRIX FILE -->
<bmatrixfile>1_9_5_8_1.BMTXT</bmatrixfile>
<!-- GAP BETWEEN SLICES (in mm. Write 0 for contiguous slices) -->
<slice_gap>0.000</slice_gap>
<!-- SLICE THICKNESS (in mm) -->
<slice_thickness>2.500</slice_thickness>
<!-- IMAGE PLANE (axial,coronal,sagittal) -->
<image_plane>axial</image_plane>
<!-- NAME OF FILE CONTAINING PATH OF RAW IMAGES -->
<raw_image_path_filename>1_9_5_8_1.path</raw_image_path_filename>
  
```

**Figure 3. A sample list file for a PAR/REC image**

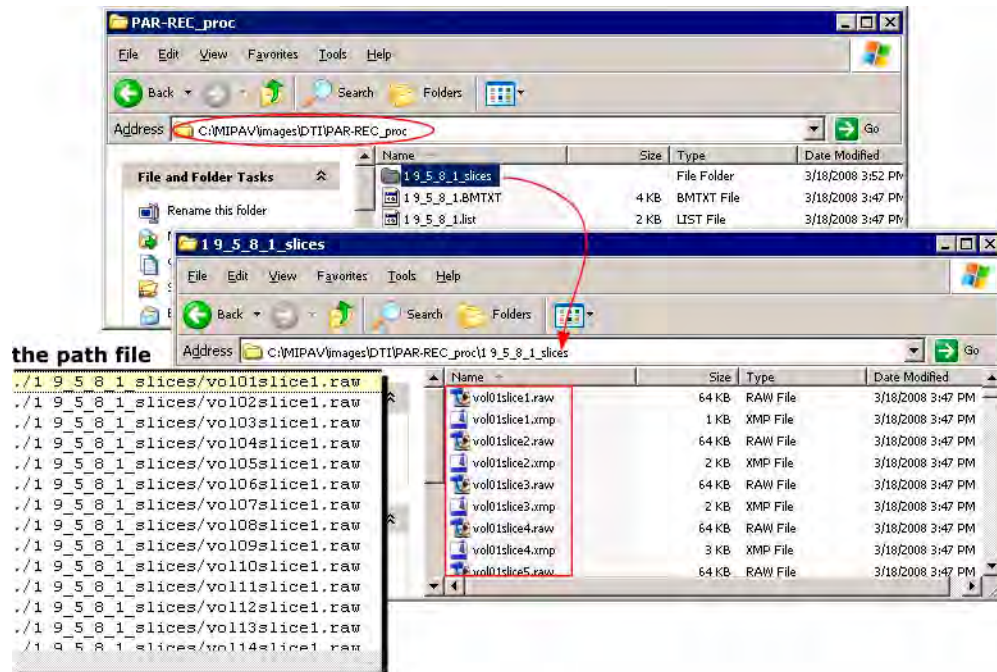


Figure 4. A sample path file for a Phillips PAR/REC image

## B-MATRIX FILE FOR DICOM IMAGE DATASETS

If a user provides a b-matrix file, the method simply copies that file to the `study_name_proc` directory and renames it using the correct naming syntax, which is as follows: `your_study_name.BMTXT`.

If the user did not provide the b-matrix file, the method gets b-values for each volume set directly from the provided image dataset.

### For DTI image datasets:

- If the image was acquired using the Siemens magnet, then the method extracts b-value from a public tag `0018,0024`. The b-value is the number following after the keystroke `ep_b` and before `#`.
- If the image was acquired using the General Electric (GE) magnet and the version of the software used is 8 or less, then the method extracts

the b-value from a private tag 0019,10B9 and the b-value is the number following after the ep\_b keystroke.

- If the image was acquired using the GE magnet and the software's version is bigger than 8, then the method extracts the b-value from a private tag 0043,1039 and the b-value is the first number in the string.

#### **For eDTI image datasets:**

- If the image was acquired using the Siemens magnet, then the method extracts the b-value from a public tag 0018,0024 and the b-value is the number following after the ep\_b keystroke and before #.
- If the image was acquired using the GE magnet, then the method extracts the b-value from a private tag 0043,1039 and the b-value is the first number in the string.

---

**Note:** if for some reason the algorithm fails to extract b values and create the b-matrix file, the user has the option to provide his own b-matrix file, and then rerun the algorithm.

---

#### **B-matrix file for PAR/REC image datasets**

For the software version 4.1 and greater, the method extracts b values and gradient information from a PAR file. It creates the b-matrix file that has the name `your_study_name.BMTXT` and copies it to the same directory where the list and path files are stored.

For the software version 4.0 and lesser the method extracts only the b values, but the gradient file must be provided by the user.

If the user provides the b-matrix file, the method simply copies that file to the `your_study_name_proc` directory and renames it using the correct naming syntax, which is as follows `your_study_name.BMTXT`.

---

## **IF THE REGISTRATION OPTION IS ACTIVATED**

- 1** The algorithm registers the image dataset to the reference volume using either the default, or the user defined settings of the Optimized

Automatic Registration algorithm. For more information, refer to “Optimized Automatic Registration 3D” on page 596.

Note: For DICOM files you should provide the path to the first slice of the B0 volume. See Figure 5. For PAR/REC files the algorithm uses the B0 volume as a reference volume and it finds it automatically.

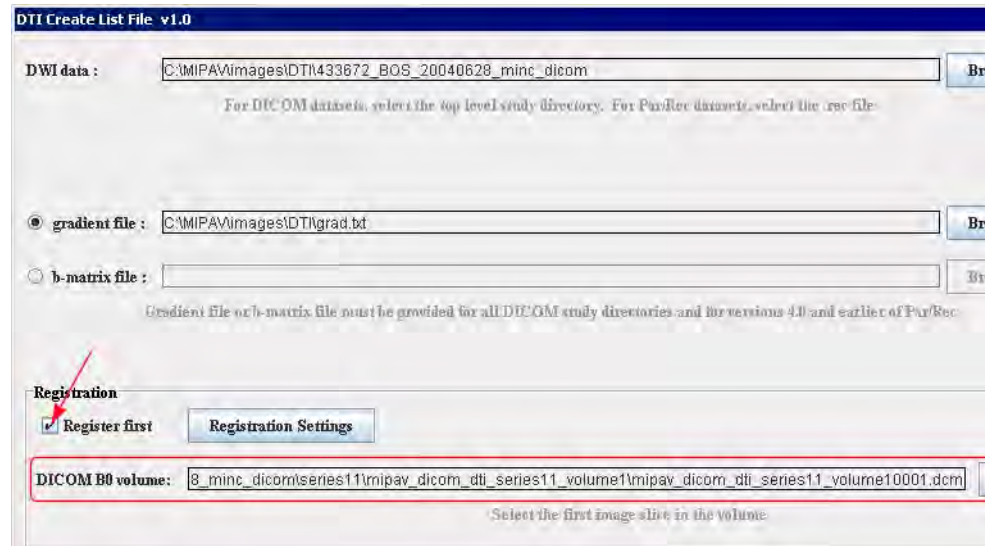


Figure 5. The DTI Create List File dialog box with the Registration option activated for DICOM images

- 2** Under the `your_study_name_proc` directory, it creates the subdirectory with the following name `your_study_name_registered_slices`.
- 3** In that subdirectory the algorithm saves the result image volume after registration.
- 4** When the registration is done, the algorithm creates the b-matrix, list and path files and stores them in the `your_study_name_proc` directory. In this case the file names are as follows:
  - `your_study_name_registered.BMTXT` – for the b-matrix file,
  - `your_study_name_registered.list` – for the list file,
  - `your_study_name_registered.path` – for the path file.
- 5** These files are stored in the `your_study_name_proc` directory.

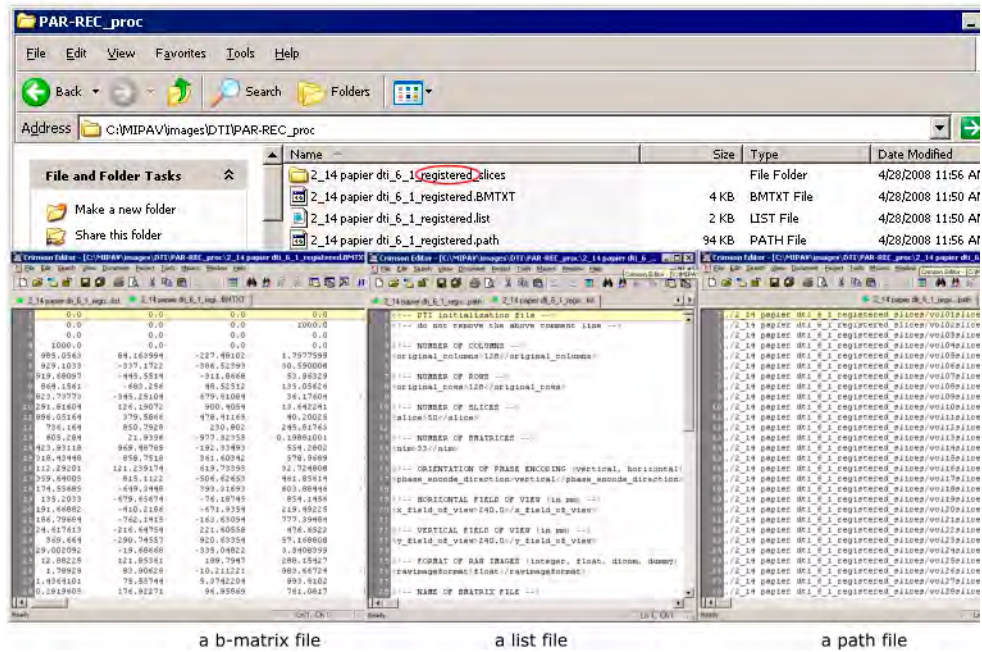


Figure 6. The b-matrix, list and path files and the default directory for the PAR/REC dataset if the Register First option is activated in the dialog box

## Running the DTI Create List File utility

### FOR DICOM DATASETS

- 1 Run MIPAV.
- 2 Navigate to File > Diffusion Tensor Imaging > Create List File.
- 3 Run the DTI Create List File utility.
- 4 In the DTI Create List File dialog box that appears:
  - Navigate to the study path directory;
  - Open the gradient file or the b-matrix file (depending on what you have). Note that for interleaved datasets, you must provide the b-matrix file.



- 5 If you need to register your dataset before running the Create List File utility, check the **Register first** box. Provide the path to the first slice of the B0 volume, see Figure 7. And then click the Registration Setting button. This will call the Optimized Automatic Registration dialog box. In the dialog box, adjust the registration parameters and then press OK. For more information, refer to “If the Registration option is activated” on page 134 and “Optimized Automatic Registration dialog box options” on page 609.
- 6 Press OK. The plug-in begins to run and the list file, path file, and b-matrix file appear in the sub directory of the study path directory that has the following name `your_study_name_proc`. See Figure 2 and Figure 7.

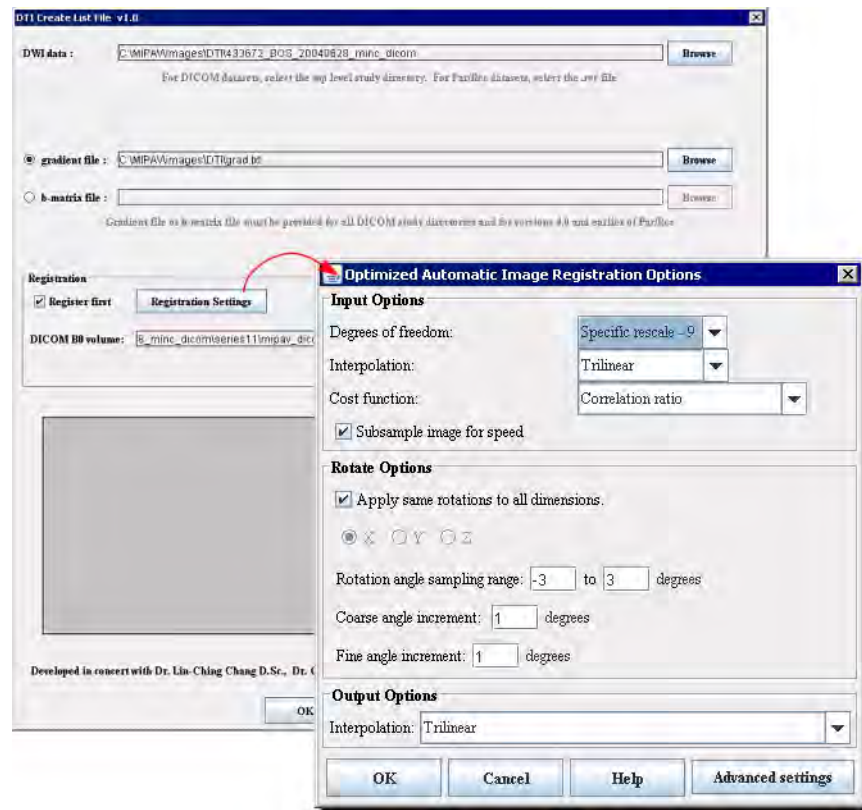


Figure 7. The DTI Create List File dialog box opened for a DICOM image **For PAR/REC datasets**

- 7 Run MIPAV.

- 8 Navigate to File > Diffusion Tensor Imaging > Create List File.
- 9 Run the DTI Create List File utility.
- 10 In the DTI Create List File dialog box that appears:
  - Open the REC file;
  - Open the gradient file if the software version is less than 4.0. Note that for interleaved datasets, you must also provide the b-matrix file.
- 11 If you need to register your dataset before running the Create List File utility, check the **Register first** box. This activates the Registration Settings button. Click the button to open the Optimized Automatic Registration dialog box. In the dialog box, adjust the registration parameters and then press OK. For more information, refer to “If the Registration option is activated” on page 134 and “Optimized Automatic Registration dialog box options” on page 609.

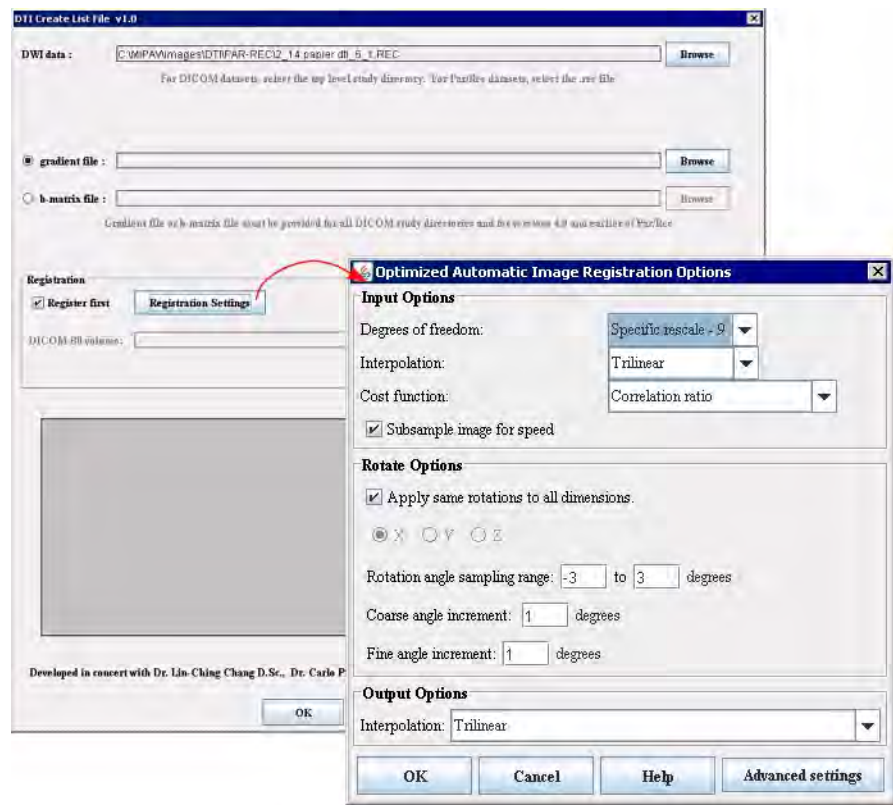


Figure 8. Running the DTI Create List File plug-in for PAR/REC datasets

- 12** Press OK. The plug-in begins to run and the list file, path file, and b-matrix file appear in the sub directory of the study path directory that has the following name `your_study_name_proc`. See Figure 6 and Figure 8.

---

## DTI Color Display

*A common problem in biomedical sciences is the in vivo identification and analysis of anatomical structures. The DTI Color Display plug-in introduces a novel technique to visualize nerve fiber tracts in diffusion-weighted magnetic resonance imaging data. The method uses the improved diffusion tensor-based Directionally Encoded Color (DEC) schemes which can be used, for example, to identify and display white matter fiber tracts in the human brain.*

### Background

The method takes as an input indices of diffusion anisotropy and the principal directions of diffusion (eigenvectors of diffusion tensor  $\mathbf{D}$ ), which characterize specific features of the diffusion process. And then, it uses a color (RGB and/or HSV) representation of the components of the eigenvector associated with the largest eigenvalue of the diffusion tensor ( $V_{\max}$ ) weighted by some measure of diffusion anisotropy. This technique provides an effective way to display the directions of anisotropic structures in a single brain image, because:

- The principal directions of diffusion provide information on the spatial orientation of anisotropic structures that may be useful to further characterize tissue structural anatomy.
- Anisotropy measures yield information about the structure of normal and pathological white matter.

**There are three issues or three sources of misinterpretation in the DEC schemes that DTI Color Display method improves:**

- Antipodal symmetry of the eigenvectors of  $\mathbf{D}$ , refer to Section “Antipodal Symmetry of the Eigenvectors of  $\mathbf{D}$ ” on page 141
- Nonlinear effects of the display device, refer to Section “Color processing” on page 146
- Properties of human color vision, also refer to Section “Color processing” on page 146

---

## ANTIPODAL SYMMETRY OF THE EIGENVECTORS OF D

The method is based on assumptions that the orientation of fibers is described by the diffusion tensor eigenvector associated with the largest eigenvalue,  $V_{\max}$  and the parallel and anti-parallel vectors convey the same information. Thus, to have a unique vector representation, the vectors have to be mapped to only one hemisphere in a spherical coordinate system. The plane that divides this spherical coordinate system into two hemispheres, e.g., *the plane of discontinuity* can be positioned in an arbitrary direction; however, within this plane, fiber orientation will still be represented by two antipodally symmetric vectors  $V_{\max}$  and  $-V_{\max}$ .<sup>1</sup>

To reduce these discontinuity artifacts, the method uses a spherical coordinate system fixed to defined anatomical landmarks and independent from the laboratory reference frame in which the tensor dataset is acquired. Within that anatomical coordinate system the eigenvector  $V_{\max}$  is a unit vector completely determined by its polar *theta* and azimuthal *phi* angles that are defined as shown in Figure 1. A particular color representation is shown on a unit sphere, which is called *the color representation sphere*.

Only the vectors expressed in the upper hemisphere, e.g. *z more or equal to 0* or *theta* in a range of  $[0, \text{Pi}/2]$  and the XY plane is *the plane of discontinuity*, are used for unique color representation. The lower hemisphere is just an antipodally symmetric copy of the upper hemisphere. Refer to Figure 1 in which *the color representation sphere* is split along the plane of discontinuity. The brightness of a DEC map is weighted by a measure of diffusion anisotropy.<sup>2</sup>

**The following considerations were also used to design the color representation sphere:**

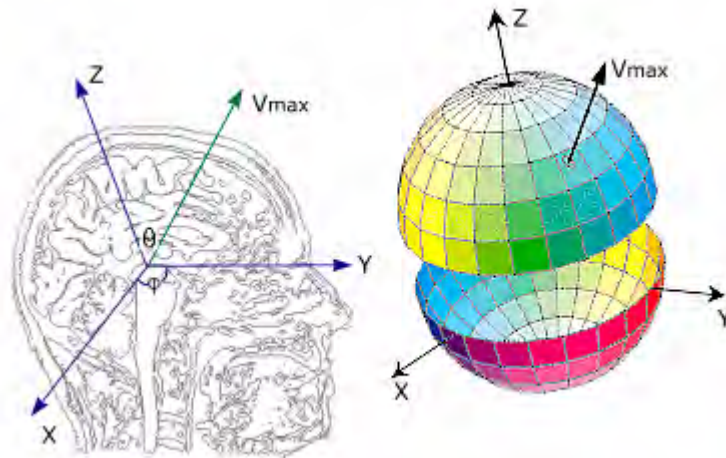
- Color differences are proportional to actual differences in direction, characterized either by the Euclidian distance between vectors or by the angle between them.

---

1. In statistics, directional data with such antipodal symmetry are referred to as axial data.

2. Pi = 3.14159265

- Only the directional information contained in the  $\mathbf{D}$  is displayed, and the representation is not affected by parameters related to the magnitude of diffusivity, such as Trace  $\mathbf{D}$  or diffusion coefficients.
- Commonly identifiable color groups (e.g., red, green, blue, yellow, cyan, magenta, etc.) are associated with a characteristic direction (e.g., direction along x, y, z, and directions bisecting the XY, YZ, XZ planes).



**Figure 1. Definition of the anatomical reference frame for brain images.** The yz plane corresponds to the sagittal plane aligned with the inter-hemispheric fissure; the y axis corresponds to the anterior posterior intercommissural line. When no anisotropy filter is used, a particular color representation can be shown on a unit color representation sphere. The color representation sphere shown in this picture is for the no symmetry scheme (described in Section “No Symmetry” on page 143). The sphere is anchored to the anatomic frame of reference; thus, for a given scheme, an anisotropic structure has the same color independent of the view (axial, coronal, or sagittal). The figure also shows how  $V_{\max}$  for a generic direction is mapped on the color sphere.

The method allows different directions in the color maps to be represented with the same color if some form of symmetry exists between these directions. However, it causes an orientational ambiguity, because the color representation of orientation is not unique, in some situations using such symmetries is justified by existing symmetries in the human anatomy. With regard to which type of symmetry is used, the method offers the following options: No Symmetry, Rotational Symmetry, Mirror Symmetry, and Absolute Value. See Figure 6.

---

Proper rotations should be done to convert image data from the laboratory frame of reference to the specified anatomical frame of reference before applying the method. See also Section “Importance of midsagittal alignment” on page 150.

---

## No Symmetry

This representation is implemented by relating the azimuthal angle of the vector  $\phi$  to color hue and the polar angle  $\theta$  less or equal  $\pi/2$  to color saturation. For HSV scheme (H=Hue, S=Saturation, V=Value) the following equations are used:

EQUATION 1

$$H = (\phi - \phi_R + 2\pi) \bmod(2\pi)$$

$$S = \frac{\sin(p_S \theta)}{\sin(p_S \pi/2)}$$

$$V = 1$$

Where,  $\phi_R$  is the  $\phi$  angle of the vector  $V_{\max}$  lying in the XY plane that will be represented in a pure red color, and  $p_S$  is an heuristic parameter that takes values in the interval [0,1]. When  $p_S < 0$ , S is approximately a linear function of  $\theta$ . No Symmetry representation is unique because a particular color describes a single direction, but it suffers from discontinuity artifacts.

## Rotational Symmetry

In this scheme, any vector and its pair rotated by  $180^\circ$  around the z axis have the same color representation, e.g.  $V_{\max}(\theta, \phi) = V_{\max}(\theta, \phi + \pi)$  or  $V_{\max}(v_x, v_y, v_z) = V_{\max}(-v_x, -v_y, v_z)$ . This representation is implemented in a fashion similar to the No Symmetry approach, but with the constraint  $H(\phi) = H(\phi + \pi)$ .

EQUATION 2

$$H = 2(\phi - \phi_R + 2\pi) \bmod 2\pi$$

Where,  $\phi_R$  is the  $\phi$  angle of the vector lying in the XY plane that will be represented in a pure red color.  $p_S$  in the interval [0,1] is the same heuristic parameter as used for No Symmetry and equations for S and V are also the same that were used for No Symmetry. See also Equation 1.

Rotational Symmetry scheme removes the discontinuity artifacts but does not provide a unique representation of directions and is, also, perceptually highly nonuniform.

### Mirror Symmetry

The mirror symmetry representation scheme implies that any two vectors that are mirror images of each other relative to the YZ plane of the anatomical frame of reference will have the same color representation, e.g.  $V_{\max}(-v_x, v_y, v_z) = V_{\max}(v_x, v_y, v_z)$ .

EQUATION 3

$$H = 2((\phi - \phi_R + \pi) \bmod \pi)$$

Where,  $\phi$  is an azimuthal angle of  $V_{\max}(|v_x|, v_y, v_z)$ . Mirror symmetry exists between many structures in the left and the right brain hemispheres. Equations for S and V are also the same that were used for No Symmetry, refer to Equation 1.

### Absolute Value

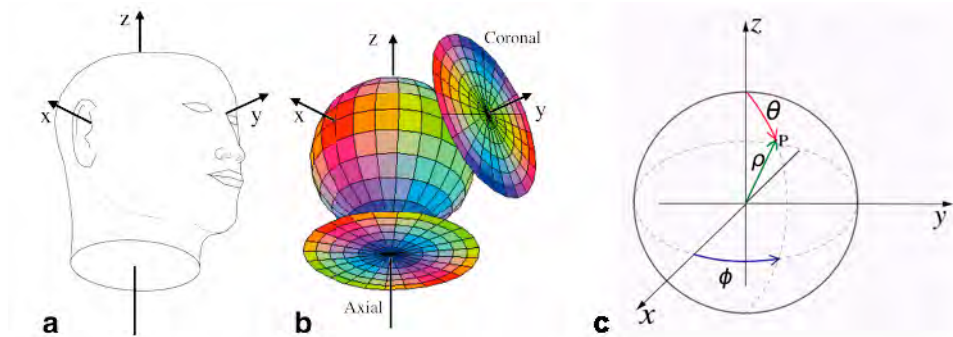
The Absolute Value scheme implies that the sign of any vector is unimportant; only the absolute value is important. This yields a simple representation in RGB color space in which RGB components are associated with the absolute value of the components of the vector, e.g.

EQUATION 4

$$R_I = |v_x|, G_I = |v_y|, B_I = |v_z|$$



Where  $R_I$ ,  $G_I$ , and  $B_I$  are the RGB components of the color representation of  $V_{max}$ , which are defined in the interval (0,1). See also Figure 2.



**Figure 2.** a: Anatomic reference frame. b: Color representation sphere for the absolute value scheme. c: A unit sphere for your reference. In (b) the projection of the color representation sphere onto a color circle for axial and coronal views is shown, assuming that axial and coronal slices are exactly perpendicular to the z and y axes of the anatomical reference frame, respectively. Note that the grid for coronal view is not the projection of the constant  $\theta$  and  $\phi$  lines but rather the grid of constant  $\theta_v$  and  $\phi_v$  as defined in the text.

## COLOR CIRCLES

The method uses color circles to relate a chosen color representation sphere with its corresponding fiber orientation. The color circles are constructed by projecting a chosen color representation sphere onto viewing plane using the equal area Lambert's projection. According to the standard convention for viewing MR images, for axial images the representation color sphere is viewed from below (negative z axis of our anatomical coordinate system pointing at the viewer, see Figure 2-a) and for coronal images from the front (positive y axis pointing at the viewer, see Figure 2-b). See also Figure 6.<sup>1</sup>

In Figure 2, the orientation of a structure represented with a given color, can be approximately obtained from the color circle by placing one end of a pencil, whose length is approximately equal to the radius of the circle, at the center of the circle and the other end right above the corresponding color.

1. Lambert's Cylindrical Equal Area Map projection implies the following: meridians are equally spaced; parallels get closer near poles; parallels are sines; true scale at equator.

## Building color circles

Figure 2 shows a grid of parallels and meridians added to the circle at 15 degree intervals to indicate the angles  $\theta_v$  (angle at which the structure is positioned relative to the viewing direction) and  $\phi_v$  (angle within the viewing plane). Here,  $\phi_v$  and  $\theta_v$  are related to the 2D cylindrical coordinates  $\rho$  and of the equal area color circle as  $\rho = 2\sin(\theta_v/2)$  and  $\phi = \phi_v$  as shown in Figure 2.

---

## COLOR PROCESSING

The color processing sequence provided by the method aims at faithfully representing 3D antipodally symmetric vector data, taking into account artifacts associated with particular color display and human color perception. The final goal is that the differences in orientation of such data be proportional to the perceived color differences, as much as that is possible. Since color perception properties, as well as those of antipodally symmetric data, do not allow this to be achieved precisely, the method resorts to approximate schemes using symmetries in data, then it provides tools for correcting the direct color representation using red, green and blue shifts, which aim to equalize different perceptual qualities of those primary colors. Then intensity scaling corrects for our non-uniform perception of luminosity, and finally the gamma correction should be applied to correct for the common non-linearity effect introduced by different color display.

### Color processing heuristic parameters

**Anisotropy Max and Min** are used to set the Anisotropy filter. After the filter is set, only structures where the Anisotropy index was larger than anisotropy min and smaller than anisotropy max are represented, the others are truncated and set to black. `\{DTIAdjustExp.png\}`.

**Gamma Correction** is used to encode RGB values into luminosity produced by the device, e.g. your computer monitor.

**pB Sat.Blue** is used to decrease the saturation of blue hue to achieve better uniformity of perceived brightness for different hue values (e.g. red, green, and blue), and more faithful representation of diffusion anisotropy. It is also used as a basis for calculation brightness of red color.

**pG Dim Green** is used to adjust brightness for green color.

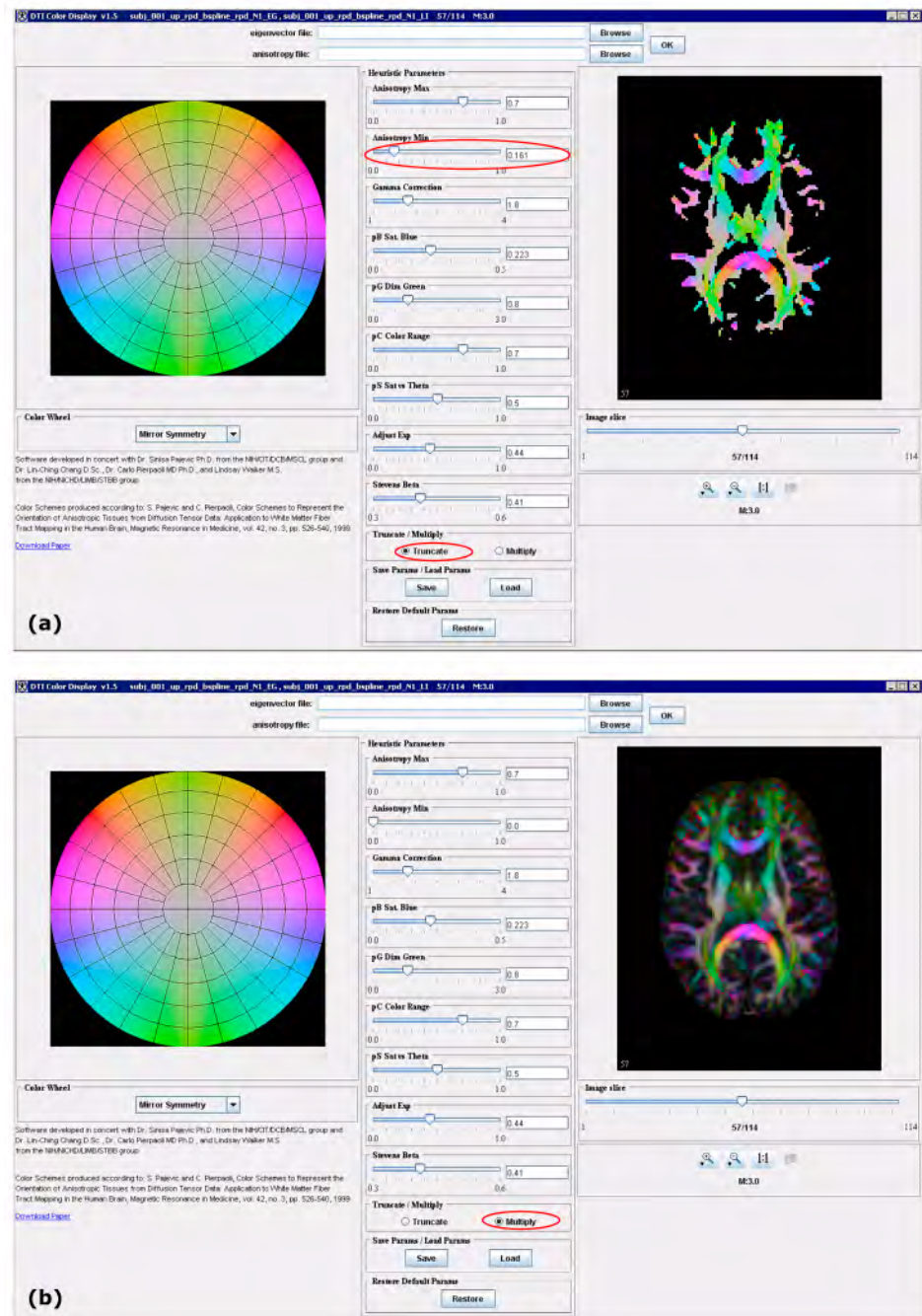
**pC Color Range.** pC Color Range = 0 yields the maximal range of colors without regard to uniform brightness. pC Color Range = 1 reduces the range of colors, but allows a more faithful representation of anisotropy and a more uniform color representation.

**pS Sat vs. Theta** is used to modify the dependence between the saturation and the angle *theta*.

**Adjust Exp** is used to modify the nonlinear relationship between the perceived brightness and the anisotropy measure. The algorithm uses two types of anisotropy filters in color representations. See also Figure 3.

- In the first one, each of the color components is, first, normalized based on the *anisotropy min* and *max* values, e.g.  $newIntensityValue = (value - anisotropyMin) / (anisotropyMax - anisotropyMin)$ , and then the normalized value is multiplied to a weighting factor equals *AdjustExp/StephensBeta*. This corresponds to the **Multiply** option activated in the DTI Color Display dialog box, refer to Figure 3- (b).
- In the second one, only structures where the lattice index was larger than *anisotropy min* are represented, the others are truncated and set to black. This corresponds to the Truncate option activated in the DTI Color Display dialog box, refer to Figure 3- (a).

**Stevens law exponent** or *StephensBeta* which employs the simple nonlinear relationship between the true luminosity of monochromatic light and its perceived brightness, is also treated as heuristic parameter, i.e. it can be adjusted.



**Figure 3. Applying the Adjust Exp. slider. The DTI Color Display dialog box shows two representations of the same image. (a): the Truncate option activated, thus the image intensity values, which are lower than the Anisotropy min threshold, are set to zero. (b): the Multiply option is activated, that makes the brightness of a given color to be modulated by the degree of anisotropy.**

## Blue, red and green shifts

Uniform brightness is important to view the directional information and the degree of anisotropy simultaneously. In other situations, when the degree of anisotropy is less important one may use a larger range of colors. This is governed by the parameter *pC Color Range*.

To preserve uniform brightness, the method, first, shifts blue toward the white point according to the following equations:

EQUATION 5

$$R_S = C_B + (1 - C_B) * R_I$$

$$G_S = C_B * B_I + (1 - C_B) * G_I$$

$$B_S = B_I$$

$$C_B = \max\left[\frac{3}{2}p_B\left(b - \frac{1}{3}\right)p_C, 0\right]$$

$$b = \frac{B_I}{R_I + G_I + B_I}$$

Where  $R_S$ ,  $G_S$ , and  $B_S$  are shifted intensities for red, green and blue components,  $p_B$  represents the *pB Sat.Blue* parameter and  $p_C$  represents *pC Color Range*. See also “Color processing heuristic parameters” on page 146.

Then, the algorithm performs the red shift in the same manner, but using the smaller *pB Sat.Blue*, e.g. *pB for red shift* =  $p_B/4$ .

## And finally, it performs the green shift:

It calculates new shifted intensities for red, green and blue components taking into account the following heuristic parameters: *pG Dim Green*, *pC Color Range*, and *StephensBeta*. See also “Color processing heuristic parameters” on page 146.

## Gamma correction

The default values for the gamma correction exponent and StephensBeta are as follows: gamma = 2.2 and StephensBeta = 0.4. However, the method treats them as heuristic parameters, i.e., they can be empirically adjusted. See also Figure 3.

---

## IMPORTANCE OF MIDSAGITTAL ALIGNMENT

In real life, the anatomic coordinate system and the reference system for image display do not necessarily coincide, i.e., axial, coronal, and sagittal images are not lying in the XY, XZ, and YZ planes of the anatomical coordinate system, respectively. In such case, the color coordinate system appears shifted relative to the color circle grid and the above proposed symmetries in color representation (e.g. rotational symmetry and mirror symmetry) would not be easily seen on the color circle. If the misalignment between the anatomical coordinate system and the reference system for image display is severe, images should be rotated and interpolated before applying the method.

---

## IMAGE TYPES

The algorithm works with any image types supported by MIPAV. However, in order to run the algorithm you should provide two files: *an eigenvector file* and *an anisotropy file*. The eigenvector file should be 4D image file with nine time slices, where the first three slices contain the principal eigenvector information. The anisotropy file should be 3D file with anisotropy value range (0,1). The files must have the same 3D dimensionality.



Figure 4. An eigenvector (left) and anisotropy (right) files opened in MIPAV.

---

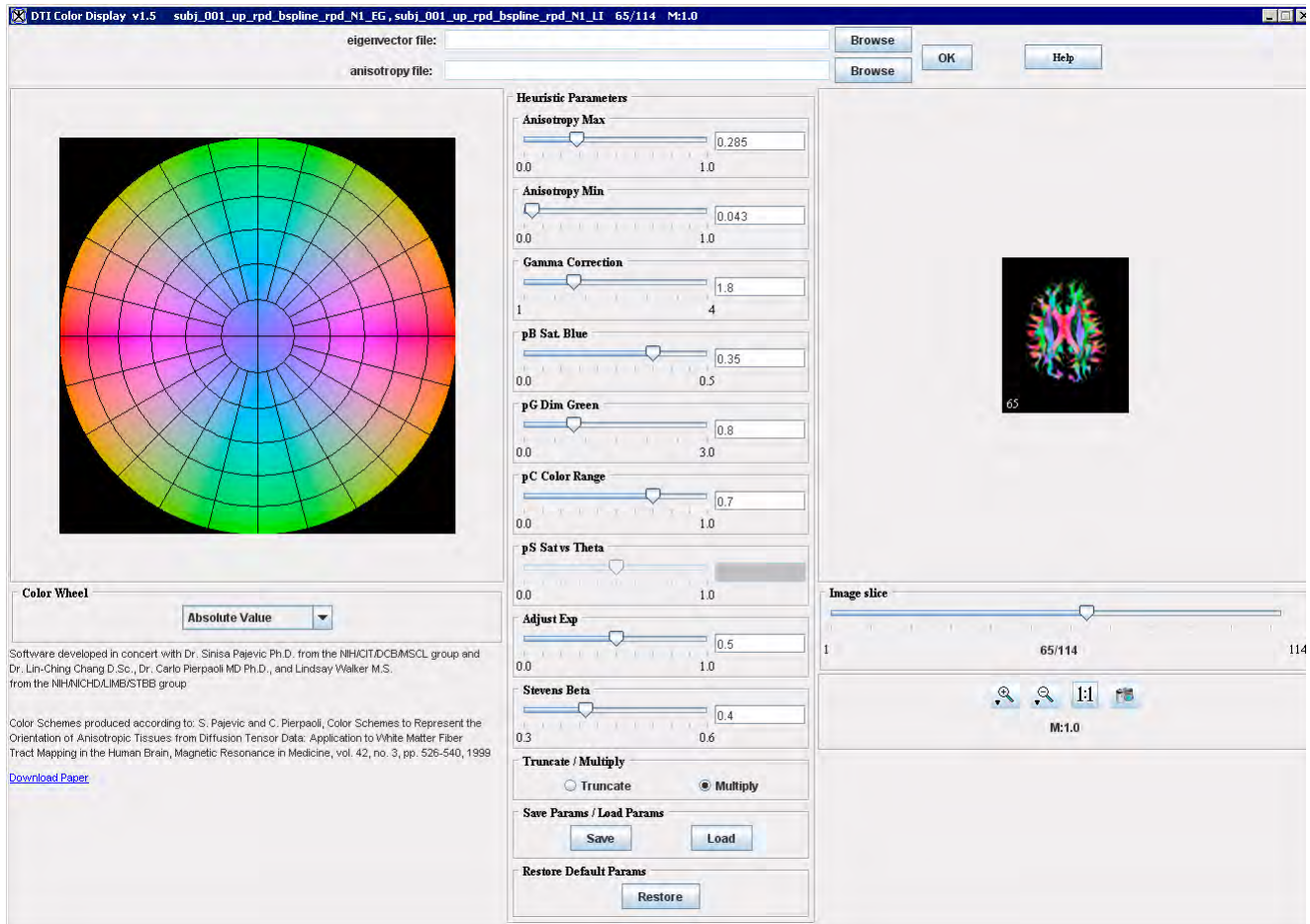
## REFERENCES

Sinisa Pajevic and Carlo Pierpaoli [1999]. "Color Schemes to Represent the Orientation of Anisotropic Tissues From Diffusion Tensor Data: Application to White Matter Fiber Tract Mapping in the Human Brain", *Magnetic Resonance in Medicine*, vol. 42, pp 526---540.

## Applying DTI Color Display

- 1 Install the plug-in.
- 2 Then select Plugins > General > DTI Color Display. The DTI Color Display dialog box opens displaying the Absolute Value color wheel.
- 3 In the dialog box, select the *eigenvector* and *anisotropy* files.
- 4 Click OK. The image appears in the Image window.
- 5 Use the Image slice slider to select the slice of interest.
- 6 Select the color wheel which you would like to apply to the image. Note that:
  - Absolute Value yields a simple representation in RGB color space in which RGB components are associated with the absolute value of the components of the diffusion vector.
  - In No Symmetry scheme, a particular color describes a single direction, but this representation suffers from discontinuity artifacts.
  - Rotational Symmetry scheme removes the discontinuity artifacts but does not provide a unique color representation of directions and is, also, perceptually highly nonuniform.
  - Mirror Symmetry implies that any two vectors that are mirror images of each other relative to the YZ plane of the anatomical coordinate system will have the same color representation. See also Figure 6.

Then, use the Heuristic Parameters sliders to adjust the parameters to achieve the most satisfactory image representation. See also Section "Color processing heuristic parameters" on page 146.



**Figure 5. DTI Color Display dialog box.**

- In the Color Wheel window, use the list box to select the color scheme.
- Use the Heuristic Parameters options to adjust colors and also alleviate artifacts associated with the properties of your color display and perception. For more information, refer to Section “Color processing heuristic parameters” on page 146.
- Use the Save parameters option to save current parameters in a file.
- Use the Load parameters option to load saved parameters.
- Use the Restore Default parameters option to restore default values.
- In the Image window, use the Image Slice slider to select the slice that you would like to examine.
- Use the Zoom In and Zoom Out to magnify or minify the image.
- Use 1:1 option restore the image to its original size.
- And you can also capture the image into a new frame using the Camera icon. Note that this option becomes available only if the image is restored to its original size.



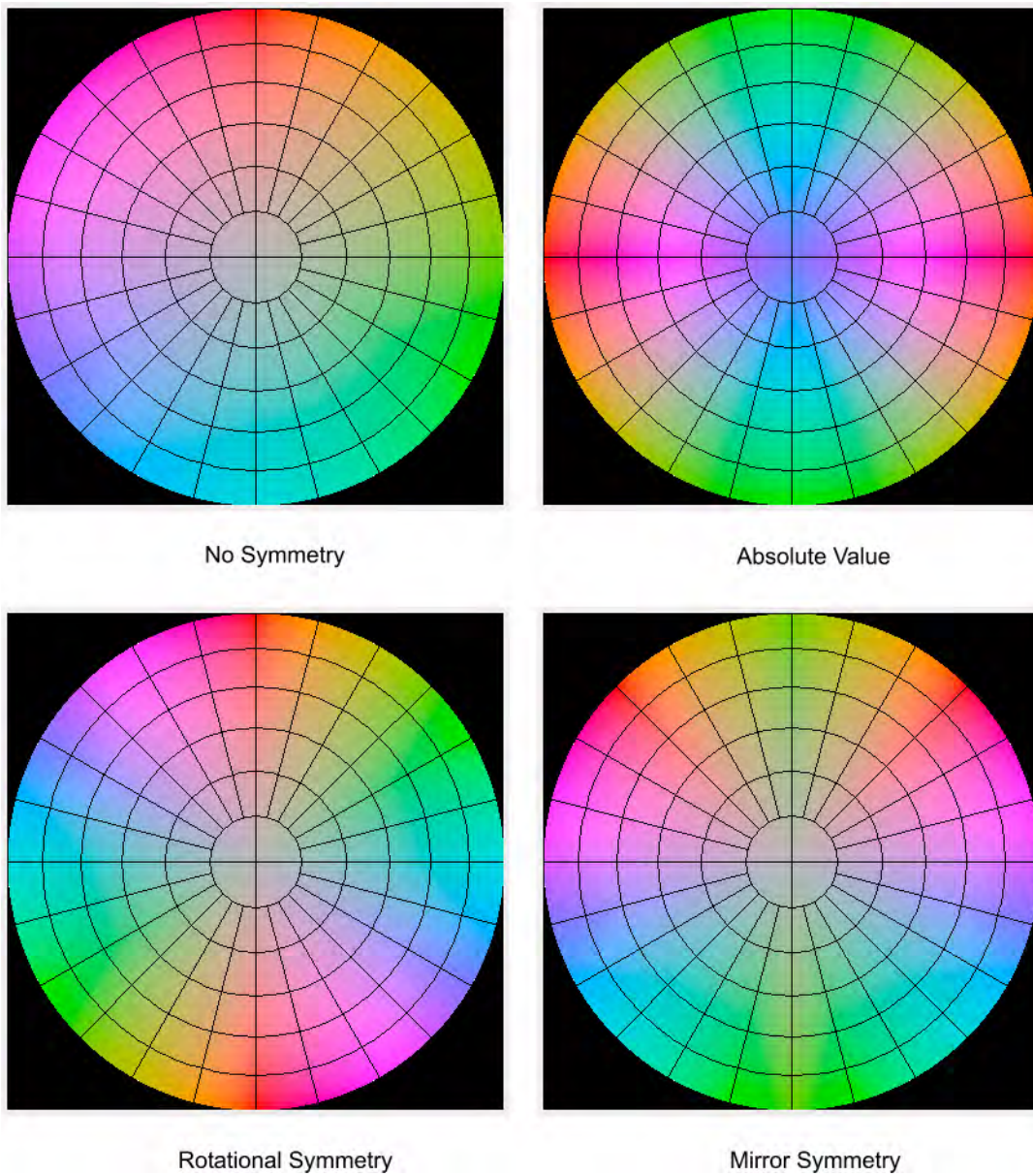


Figure 6. Four types of color wheels which are used in the algorithm.

## Edge Detection: Zero X Laplacian

*Edge detection is possibly the most common method for segmenting objects in medical images. Typically, these algorithms find edges that form a closed contour, which completely bound an object. Currently, MIPAV offers two edge detection algorithms: Zero X Laplacian, and “Edge Detection: Zero X Non-Maximum Suppression”.*

### Background

The Laplacian is a 2-D isotropic measure of the 2-nd spatial derivative of an image and can be defined as:

EQUATION 1

$$\nabla^2 f = \frac{d^2 f}{dx^2} + \frac{d^2 f}{dy^2}$$

in 2D images and

EQUATION 2

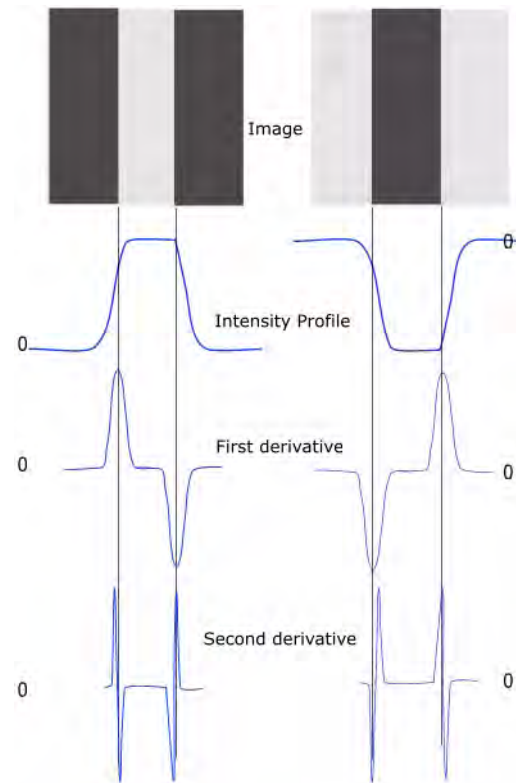
$$\nabla^2 f = \frac{d^2 f}{dx^2} + \frac{d^2 f}{dy^2} + \frac{d^2 f}{dz^2}$$

in 3D images.

The Laplacian of an image highlights regions of rapid intensity change and therefore can be used for edge detection. Zero X Laplacian algorithm finds edges using the *zero-crossing property of the Laplacian*. The zero crossing detector looks for places in the Laplacian of an image where the value of the Laplacian passes through zero – i.e. points where the Laplacian changes its sign. Such points often occur at the edges in images – i.e. points where the intensity of the image changes rapidly.

Figure 1 on page 155 shows that in the approach of a change in intensity, the Laplacian response is positive on the darker side, and negative on the lighter side. This means that at a reasonably sharp edge between two regions of uniform but different intensities, the Laplacian response is:

- zero at a long distance from the edge,
- positive just to one side of the edge,
- negative just to the other side of the edge,
- zero at some point in between, on the edge itself.



**Figure 1. Edge detection by derivative operations – for given two images (light stripe on a dark background and dark stripe on a light background) the first and second derivatives were taken. Note that the second derivative has a zero crossing on the location of each edge**

However, as a second order derivative, the Laplacian is very sensitive to noise, and thus, to achieve the best result, it should be applied to an image that has been smoothed first. This pre-processing step reduces the high frequency noise components prior to the differentiation step.

The concept of the Zero X Laplacian algorithm is based on convolving the image with 2D Gaussian blur function, first, and then applying the Laplacian. The 2D Gaussian blur function can be defined as

EQUATION 3

$$h(x, y) = -exp\left\{\frac{x^2 + y^2}{2\sigma^2}\right\}$$

where  $\sigma$  –*sigma* is a standard deviation. This function blurs the image with the degree of blurring proportional to *sigma*.

In that case the 2-D Laplacian of the Gaussian (that is the second derivative of *h* with respect to *r*) is centered on zero and with Gaussian standard deviation *sigma* has the form of

EQUATION 4

$$\nabla^2 f = -\left\{\frac{r^2 - \sigma^2}{\sigma^4}\right\}exp\left\{-\frac{r^2}{2\sigma^2}\right\}$$

where  $r^2 = x^2 + y^2$ . See also Figure 2.

Since the convolution operation is associative, the method convolves the Gaussian smoothing filter with the Laplacian filter, first, and then convolves this hybrid function (defined by Equation 4) with the image to achieve the required result.

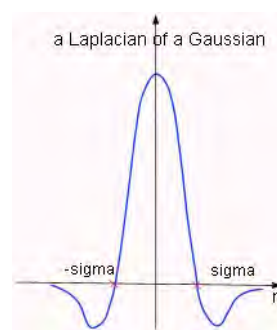


Figure 2. The picture shows a cross section of the circularly symmetric function defined by Equation 4. The function crosses the zero point at *r* equals + or -*sigma*

Once the image has been Laplacian of a Gaussian filtered, the algorithm detects the zero crossings. Note that, zero crossings might occur at any place

where the image intensity gradient starts increasing or starts decreasing, and this may happen at places that are not obviously edges. To cure that problem, the method applies the Marching Squares algorithm to detect those zero crossings that corresponds to edges. To learn more about Marching Squares, refer to Volume 2, *Algorithms*, “Extract Surface (Marching Cubes)” .

The resulting image is an unsigned byte image with values of 255 at the edges and 0 elsewhere.

---

## IMAGE TYPES

You can apply this algorithm to 2D and 3D grayscale images.

---

## REFERENCES

See the following references for more information about this algorithm:

<http://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm>.

<http://homepages.inf.ed.ac.uk/rbf/HIPR2/zeros.htm>

Rafael C. Gonzalea, Richard E. Woods, “Digital Image Processing” Second Edition, Prentice Hall, 1992, pp. 581–585.

## Applying the Zero X Laplacian algorithm

To run this algorithm, complete the following steps:

- 1** Open an image of interest;
- 2** Select Algorithms > Edge Detection > Zero X Laplacian. The EdgeLap dialog box opens. For the dialog box options, refer to Figure 4.
- 3** Complete the fields in the dialog box. Click OK.

The algorithm begins to run, and a status window appears. When the algorithm finishes, the resulting image appears as an unsigned byte mask in a new image window as shown in Figure 3.

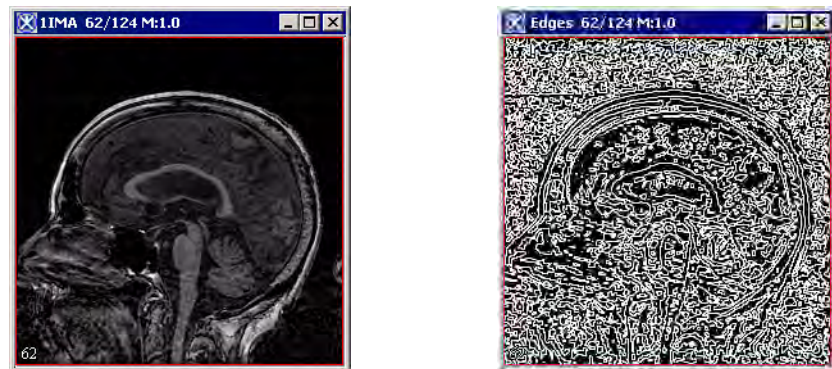


Figure 3. The original image and its unsigned byte mask with detected edges

<b>Scale of the Gaussian</b>	Specifies the standard deviation of the Gaussian filter in X, Y, and Z directions which were used to determine edges in the image.	
<b>Use image resolutions to normalize Z scale</b>	If checked, Z scale*(X resolution/Y resolution) will be used to generate the edge map.  If unchecked, Z scale will be used in generation of the edge map.	
<b>Options</b>		
<b>Use separable convolution kernels</b>	If checked, the algorithm convolves the image with X kernel and Y kernel independently, and then combines the result images.	
<b>Process each slice independently (2.5D)</b>		Applies the algorithm to each slice individually.
<b>Destination</b>	New image – this is the default option; Replace image – this option cannot be selected.	
<b>Process</b>	<b>Whole Image</b> – if checked the edge detection will be calculated for the whole image; <b>VOI region</b> – if checked the edge detection will be calculated for the selected VOI.	
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.	
<b>Cancel</b>	Closes the dialog box.	
<b>Help</b>	Displays online help for this dialog box.	

Figure 4. The EdgeLap dialog box options

## Edge Detection: Zero X Non-Maximum Suppression

*This method produces an edge map of the zero-crossings of the non-maximum suppression for 2D and 2.5D images. Edges are defined as the union of points for which the gradient magnitude assumes a maximum in the gradient direction.*

### Background

For a 2D image, let's introduce a local coordinate system (u, v) such that at any point in the image  $P_0(x_0, y_0)$  v-axis is parallel to the gradient direction at  $(x_0, y_0)$  and the u-axis is perpendicular to it, i.e.,

EQUATION 1

$$\begin{pmatrix} \cos\alpha \\ \sin\alpha \end{pmatrix} = \frac{1}{\sqrt{L_x^2 + L_y^2}} \begin{pmatrix} L_x \\ L_y \end{pmatrix}_{(x_0; y_0)}$$

Where L is a convolution of an image (represented as a function  $f(x, y)$  with the Gaussian kernel defined as:

EQUATION 2

$$g(x, y) = \frac{1}{2\pi t} \exp\left\{-\frac{x^2 + y^2}{2t}\right\}$$

so that

EQUATION 3

$$L(x, y; t) = g(x, y; t) \circ f(x, y)$$

where the circle represents convolution of  $g(x, y; t)$  and  $f(x, y)$  and semicolon in the argument of g implies that the convolution is performed only over the variables x and y, while the scale parameter  $t$  after the semicolon just

indicates which scale-space level is being defined. Directional derivatives in this local (u,v) system are related to partial derivatives in the Cartesian coordinate system by the following system of equations,

EQUATION 4

$$\begin{aligned}L_u &= \sin \alpha * L_x - \cos \alpha * L_y \\L_v &= \cos \alpha * L_x + \sin \alpha * L_y\end{aligned}$$

where, at a given point  $P_0$

EQUATION 5

$$\begin{aligned}\cos \alpha &= \frac{L_x}{\sqrt{(L_x^2 + L_y^2)}} \\ \sin \alpha &= \frac{L_y}{\sqrt{L_x^2 + L_y^2}}\end{aligned}$$

And this (u,v) coordinate system is characterized by the fact that one of the two first-order derivatives  $-L_u$ , is zero, e.g.

EQUATION 6

$$L_v = \frac{L_x^2 + L_y^2}{\sqrt{L_x^2 + L_y^2}} = \sqrt{L_x^2 + L_y^2}$$

EQUATION 7

$$L_u = \frac{(L_x L_y - L_x L_y)}{\sqrt{(L_x^2 + L_y^2)}} = 0$$

EQUATION 8

$$L_v^2 = L_x^2 + L_y^2$$



Below, are equations used in the method for calculating the second and third derivative –  $L_{vv}$  and  $L_{vvv}$ .

EQUATION 9

$$L_{vv} = (\cos \alpha * L_x + \sin \alpha * L_y)^2$$

EQUATION 10

$$L_{vv} = \cos^2 \alpha * L_{xx} + 2 * \cos \alpha * \sin \alpha * L_{xy} + \sin^2 \alpha * L_{yy} = \frac{L_x^2 * L_{xx} + 2 * L_x * L_y * L_{xy} + L_y^2 * L_{yy}}{L_x^2 + L_y^2}$$

EQUATION 11

$$\begin{aligned} L_{vvv} &= (\cos^2 \alpha * L_{xx} + 2 * \cos \alpha * \sin \alpha * L_{xy} + \sin^2 \alpha * L_{yy}) \cos \alpha * L_x + \sin \alpha * L_y) \\ &= \cos^3 \alpha * L_{xxx} + 3 * \cos^2 \alpha * \sin \alpha * L_{xxy} + 3 * \cos \alpha * \sin^2 \alpha * L_{xyy} + \sin^3 \alpha * L_{yyy} \end{aligned}$$

EQUATION 12

$$L_{vvv} = \frac{L_x^3 * L_{xxx} + 3 * L_x^2 * L_y * L_{xxy} + 3 * L_x * L_y^2 * L_{xyy} + L_y^3 * L_{yyy}}{(L_x^2 + L_y^2)^{3/2}}$$

EQUATION 13

$$L_v^3 * L_{vvv} = L_x^3 * L_{xxx} + 3 * L_x^2 * L_y * L_{xxy} + 3 * L_x * L_y^2 * L_{xyy} + L_y^3 * L_{yyy}$$

According to the notion of non-maximum suppression, an edge point is defined as a point at which the gradient magnitude assumes a maximum in the gradient direction. In terms of directional derivatives, such definition can be also expressed as a combination of two conditions:

- The second-order directional derivative in the v-direction  $L_{vv}$  being zero;

- And the third-order directional derivative in the same direction  $L_{VVV}$  being negative, e.g.:

EQUATION 14

$$\begin{cases} L_{VV} = 0, \\ L_{VVV} < 0 \end{cases}$$

Since only the sign information is important, these conditions can be restated as:

EQUATION 15

$$\begin{cases} L_v^2 * L_{VV} = 0 \\ L_v^3 * L_{VVV} < 0 \end{cases}$$

Here,  $L$  is the scale-space representation of an image  $f(x,y;t)$  at a certain scale  $t$ , and the last equation defines the edges of  $f$  at that scale.

This condition can also be formulated in terms of zero-crossings of the partial derivative of  $L$  with respect to the scale parameter (i.e., as a directional derivative in the vertical scale direction). For more information about the zero-crossings method for detecting edges used in MIPAV, refer to MIPAV User Manual, Volume 2, Algorithms, “Edge Detection: Zero X Laplacian” on page 154.

The method evaluates detected zero-crossings using the Marching Squares algorithm to determine only those zero-crossings that corresponds to edges. To learn more about the Marching Squares algorithm, refer to MIPAV User Manual, Volume 2, Algorithms, “Extract Surface (Marching Cubes)” on page 395.

The resulting image is an unsigned byte image with values of 255 at the edges and 0 elsewhere.



Figure 1. The original image and its unsigned byte mask with detected edges.

---

## IMAGE TYPES

You can apply this algorithm to 2D and 3D grayscale images.

---

## REFERENCES

Bart M. ter Haar Romeny (Ed.), "Geometry-Driven Diffusion in Computer Vision", Chapter 2: *Linear Scale-Space II: Early Visual Operations* by Tony Lindeberg and Bart M. ter Haar Romeny, Kluwer Academic Publishers, 1994, pp. 39–46.

MIPAV User Manual, Volume 2, Algorithms, "Edge Detection: Zero X Laplacian" on page 154.

Tony Lindeberg [1996]. "Edge detection and ridge detection with automatic scale selection" *Technical report* ISRN KTH/NA/P-96/06-SE, May 1996, Revised August 1998. *Int. J. of Computer Vision*, vol 30, number 2, 1998. (In press). Shortened version in Proc. CVPR'96, San Francisco, June 1996. <http://www.nada.kth.se/~tony>

## Applying the Zero X Non-Maximum Suppression algorithm

To run this algorithm, complete the following steps:

- 1 Open an image of interest;
- 2 Select Algorithms > Edge detection > Zero x non-maximum suppression. The EdgeNMSuppression dialog box opens. For the dialog box options, refer to Figure 2.
- 3 Complete the fields in the dialog box. Click OK.

The algorithm begins to run, and a status window appears. When the algorithm finishes, the resulting image appears as an unsigned byte mask in a new image window as shown in Figure 3.

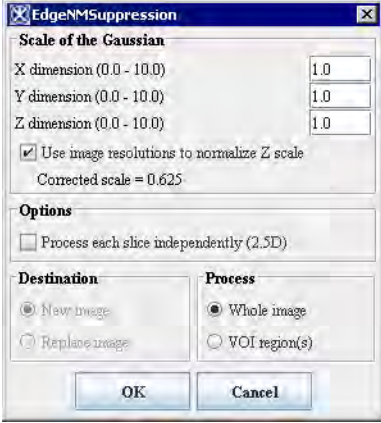
<b>Scale of the Gaussian</b>	<p>Specifies the standard deviation (from 1.0 to 10.0) of the Gaussian filter in X, Y, and Z directions which were used to determine edges in the image.</p> <p>Note that larger standard deviation produces an image with smaller number of edges. Smoothing the image before applying the algorithm also reduce the number of detected edges. Refer to Figure 3 for more information.</p>	
<b>Use image resolutions to normalize Z scale</b>	<p>If checked, the following equation – <math>Z\ scale * (X\ resolution / Z\ resolution)</math> will be used to generate the edge map.</p> <p>If unchecked, only Z scale will be used in generation of the edge map.</p>	
<b>Process each slice independently (2.5D)</b>	Applies the algorithm to each image slice individually.	
<b>Destination</b>	<b>New image</b> – this is the default option; <b>Replace image</b> – this option cannot be selected.	
<b>Process</b>	<b>Whole Image</b> – if checked the edge detection will be calculated for the whole image; <b>VOI region</b> – if checked the edge detection will be calculated for the selected VOI.	
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.	
<b>Cancel</b>	Closes the dialog box.	

Figure 2. The EdgeNMSuppression dialog box options

Help Displays online help for this dialog box.

Figure 2. The EdgeNMSuppression dialog box options

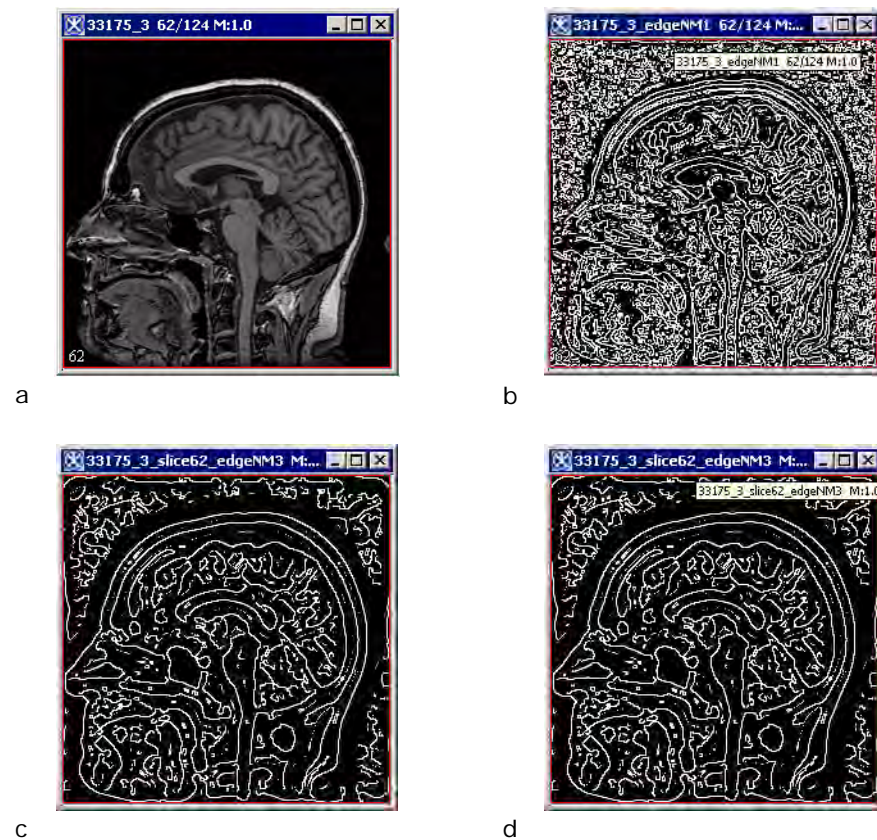
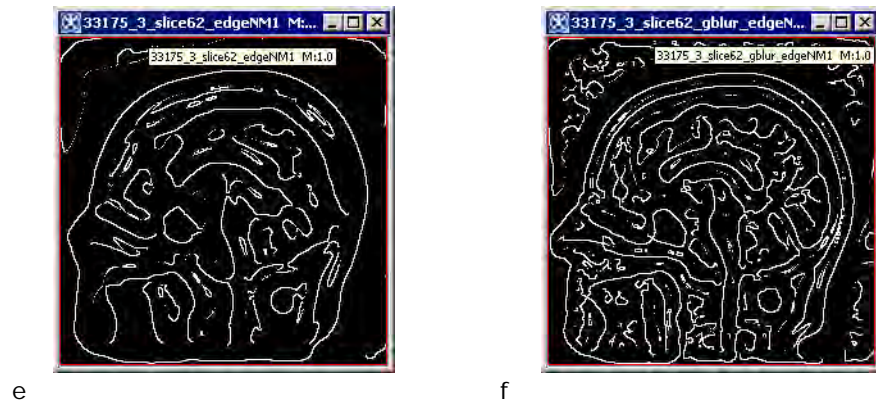


Figure 3. The original image (a) and its unsigned byte mask with detected edges depending on the standard deviation: (b) – standard deviation is set to 1; (c) – standard deviation is set to 4; (d) – standard deviation is set to 6; (e) – standard deviation is set to 8; (f) – standard deviation is set to 6, but the original image has been smoothed before applying the algorithm.



**Figure 3.** The original image (a) and its unsigned byte mask with detected edges depending on the standard deviation: (b) – standard deviation is set to 1; (c) – standard deviation is set to 4; (d) – standard deviation is set to 6; (e) – standard deviation is set to 8; (f) – standard deviation is set to 6, but the original image has been smoothed before applying the algorithm.

---

## Extract Brain: Extract Brain Surface (BET)

This algorithm extracts the surface of the brain from a T1-weighted MRI, eliminating the areas outside of the brain. This algorithm is similar to the brain extraction tool developed at the Oxford Center for Functional Magnetic Resonance Imaging of the Brain.

### Background

The procedure used by this algorithm to extract the brain surface consists of four major steps:

- Step 1, Estimating image parameters
- Step 2, Selecting an initial closed mesh within the brain
- Step 3, Evolving a mesh in a surface tangent direction, in a surface normal direction, and in a vertex normal direction
- Step 4, Identifying the voxels inside and on the brain surface

---

### STEP 1, ESTIMATING IMAGE PARAMETERS

A number of important parameters must be calculated from histograms composed of 1024 bins. These parameters are used to estimate the initial axes of the ellipsoid. As well as affect the evolution of an initial ellipsoid to the surface of the brain. Using an ellipsoid to initial surface as opposed to a sphere significantly improves the segmentation especially around the eyes and sinus cavities.

There are three such intensity values:

- Minimum threshold:  $t_{\min} = 0.5$  background threshold
- Background threshold:  $t_{\text{back}} =$  histogram index with the greatest count
- Median threshold:  $t_{\text{med}} =$  median pixel value with the initial ellipsoid that are greater than  $t_{\text{back}}$

## STEP 2, SELECTING AN INITIAL CLOSED MESH

MIPAV uses a different approach to constructing an initial mesh (i.e., ellipsoid), which is used as the initial surface for approximating the brain surface. The final approximation to the brain surface lies near the CSF and scalp. The shape of the mesh at the top of the head is nearly the shape of the scalp.

As a way of identifying the scalp voxels that lie at the top of the head, this algorithm locates all the bright voxels using the threshold  $t_{\text{bright}}$ . Voxels in the lower half of the head also show up due to fatty tissue, much of it at the base of the neck. Empirically, the number of bright voxels near the scalp at the top of the head appear to be noticeably smaller than those voxels in the bottom half of the head. This algorithm stores only those voxels that are near the scalp at the top of the head. The voxel locations are fit with an ellipsoid by means of a least squares algorithm for estimating the coefficients of the quadratic equation that represent the ellipsoid.

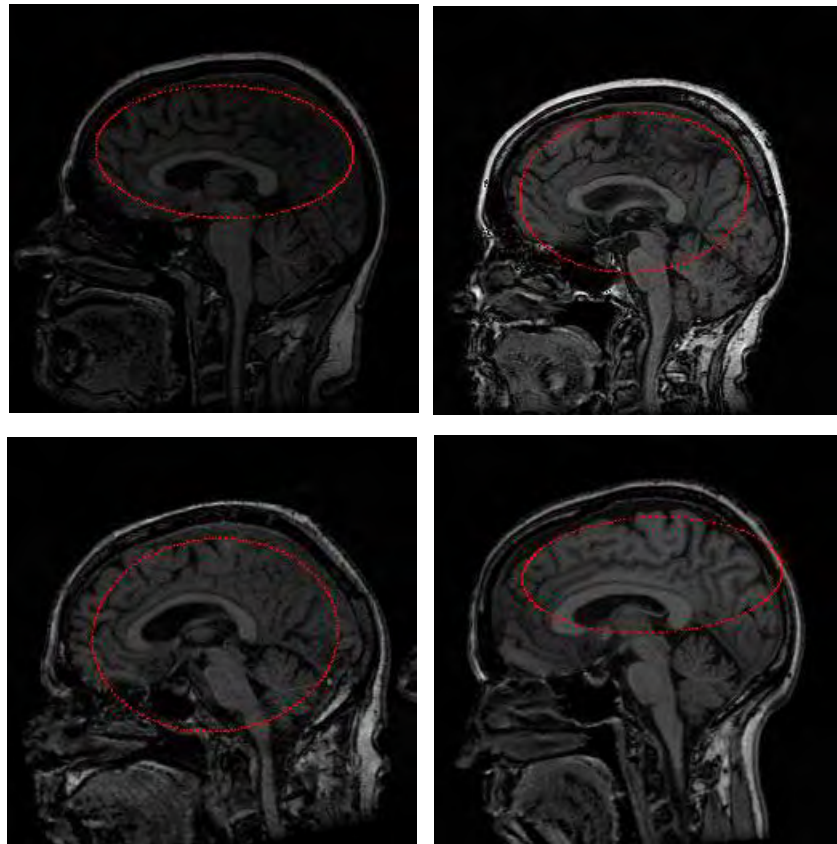
The ellipsoid obtained is reduced in scale by approximately half. The initial ellipsoids in all the test images evolved to reasonable brain surface approximations in a reasonable amount of time. Figure 1 shows the ellipsoid of intersection of the ellipsoid with the middle slice of the images.

---

**Note:** If it is unable to calculate the parameters to form an ellipsoid, MIPAV estimates a sphere instead to initialize the surface extraction process.

---





**Figure 1.** The initial ellipsoids intersected with the middle slices.

The ellipsoid is topologically equivalent to a sphere centered at the origin and with unit radius. Therefore, the initial mesh that approximates an ellipsoid is constructed by tessellating this sphere and then by applying an affine transformation to map the sphere-mesh vertices to ellipsoid-mesh vertices. The tessellation is performed by starting with an octahedron inscribed in the sphere. The octahedron vertices are  $(\pm 1, 0, 0)$ ,  $(0, \pm 1, 0)$ , and  $(0, 0, \pm 1)$ . There are initially 6 vertices, 12 edges, and 8 triangles. Each level of subdivision requires computing the midpoints of the edges and replacing each triangle by four subtriangles as shown in Figure 2.

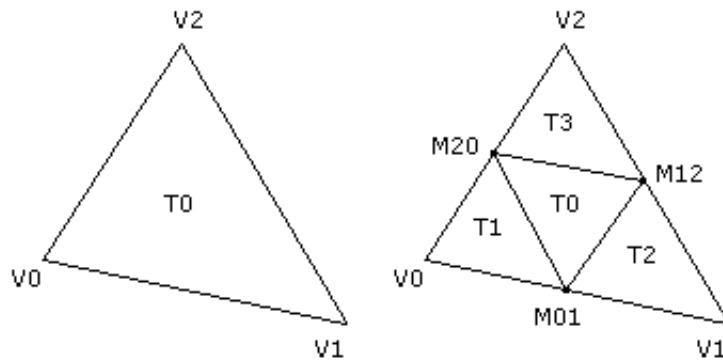


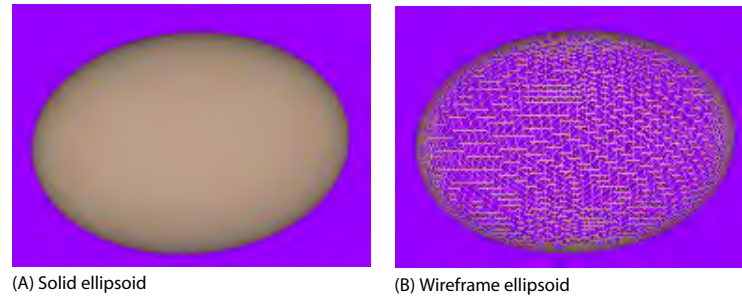
Figure 2. Subdivision of a triangle.

To avoid a lot of dynamic allocation of memory, it is possible to compute the total number of vertices, edges, and triangles that are required for the mesh given the number of subdivisions. Let  $V_0$ ,  $E_0$ , and  $T_0$  be the current quantities of vertices, edges, and triangles. Each subdivide edge leads to a new vertex, so the new number of vertices is  $V_1 = V_0 + E_0$ . Each triangle is replaced by four triangles, so the new number of triangles is  $T_1 = 4T_0$ . Each edge is split into two edges and each subdivided triangle creates three additional edges, so the new number of edges is  $E_1 = 2E_0 + 3T_0$ . These recurrence formulas are iterated over the desired number of subdivisions to compute the total objects required.

All edges are subdivided first. When each edge is subdivided, the index of the newly generated vertex must be stored for later use by triangle sharing the edge that is being subdivided. A hash map is used to store an edge as a pair of vertex indices as the key of the map. The value of the map is the index of the vertex that is generated as the midpoint. The triangles are then subdivided. The edges of the triangle are looked up in the hash map to find the indices of the three new vertices to be used by the subtriangles.

The subtriangles are stored as triples of vertex indices. After the triangles are subdivided, the original edges must all be discarded and replaced by the subdivided edges in the new mesh. These steps are repeated for all subdivision steps. Once the subdivided mesh is generated, we also need to know the vertices adjacent to a given vertex. This information is required to compute the average of the neighboring vertices. The mean length of all the

edges of the mesh can be computed using the edge hash map as storage for the edges, so after triangle division the edge hash map must exist throughout the mesh updates. Figure 3 shows a rendering of an initial ellipsoid, both solid and wireframe.



**Figure 3. An initial ellipsoid: (A) solid and (B) wireframe.**

The median intensity threshold,  $t_{med}$ , is obtained by iterating over all the voxels with the image with intensity larger than  $t_{back}$  and which are located inside the initial ellipsoid. The corresponding set of intensities is sorted and the middle value selected for the median intensity threshold.

### STEP 3, EVOLUTION OF THE MESH

The first step is to compute the average edge length  $L$  of the edges in the mesh. This is easily done by iterating over the hash map of edges, looking up the vertices from the indices stored in each edge, calculating the edge length, accumulating the lengths, then dividing by the total number of edges.

Second, the vertex normals are needed for evolving the mesh in a direction normal to the surface. This is also easily done by iterating over the triangles. A running sum of non-unit triangle normals is maintained for each vertex. Once the sums are calculated, the vertex normals are normalized by an iteration over the array storing them.

Third, the surface tangents, normals, and curvatures must be computed. If

$\vec{V}$  is a vertex, then  $\vec{V}_{\text{mean}}$  is the average of all the neighbors of  $\vec{V}$ . The neighbors are rapidly accessed by using the adjacency structure that keeps track of the array of indices of neighbors for each vertex, so the mean vertex is easily computed. The difference between vertex and mean is

$\vec{S} = \vec{V}_{\text{mean}} - \vec{V}$ . This vector is decomposed into a surface normal and surface tangential component. The normal component is in the direction of the vertex normal  $\vec{V}_N$ ,

$$\vec{S}_N = (\vec{S} \times \vec{V}_N) \vec{V}_N$$

The tangential component is

$$\vec{S}_T = \vec{S} - \vec{S}_N$$

The estimate of surface curvature is chosen to be:

$$\kappa = \frac{2|\vec{S}_N|}{L^2}$$

where  $L$  is the mean edge length of the mesh. The minimum and maximum curvatures are maintained as the various surface quantities are computed for all the vertices. These values are denoted  $\kappa_{\min}$  and  $\kappa_{\max}$  and are used to compute two parameters that are used in the update of the mesh in the surface normal direction,

$$E = \frac{1}{2(\kappa_{\min} + \kappa_{\max})}, F = \frac{6}{\kappa_{\max} - \kappa_{\min}}$$

Once all these values are computed, each vertex of the mesh is updated by

$$\vec{V}_+ = 0.5\vec{S}_T + c_1\vec{S}_N + c_2\vec{V}_N$$

It is important to note that the *BET: Brain Extraction Tool* paper (refer to “Reference” on page 176) uses the notation  $s_n$  to denote the surface normal,

usually a non-unit vector, and  $\hat{s}_n$  to denote the normalized vector for  $s_n$ .

However, Equation (13) in the *BET* paper incorrectly lists the last term in

the update equation to use  $\hat{s}_n$  when in fact it should be  $\vec{V}_N$ . Both vectors are parallel, but not necessarily pointing in the same direction. The discussion about Equation (8) in the *BET* paper makes it appear that the vector should be the vertex normal, not the surface normal. In any case, MIPAV uses  $\vec{V}_N$ .

The  $0.5\vec{S}_T$  allows the mesh to move a little bit in the tangential direction to “align” the vertices with the desired goal of being located at the mean vertices. This allows the mesh to shift about somewhat.

The  $c_1\vec{S}_N$  is a *smoothing* term. The coefficient  $c_1$  represents a stiffness of the mesh against motion in the surface normal direction. The formula for  $c_1$  in the *BET* paper is

$$c_1 = \frac{1}{2}(1 + \tanh((F)(\kappa - E)))$$

where  $\kappa$  is the estimate of surface curvature at the vertex. This formula made the surface too stiff and unable to expand to a good approximation of the brain surface. So the formula was modified to add a *stiffness parameter*,  $\sigma$ , with a default value of 0.1 and use.

$$c_1 = \frac{\sigma}{2}(1 + \tanh(F(\kappa - E)))$$

Moreover,  $\sigma$  is allowed to increase over time, then decrease later, to properly evolve the surface.

The  $c_2\vec{V}_N$  term controls the surface evolution based on the MRI data. The algorithm is almost exactly the formula described in the *BET* paper. A ray pointing inside the surface is sampled to a certain maximum depth and the sampled intensities are used to construct a minimum  $I_{\min}$  and a maximum  $I_{\max}$ . The coefficient for the update is then

$$c_2 = 0.1L\left(-b_t + \frac{I_{\min} - t_{\min}}{I_{\max} - t_{\min}}\right)$$

where

$L$  is the mean edge length

$b_t$  is what the *BET* paper calls the brain selection term (value chosen to be 1/2)

$t_{\min}$  is the minimum intensity threshold calculated during histogram analysis

Figure 4 shows a rendering of a brain surface extracted using the methods discussed in this section.

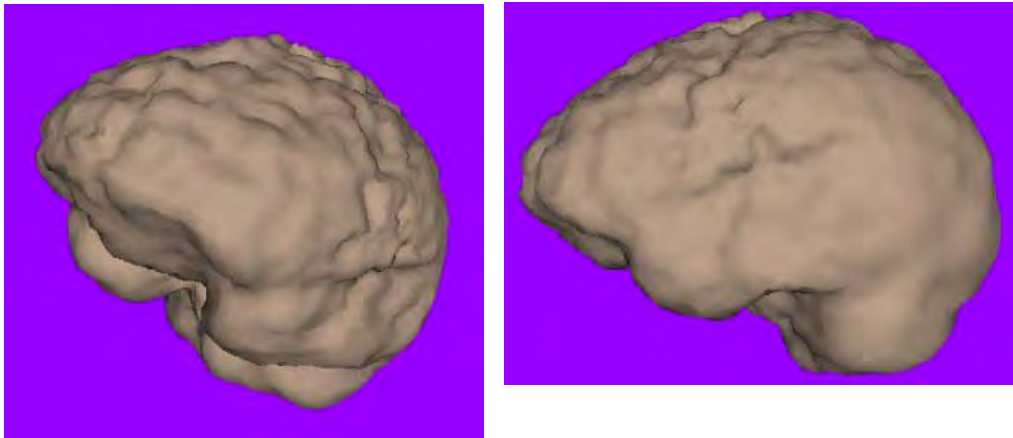


Figure 4. Two views of a brain surface mesh.

## STEP 4, SELECTING BRAIN VOXELS

This algorithm selects in two phases those voxels that are inside or on the triangle mesh that represents the surface of the brain. The first phase makes a pass over the triangles in the mesh. Each triangle is rasterized in the 3D voxel grid. The rasterization is done three times in the coordinate directions. The idea for a given direction is to cast rays in that direction and find voxels that are approximately on or near the triangle. The final result of all the rasterization is a surface that is a few voxels thick. More important is that it is a closed surface.

The end result is a ternary 3D image whose voxels are 0 for background, 1 for surface, and 2 for interior. The distinction between the surface and interior voxels allows an application to color the surface voxels in image slices to see how accurate the segmentation is. Figure 5 shows a few slices of the original MRI with the brain voxels colored.

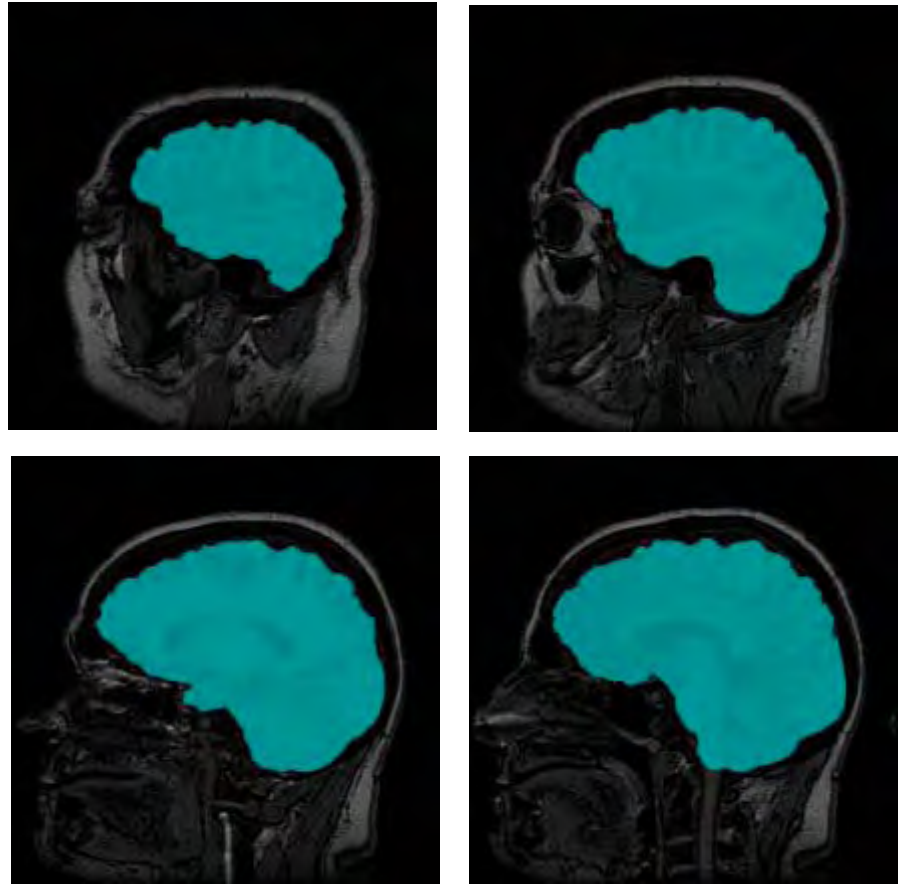


Figure 5. MRI slices with colored brain voxels.

---

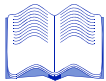
## IMAGE TYPES

You can apply this algorithm only to 3D MRI images.

---

## SPECIAL NOTES

You can save the triangle mesh in the sur format that the level surface viewer supports. This allows you to render the extracted surface. The code also gives you access to the ternary mask image to do with as you please.



---

## REFERENCE

Refer to the following reference for more information about this algorithm:

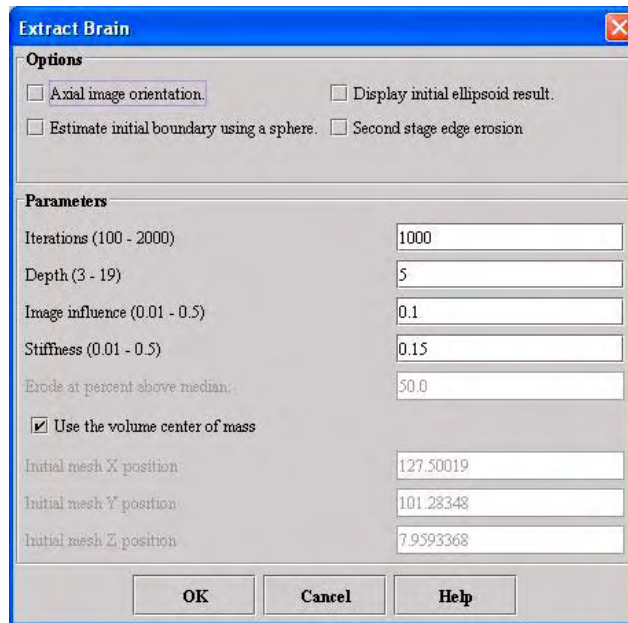
Smith, Stephen. *BET: Brain Extraction Tool*. FMRIB Technical Report TR00SMS2, Oxford Center for Functional Magnetic Resonance Imaging of the Brain, Department of Clinical Neurology, Oxford University, Oxford, England.

## Applying the Extract Brain Surface algorithm

To use this algorithm, do the following:

- 1** Select Algorithms > Extract Brain. The Extract Brain dialog box opens (Figure 6).
- 2** Complete the information in the dialog box.
- 3** Click OK. The algorithm begins to run, and a progress bar appears with the status. When the algorithm finishes running, the progress bar disappears, and the results replace the original image.





<b>Axial image orientation</b>	Specifies that the image is in the axial orientation. Unless you first change the default values of the Initial mesh position boxes, marking or clearing this check box changes their values.
<b>Estimate initial boundary using a sphere</b>	Uses a sphere shape instead of an ellipse for estimating the initial boundary. Unless you first change the default values of the Initial mesh position boxes, marking or clearing this check box changes their values.
<b>Display initial ellipsoid result</b>	Displays the ellipsoid sphere to be used in extracting the brain. This option allows you to verify that the initial ellipsoid sphere correctly covers the area in the image where the brain is located.
<b>Second stage edge erosion</b>	Performs erosion process at the edge of the brain to remove nonbrain voxels. When you select this check box, the Erode at percent above median text box is enabled.
<b>Iterations</b>	Specifies the number of times to run the algorithm when evolving the ellipsoid into the brain surface. The default is 1000.
<b>Depth</b>	Specifies the maximum depth inside the surface of the brain to use in sampling intensities. The default value is 5.
<b>Image influence</b>	Controls the surface evolution by sampling to the specified depth to calculate the maximum and minimum intensities. The default value is 0.10.

Figure 6. Extract Brain dialog box

<b>Stiffness</b>	Controls the stiffness of the mesh against motion in the surface normal direction. The default value is 0.15.
<b>Erode at percent above median</b>	Removes voxels (nonbrain material) that are at the edge of the segmentation and that have an intensity above the percent of the median voxel intensity of the brain. Type the appropriate percentage in this text box.  This text box is enabled when you select Second stage edge erosion.
<b>Use the volume center of mass</b>	Constructs the mesh using the volume center of mass. When you select this check box, you <i>cannot</i> specify the Initial <i>X</i> , <i>Y</i> , or <i>Z</i> mesh positions.
<b>Initial mesh <i>X</i> position</b>	Constructs the mesh using the <i>X</i> position that you specify. To specify a value for this text box, clear Use the volume center of mass.  Unless you first change the default value of this box, marking or clearing either the Axial image orientation or Estimate initial boundary using a sphere check boxes changes the default value.
<b>Initial mesh <i>Y</i> position</b>	Constructs the mesh using the <i>Y</i> position that you specify. To specify a value for this text box, clear Use the volume center of mass.  Unless you first change the default value of this box, marking or clearing either the Axial image orientation or Estimate initial boundary using a sphere check boxes changes the default value.
<b>Initial mesh <i>Z</i> position</b>	Constructs the mesh using the <i>Z</i> position that you specify. To specify a value for this text box, clear Use the volume center of mass.  Unless you first change the default value of this box, marking or clearing either the Axial image orientation or Estimate initial boundary using a sphere check boxes changes the default value.
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes this dialog box.
<b>Help</b>	Displays online help for this dialog box.

Figure 6. Extract Brain dialog box (continued)

---

## Extract Brain: Extract Brain Surface (BSE)

This algorithm strips areas outside the brain from a T1-weighted magnetic resonance image (MRI). It is based on the Brain Surface Extraction (BSE) algorithms developed at the Signal and Image Processing Institute at the University of Southern California by David W. Shattuck. This is MIPAV's interpretation of the BSE process and may produce slightly different results compared to other BSE implementations.

### Background

This algorithm works to isolate the brain from the rest of a T1-weighted MRI using a series of image manipulations. Essentially, it relies on the fact that the brain is the largest area surrounded by a strong edge within an MRI of a patient's head. There are essentially four phases to the BSE algorithm:

- Step 1, Filtering the image to remove irregularities, refer to page 179;
- Step 2, Detecting edges in the image, refer to page 179;
- Step 3, Performing morphological erosions and brain isolation, see page 180;
- Step 4, Performing surface cleanup and image masking, refer to page 180.

---

### STEP 1, FILTERING THE IMAGE TO REMOVE IRREGULARITIES

The first step that MIPAV performs is to filter the original image to remove irregularities, thus making the next step—edge detection—easier. The filter chosen for this was the Filters (Spatial): Regularized Isotropic (Nonlinear) Diffusion. Figure 1-A shows the original image, and Figure 1-B shows the image after it is filtered.

---

### STEP 2, DETECTING EDGES IN THE IMAGE

Next, MIPAV performs a thresholded zero-crossing detection of the filtered image's laplacian. This process marks positive areas of the laplacian image

as objects by setting them to 1 and identifies nonobject areas by setting their values to 0 (Figure 1-C).

---

### **STEP 3, PERFORMING MORPHOLOGICAL EROSIONS AND BRAIN ISOLATION**

During this step, the software performs a number of 3D (or, optionally, 2.5D) morphological erosions on the edge image mask to remove small areas identified as objects that are not a part of the brain. It then performs a search for the largest 3D region within the image, which should be the brain (Figure 1-D). It erases everything outside this region and then performs another morphological operation, dilating the brain image back to approximately its original size and shape before the erosion (Figure 1E).

---

### **STEP 4, PERFORMING SURFACE CLEANUP AND IMAGE MASKING**

Once MIPAV isolates the brain, it needs to clean up the segmentation a bit by performing more morphological operations. It first performs a 2.5D closing with a circular kernel in an attempt to fill in interior gaps and holes that may be present. Since it is better to have too much of the original volume in the extracted brain than to miss some of the brain, MIPAV performs an extra dilation during the closing operation, making the mask image slightly larger. If a smaller mask is desired, the closing kernel size can be reduced (keep in mind that this size is in millimeters and is the diameter of the kernel, not its radius).

As an option, MIPAV can then fill in any holes that still exist within the brain mask. Finally, it uses the mask to extract the brain image data from the original volume (Figure 1F).

---

### **SELECTING PARAMETERS**

Careful parameter selection must be done for the BSE algorithm to produce good results. For example, excessive erosion or dilation, closing kernel size,

or edge detection kernel size can remove detail from the brain surface or remove it completely.

### Edge detection kernel size parameter

The edge detection kernel size parameter is especially sensitive. Small changes to it (e.g., from the default of 0.6 up to 0.7 or down to 0.5 in the following example) can result in large changes to the extracted brain volume. Refer to Figure 1.



---

**Recommendation:** To find an optimal set of parameters values, run this algorithm repeatedly on a representative volume of the MRI images that you want to process with different parameter values and with Show intermediate images selected.

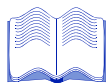
---

Figure 1 shows images that were produced from running this algorithm with the default parameters against a 256 x 256 x 47 MRI. In each image, the middle slice is shown.

---

## IMAGE TYPES

You can apply this algorithm only to 3D MRI images. The resulting image is of the same data type as the original image.



---

## REFERENCES

Refer to the following references for more information about the Brain Surface Extraction algorithm and general background information on brain extraction algorithms.

A. I. Scher, E. S. C. Korf, S. W. Hartley, L. J. Launer. "An Epidemiologic Approach to Automatic Post-Processing of Brain MRI."

David W. Shattuck, Stephanie R. Sandor-Leahy, Kirt A. Schaper, David A. Rottenburg, Richard M. Leahy. "Magnetic Resonance Image Tissue Classification Using a Partial Volume Model." *NeuroImage* 2001; 13(5):856-876.

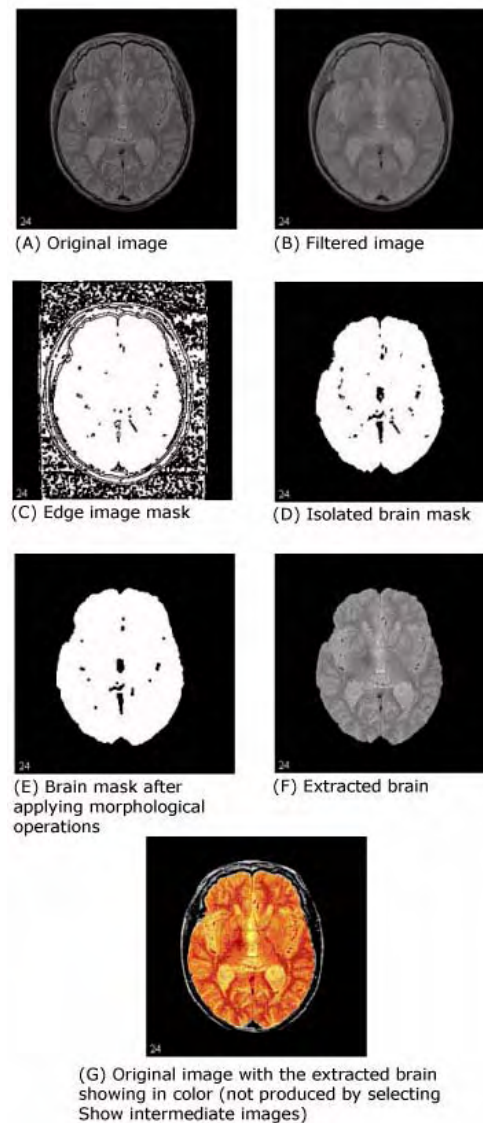


Figure 1. Examples of Extract Brain Surface (BSE) image processing

## Applying the Extract Brain Surface (BSE) algorithm

To use this algorithm, do the following:

- 1** Select Algorithms > Extract Brain Surface (BSE). The Extract Brain Surface (BSE) dialog box opens (Figure 2 on page 183).
- 2** Complete the information in the dialog box.

- Click OK. The algorithm begins to run, and a progress bar appears with the status. When the algorithm finishes running, the progress bar disappears, and the results replace the original image.

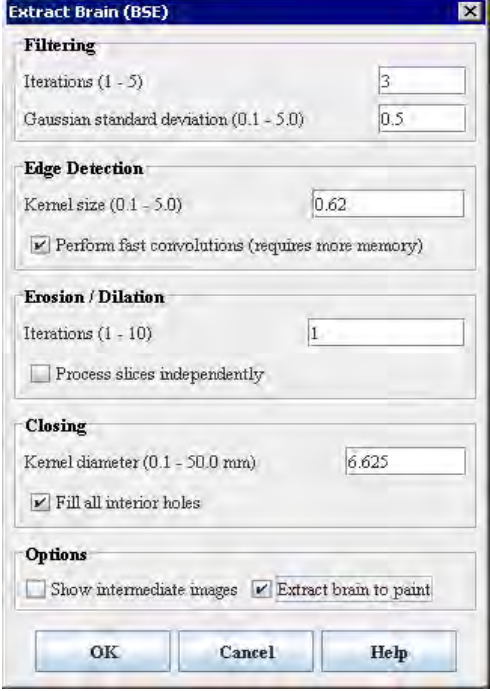
<b>Filtering</b>		
<b>Iterations (1-5)</b>	Specifies the number of regularized isotropic (nonlinear) diffusion filter passes to apply to the image. This parameter is used to find anatomical boundaries separating the brain from the skull and tissues. For images with a lot of noise, increasing this parameter will smooth noisy regions while maintaining image boundaries.	
<b>Gaussian standard deviation (0.1-5.0)</b>	Specifies the standard deviation of the Gaussian filter used to regularize the image. A higher standard deviation gives preference to high-contrast edges for each voxel in the region.	
<b>Edge Detection</b>		
<b>Kernel size (0.1-5.0)</b>	Specifies the size of the symmetric Gaussian kernel to use in the Laplacian Edge Detection algorithm. An increase in kernel size will yield an image which contains only the strongest edges. Equivalent to using a narrow filter on the image, a small kernel size will result in more edges.	
<b>Perform fast convolutions (requires more memory)</b>	Specifies whether to perform Marr-Hildreth edge detection with separable image convolutions.  The separable image convolution completes about twice as fast, but it requires approximately three times more memory. If memory is not a constraint, select this check box.	

Figure 2. Extract Brain Surface (BSE) algorithm dialog box

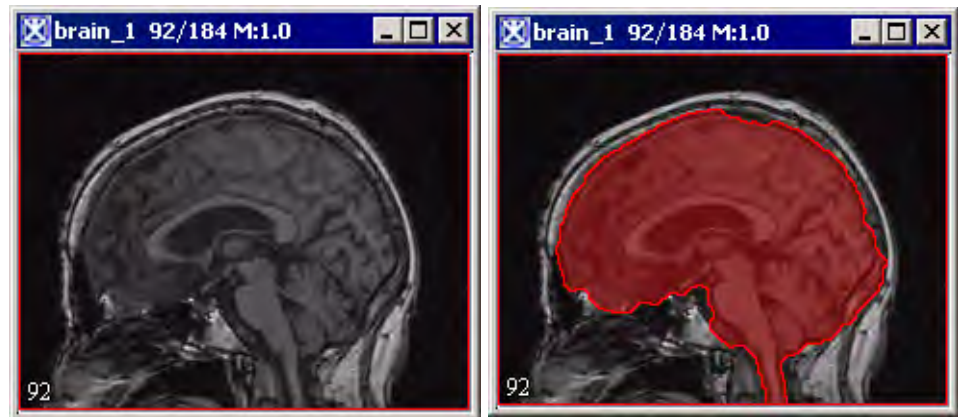
Erosion/Dilation	
<b>Iterations (1-10)</b>	<p>Specifies the number of:</p> <ul style="list-style-type: none"> <li>Erosions that should be applied to the edge image before the brain is isolated from the rest of the volume;</li> <li>Dilations to perform afterward.</li> </ul> <p>A higher number of iterations will help distinguish brain tissue from blood vessels and the inner cortical surface. Noise resulting from blood vessels or low image contrast may be present when few iterations are used.</p>
<b>Process slices independently</b>	<p>Applies the algorithm to each slice of the dataset independently. Separable image operations will again produce results more quickly while using increased memory. Since this part of the brain surface extraction is meant to fill large pits and close holes in the surface, indepent processing may not yield optimal results.</p>
Closing	
<b>Kernel diameter (in mm) (0.1-50.0)</b>	<p>Specifies the size of the kernel to use (in millimeters). The value defaults to a number of millimeters that ensures that the kernel is 6 pixels in diameter and takes into account the volume resolutions. Closing operations act to fill smaller pits and close holes in the segmented brain tissue.</p>
<b>Fill all interior holes</b>	<p>Fills in any holes that still exist within the brain mask. When optimal parameters for a given image have been used, this option will generally produce a volume of interest that lies between the inner cortical surface and the outer cortical boundary.</p>
Options	
<b>Show intermediate images</b>	<p>Shows, when selected, in addition to the final brain image, the images that are generated at various points while the BSE algorithm is running. Selecting this check box may help you in finding the optimal parameters for running the BSE algorithm on a volume.</p> <p>For an image named <i>ImageName</i>, the debugging images displayed would include:</p> <ul style="list-style-type: none"> <li>The filtered image (named <i>ImageName_filter</i>)</li> <li>The edge image (named <i>ImageName_edge</i>)</li> <li>The eroded edge image (named <i>ImageName_erode_brain</i>)</li> <li>The isolated brain mask after erosion and dilation (named <i>ImageName_erode_brain_dilate</i>)</li> <li>The brain mask after closing (named <i>ImageName_close</i>)</li> <li>The closing image is shown before any interior mask holes are filled</li> </ul>
<b>Extract brain to paint</b>	<p>Paints the extracted brain onto the current image. See also Figure 3.</p>

Figure 2. Extract Brain Surface (BSE) algorithm dialog box (continued)



<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes this dialog box.
<b>Help</b>	Displays online help for this dialog box.

**Figure 2. Extract Brain Surface (BSE) algorithm dialog box (continued)**



**Figure 3. The Extract Brain to Paint option: on your left is the original image and on your right is the result image with the brain extracted to paint.**

---

## Face Anonymizer (BET)

This Face Anonymizer (BET) algorithm extracts an approximated face from a T1-weighted MRI, eliminating facial areas outside of a certain buffer from the brain. The algorithm employs several other algorithms developed by the MIPAV project, refer to page 188 for applicable references.

### Background

The procedure used by this algorithm to return an image with an extracted face consists of five major steps:

- 1** Extract the brain, refer to page 186;
- 2** Buffer the brain, refer to page 187;
- 3** Define the face, refer to page 187;
- 4** Extract the face, refer to page 187;
- 5** Smooth the face, refer to page 187.

---

### STEP 1, EXTRACT THE BRAIN

The brain extraction portion of the algorithm employs the Extract Brain Surface (BET) tool. This tool extracts the surface of the brain from a T1-weighted MRI by expanding an initial ellipsoid. The brain extraction tool is derived from a similar tool developed at the Oxford Center for Functional Magnetic Resonance Imaging of the Brain.

Parameters that directly relate to the mechanics of the brain extraction tool can be changed using the dialog box for the face anonymizer. For more information, refer to the dialog help section on page 188.

For more information concerning the mechanics of this portion of the algorithm, see references on page 188.

---

## **STEP 2: BUFFER THE BRAIN**

The brain extracted from the brain extraction tool is buffered by a certain number of millimeters that can be set using this algorithm's dialog menu. This portion of the algorithm employs the morphological dilate algorithm to extend the brain mask derived from brain extraction tool in three dimensions. Refer to “Extract Brain: Extract Brain Surface (BSE)” for more information.

---

## **STEP 3: DEFINE THE FACE**

The face is searched for in an area defined by the MRI given orientation. This orientation is derived from the file information of the MRI and can be changed using this algorithm's dialog options. Within this region an initial shape is chosen that will expand in an effort to extract the face.

---

## **STEP 4: EXTRACT THE FACE**

Face extraction is achieved by appending spheres of random radius onto the initial shape. This size of the spheres decays exponentially as the area of the two octants is filled. Eventually the process converges to a shape that is defined as an approximate face and does not lie within the buffered brain area. For more information on the method of sphere approximation, see references on page 188.

---

## **STEP 5: SMOOTH THE FACE**

The extracted face is smoothed to approximate a quarter-cylinder within the two given octants. The final extracted face is guaranteed to not lie within the extracted brain.

---

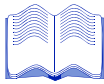
## IMAGE TYPES

You can apply this algorithm only to 3D T1-weighted MRI images. The image file must represent the whole brain (from the left to the right side and from the top to the bottom of the brain). The algorithm would not work if the image file contains only a few slices.

---

## NOTES

The resulting image is of the same data type as the original image.



## REFERENCE:

### Within the MIPAV project:

The brain extraction tool, which extracts the surface of the brain from a T1-weighted MRI and the dilate algorithm, which buffers the extracted brain by a set number of millimeters, are described here: “Extract Brain: Extract Brain Surface (BSE)”.

### Outside of MIPAV:

Smith, Stephen. BET: Brain Extraction Tool. FMRIB Technical Report TR00SMS2, Oxford Center for Functional Magnetic Resonance Imaging of the Brain, Department of Clinical Neurology, Oxford University, Oxford, England. Smith's paper is also available here: <http://portal.acm.org/citation.cfm?id=246>

McIlroy, M. D. Best approximate circles on integer grids. ACM Transactions on Graphics (TOG), Volume 2 Issue 4. An approximation for spheres also available at <http://portal.acm.org/citation.cfm?id=246>.

## Applying the Face Anonymizer Algorithm

To use this algorithm, do the following:

- 1 Select Algorithms > Brain Tools > Face de-Identification. The Face Anonymizer dialog box appears.
- 2 Complete the information in the dialog box. Check that the MRI orientation information is correct.
- 3 Click OK.

- The algorithm begins to run, and a progress bar appears with the status. When the algorithm finishes running, the progress bar disappears, and the original image without the extracted face appears.

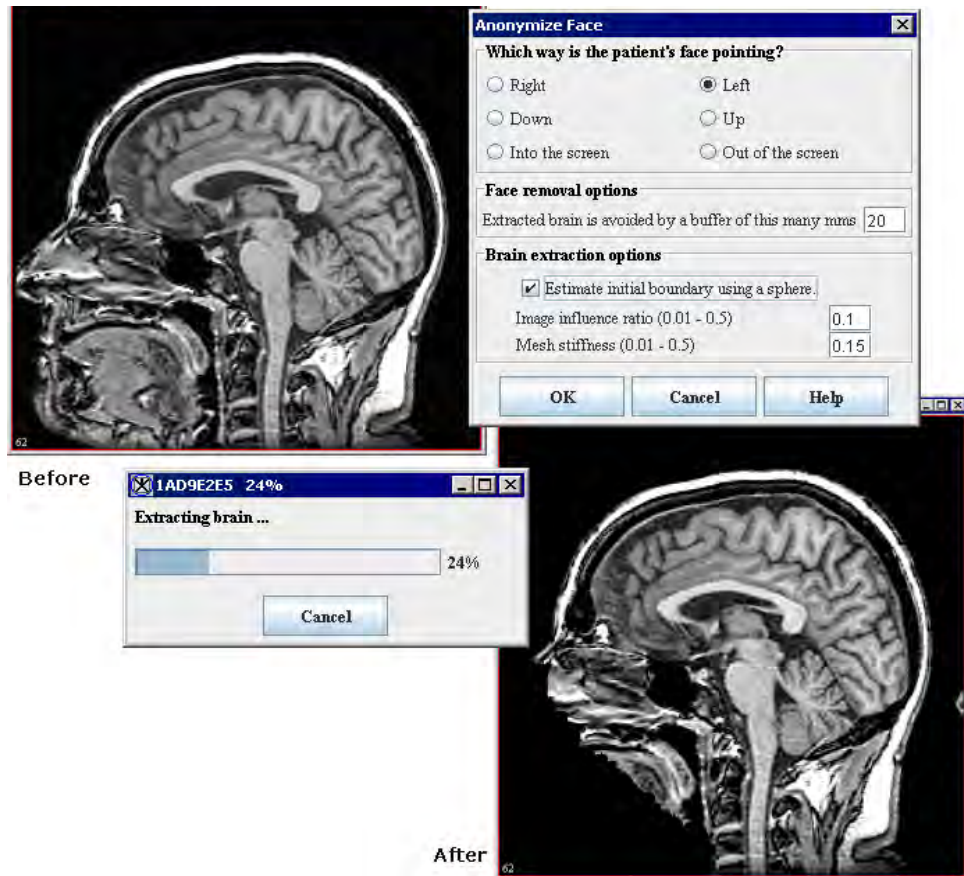
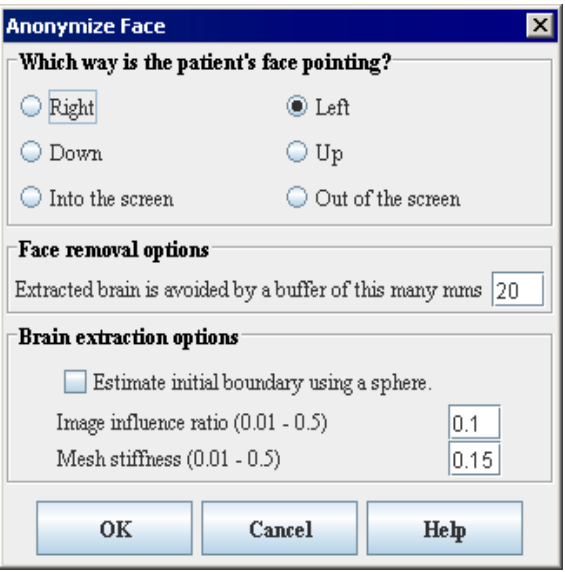


Figure 1. An example of the Face Anonymizer (BET) image processing. The image has been processed using default parameters.



**Recommendation:** To find an optimal set of parameter values, run this algorithm repeatedly on a representative volume of the 3D T1-weighted MRI images using different parameter values. Keep the records of parameter values you use.

<p><b>Which way is the patient's face pointing?</b></p>	<p>Specifies the orientation of the image. MIPAV extrapolates this value from the image file information. Confirm that the orientation given is correct:</p> <p><b>Right</b> – sagittal image with x-axis oriented posterior to anterior;</p> <p><b>Left</b> – sagittal image with X-axis oriented anterior to posterior;</p> <p><b>Down</b> – axial image with Y-axis oriented posterior to anterior;</p> <p><b>Up</b> – axial image with y-axis oriented anterior to posterior;</p> <p><b>Into the screen</b> – coronal image with Z-axis oriented posterior to anterior;</p> <p><b>Out of the screen</b> – coronal image with Z-axis oriented anterior to posterior.</p>	
<b>Face Removal Options</b>		
<p><b>Brain buffer</b></p>	<p>Guarantees that the extracted brain will be avoided by the specified number of millimeters. Refer to “Step 2: Buffer the brain” for more information.</p>	
<b>Brain Extraction Options</b>		
<p><b>Initial shape</b></p>	<p>Uses a sphere shape instead of an ellipse for estimating the initial boundary. Specifies whether a sphere should be used for initial brain extraction. For more information, see the sphere approximation option for the brain extraction tool.</p>	
<p><b>Image influence ratio</b></p>	<p>Specifies the depth to calculate in approximating brain maximum and minimum intensities. For more information, refer to “Extract Brain: Extract Brain Surface (BSE)” to learn more about the image influence option for the brain extraction tool. Default value is 0.1. Allowed values range from 0.01 to 0.5.</p>	
<p><b>Mesh stiffness</b></p>	<p>Controls the stiffness of the brain approximation mesh. For more information about the stiffness option, refer to Chapter “Extract Brain: Extract Brain Surface (BSE)” of the MIPAV manual. Default value is 0.15. Allowed values range from 0.01 to 0.5.</p>	
<p>OK</p>	<p>Confirms the dialog settings and runs the algorithm.</p>	
<p>Cancel</p>	<p>Disregards any changes that you made in this dialog box and closes this dialog box.</p>	
<p>Help</p>	<p>Displays online help for this dialog box.</p>	

**Figure 2. Face Anonymizer dialog box options**

## Fast Fourier Transformation (FFT)

There are two main approaches to filtering an image. The first is the convolution of an image and kernel in the spatial domain. The second is the multiplication of an image's fourier transform with a filter in the frequency domain. The fast fourier transformation (FFT) algorithm, which is an example of the second approach, is used to obtain a frequency-filtered version of an image. Processing images by filtering in the frequency domain is a three-step process:

- 1** Perform a forward fast fourier transform to convert a spatial image to its complex fourier transform image. The complex image shows the magnitude frequency components. It does not display phase information.
- 2** Enhance selected frequency components of the image and attenuate other frequency components of the image by multiplying by a low-pass, high-pass, band-pass, or band-stop filter.

In MIPAV, you can construct frequency filters using one of three methods: finite impulse response filters constructed with Hamming windows, Gaussian filters, and Butterworth filters. However, for the Gaussian filters, only low-pass and high-pass filters are available.

- 3** Perform an inverse fast fourier transform to reconvert the image from the frequency domain to the spatial domain.

You can also use Algorithms > Filters (frequency) to obtain a frequency-filtered version of an image in a single step. A contrast of the FFT and Filter (frequency) commands on the Algorithms menu is helpful in determining which one to select.

### Use . . .

Algorithms > Filters  
(frequency)

Algorithms > FFT

### To . . .

Obtain a frequency-filtered version of an image in a single step. All three steps in the process are performed internally and only the resultant filtered image is displayed.

Produce pictures of the fourier transform of the image and the filtered fourier transform of the image.

## Background

Since all dimensions of an  $n$ -dimensional dataset must be powers of 2, datasets with arbitrary dimensions are zero padded to powers of 2 before applying the Fast Fourier Transform and stripped down to the original dimensions after applying the Inverse Fast Fourier Transform.

If real source data is used, then the output obtained after the final Inverse Fast Fourier Transform step is purely real, no matter what filter construction method is used. Any non-zero imaginary part present after the Inverse Fast Fourier Transform is due to Randolph error. Hence, the imaginary part is discarded and only the real part is imported into the final image. Figure 1 shows the application of the FFT algorithm to a 2D image.

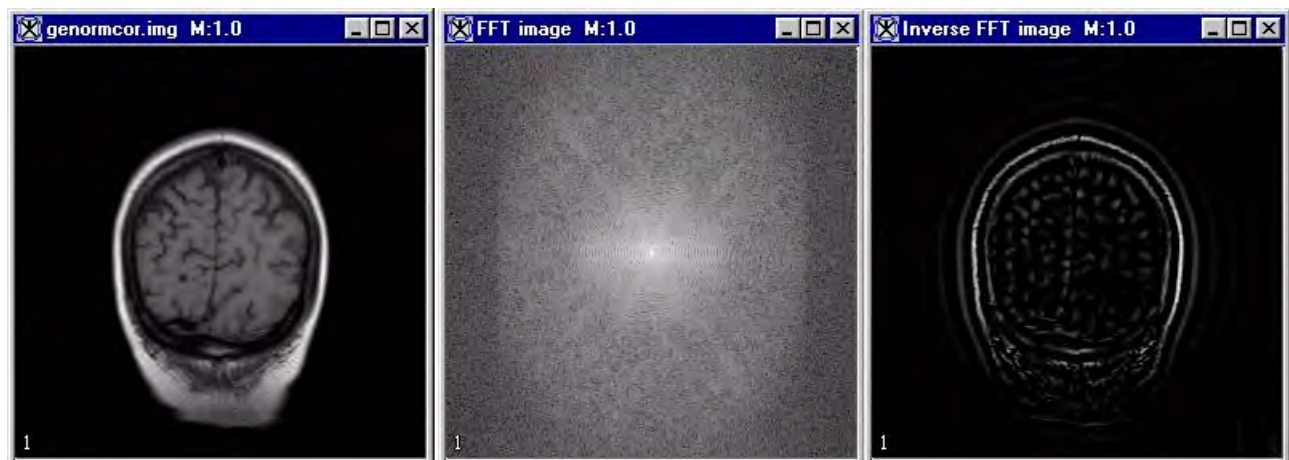


Figure 1. FFT algorithm processing

In MIPAV, frequency filters may be constructed using one of three methods: finite impulse response filters constructed with Hamming windows, Gaussian filters, and Butterworth filters. You select the method when you enter the parameters for the algorithm.



## FINITE IMPULSE RESPONSE (FIR) FILTERS WITH HAMMING WINDOWS

If you use FIR filters with Hamming windows, an ideal filter kernel is constructed in the spatial domain. A Hamming window kernel is also constructed in the spatial domain. Then, the ideal filter kernel and the Hamming window kernel are multiplied together in the spatial domain. The resultant kernel is a circle or odd diameter sphere with its center at (kernel diameter - 1)/2 in every dimension. The kernel is zero padded up to the same dimensions as the original image data.

The two fourier transforms (image and filter) are multiplied, and the inverse fourier transform is obtained. The result is a filtered version of the original image shifted by (kernel diameter - 1)/2 toward the end of each dimension. The data is shifted back by (kernel diameter - 1)/2 to the start of each dimension before the image is stripped to the original dimensions.

In every dimension the length of the resulting fourier image must be greater than or equal to the length of non-zero image data + kernel diameter - 1 so that aliasing does not occur. In aliasing, higher frequency components incorrectly shift to lower frequencies. Thus, if the image is not cropped, each frequency space dimension must be greater than or equal to the minimum power of two that equals or exceeds the original dimension size + kernel diameter - 1. If cropping is selected, the fourier image is only increased to the minimum power of 2 that equals or exceeds the original dimension size rather than the minimum power 2 that equals or exceeds the original dimension size + kernel diameter - 1. With cropping, a band of pixels at the boundaries of the image is set equal to 0. A power of 2 increase in dimension length is avoided at the cost of losing edge pixels.

The following formula defines the discrete-space fourier transform pair:

$$X(\omega_1, \omega_2) = \sum_{n_1 = -\infty}^{\infty} \sum_{n_2 = -\infty}^{\infty} x(n_1, n_2) e^{-j\omega_1 n_1} e^{-j\omega_2 n_2}$$

$$x(n_1, n_2) = \frac{1}{(2\pi)^2} \int_{\omega_1 = -\pi}^{\pi} \int_{\omega_2 = -\pi}^{\pi} X(\omega_1, \omega_2) e^{-j\omega_1 n_1} e^{-j\omega_2 n_2} d\omega_1 d\omega_2$$

While centering is invoked in FFT to provide a clearer display, centering is not used in frequency filter where no fourier images are produced.

## GAUSSIAN AND BUTTERWORTH FILTERS

In the frequency filtering step, both the real and imaginary parts of the fourier transform are scaled by the same formula, which depends only on the distance from the center of image.

The FIR filter must create two fourier transforms: one for the original image and one for the kernel. The Butterworth and Gaussian filters only need to create one fourier transform for the image since frequency scaling is done with a formula. Therefore, the FIR filter uses about twice as much memory as the Gaussian or Butterworth filters.

Centering is needed for the Gaussian and Butterworth filters since the scaling formulas are based on frequency magnitude increasing from the center of the image.

### Butterworth 2D and 3D filters

The following are the formulas for the Butterworth 2D and 3D filters, respectively.

$$distsq = (x - xcenter)^2 / (xcenter)^2 + (y - ycenter)^2 / (ycenter)^2$$

$$distsq = (x - xcenter)^2 / (xcenter)^2 + (y - ycenter)^2 / (ycenter)^2 + (z - zcenter)^2 / (zcenter)^2$$

The real data is the product of the real data and the coefficient. The imaginary data is the product of the imaginary data and the coefficient.

### Butterworth low-pass, high-pass, band-pass, and band-stop filters

The following are the equations for the Butterworth low-pass, high-pass, band-pass, and band-stop filters.

### Butterworth low-pass filter

$$\text{coefficient} = \frac{1.0}{1.0 + \left(\frac{\text{distsq}}{f1}\right)^{2 \cdot \text{butterworthorder}}}$$

### High-pass filter

$$\text{coefficient} = \frac{1.0}{1.0 + \left(\frac{f1}{\text{distsq}}\right)^{2 \cdot \text{butterworthorder}}}$$

### Band-pass filter

$$\text{width} = f2 - f1, \text{centersq} = f1 \cdot f2$$

$$\text{num} = (\text{distsq} - \text{centersq})^{2.0 \cdot \text{butterworthorder}}$$

$$\text{coeff} = \frac{\text{num}}{(\text{num} + (\text{distsq} - \text{centersq})^{2 \cdot \text{butterworthorder}})}$$

### Band-stop filter

$$\text{width} = f2 - f1, \text{centersq} = f1 \cdot f2$$

$$\text{num} = (\sqrt{\text{distsq}} \cdot \text{width})^{2 \cdot \text{butterworthorder}}$$

$$\text{coeff} = \frac{\text{num}}{(\text{num} + (\sqrt{\text{distsq}} \cdot \text{width})^{2 \cdot \text{butterworthorder}})}$$

## Gaussian low- and high-pass filters

The following equations are for the Gaussian low- and high-pass filters.

*Gaussian low-pass (2D) filter*

$$\text{coeffientforgaussianhighpass} = 1 - \text{gaussianlowpass}$$

*Gaussian low-pass (3D) filter*

$$\text{coefficient} = e^{-\frac{(x - xcenter)^2}{(2f^2 xcenter^2)}} \cdot e^{-\frac{(y - ycenter)^2}{(2f1^2 ycenter^2)}} \cdot e^{-\frac{(z - zcenter)^2}{(2f1^2 zcenter^2)}}$$



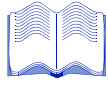
**Note:** The routine `bessj1` for determining a bessel function of the first kind and first order is taken from "Numerical Recipes in C."

## IMAGE TYPES

You cannot apply a forward FFT to complex data. However, you can apply the frequency filter or perform an inverse FFT on conjugate, symmetric, and complex data. You cannot apply this algorithm to Alpha Red Green Blue (ARGB) datasets, but you can apply it to other gray-scale 2D and 3D images.

## SPECIAL NOTES

By default, the resultant image is a float-type image. Intermediate fourier transform images and filtered fourier transform images obtained in FFT are complex images.



---

## REFERENCES

Refer to the following references for more information about this algorithm:

Nasir. Ahmed, *Discrete-Time Signals and Systems* (Reston, Virginia: Reston Publication Company, 1983), pp. 243–258.

Raphael C. Gonzalez and Richard E. Woods. *Digital Image Processing* (Boston: Addison-Wesley, 1992), pp. 201–213, 244.

Jae S. Lim and Joe S. Lim, *Two-Dimensional Signal and Image Processing* (Upper Saddle River, New Jersey: Prentice-Hall PTR, 1990), pp. 57–58, 195–213.

William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling, "Numerical Recipes in C," *The Art of Scientific Computing*, Second Edition (Cambridge, Massachusetts: Cambridge University Press, p. 1993), pp. 233, 521–525.

## Applying the FFT algorithm

Generally, when an image is filtered in the frequency domain, you use the following three-step process:

- 1** Apply Forward FFT. In this first step, the spatial image is converted to a complex fourier transform image.
- 2** Apply Frequency Filter: Next, you can enhance and attenuate selected frequency components by multiplying the fourier transform image by the low-pass, high-pass, band-pass, or band-stop filters.
- 3** Apply Inverse FFT: Finally, you can perform an inverse fast fourier transform to reconvert the image to the spatial domain.

You can use the FFT algorithm to perform one or all of these steps.

---

## STEP 1: CONVERTING DATASET FROM SPATIAL TO FOURIER DOMAIN USING THE FORWARD FFT COMMAND

To convert the dataset from the spatial to fourier domain, complete the following steps:

- 1** In the Default Image window, open a dataset that is displayed in the spatial domain.
- 2** In the MIPAV window, select Algorithm > FFT. The FFT dialog box opens (Figure 2).
- 3** Select the destination.
- 4** By default, the forward FFT is selected. (The frequency filter and inverse FFT methods cannot be applied to real spatial datasets.)
- 5** Select the filter construction method. If you select the Butterworth filter, enter the order.
  - **FIR:** Click Windowed finite impulse response. Enter the kernel diameter, which must be an odd number that is greater than or equal to 3. To save memory at the expense of zeroing edge pixels, select or clear image cropping.
  - **Gaussian:** Click Gaussian filter.
  - **Butterworth:** Click Butterworth filter, and then enter the order of the Butterworth filter.
- 6** Having selected the filter construction method, you can choose to apply the following options:
  - If you select Display log magnitude, the  $\log_{10} [1 + \text{square root}(\text{real part}^2 + \text{imaginary part}^2)]$  is shown. If the Display log magnitude is clear, square root ( $\text{square root}(\text{real part}^2 + \text{imaginary part}^2)$ ) is shown.
  - If your image has unequal dimensions, you can select Allow unequal FFT dimensions to allow unequal FFT dimensions. If you want the FFT to equalize the dimensions to produce a symmetrical image, clear the check box.
  - If you selected the FIR filter, you can crop the image to save

memory. Enter an odd number for the convolution kernel diameter. The number must be odd so there is symmetry around a center point.

- 7 Click OK to perform the forward FFT. MIPAV applies the FFT algorithm. When complete, the fourier transform of the image appears.

---

## STEP 2: ENHANCING AND ATTENUATING FREQUENCY COMPONENTS USING FREQUENCY FILTER

To apply the frequency option to enhance and attenuate frequency components, complete the following steps:

- 1 Select the fourier transform of a spatial image. The image appears in a default image window.
- 2 In the MIPAV window, select Algorithms > FFT. The FFT dialog box (Figure 2) opens.
- 3 Select the destination.
- 4 Select Frequency Filter.
- 5 Click, if desired, the check box to display the log magnitude.
- 6 Select one of the following filter specifications:
  - **Lowpass:** Type the value of the frequency. The value must be greater than 0.0, but less than or equal to 1.0. where 0.0 is the zero frequency and 1.0 is the absolute value of the frequency at the end of a centered fourier dimension.
  - **Highpass:** Enter the value of the frequency. The value must be greater than 0.0, but less than or equal to 1.0. 0.0 is the zero frequency, 1.0 is the absolute value of the frequency at the end of a centered fourier dimension.
  - **Bandpass:** Type the values of the frequencies F1 and F2, with F2 greater than F1. The value must be greater than 0.0, but less than or equal to 1.0 where 0.0 is the zero frequency and 1.0 is the absolute value of the frequency at the end of a centered fourier dimension.
  - **Bandstop:** Enter the values of the frequencies F1 and F2, with F2

greater than F1. The value must be greater than 0.0, but less than or equal to 1.0 where 0.0 is the zero frequency and 1.0 is the absolute value of the frequency at the end of a centered fourier dimension.

**7** When complete, click OK. MIPAV applies the FFT algorithm. When complete, the fourier transform of the image appears.

<b>Process each slice independently (2.5D)</b>	Filters each slice of the dataset independently of adjacent slices.
<b>Display log magnitude</b>	Displays 1 plus the magnitude of the FFT.
<b>Allow unequal FFT dimensions</b>	Allows the image to have unequal dimensions. If this option is not selected, FFT makes the dimensions equal, which may result in more memory usage during processing.
<b>Windowed finite impulse response</b>	Constructs an ideal filter kernel and a Hamming window kernel in the spatial domain and multiplies them. The resultant kernel is a circle or sphere of odd diameter with its center at (kernel diameter - 1)/2 in every dimension.
<b>Crop image to save memory</b>	Saves memory by assigning the edge pixels of the image a value of zero.
<b>Convolution kernel diameter-odd</b>	Limits the diameter of the convolution kernel to the size that you specify.
<b>Gaussian filter</b>	Applies the Gaussian filter to the image. The Gaussian filters only low-pass and high-pass filters are available.
<b>Butterworth filter</b>	Applies the Butterworth filter, which has a maximally flat response in its passband. The passband is the range of frequencies that pass with a minimum of attenuation through an electronic filter.

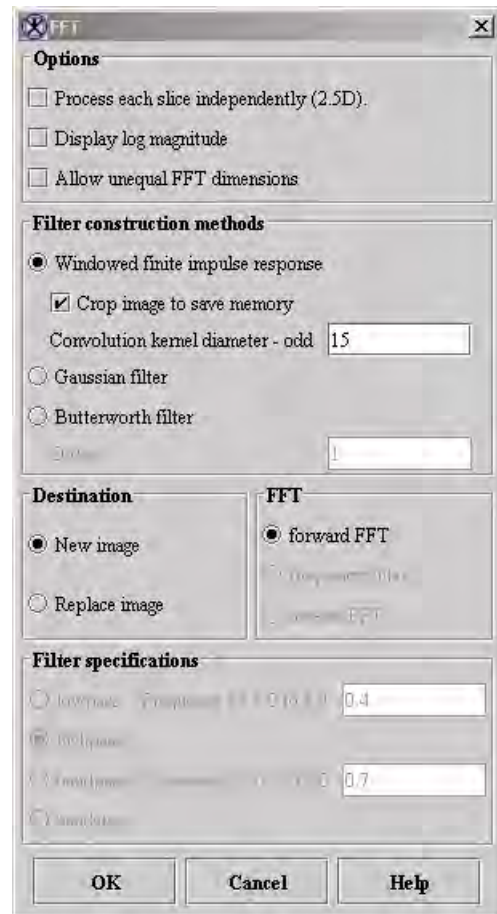


Figure 2. FFT dialog box




<b>Order</b>	Displays the order of the Butterworth filter.
<b>New image</b>	Places the result of the algorithm in a new image window.
<b>Replace image</b>	Replaces the current active image with the results of the algorithm.
<b>Forward FFT</b>	Converts spatial image to a complex fourier transform image.
<b>Frequency filter</b>	Enhances or attenuates selected frequency components by multiplying the fourier transform image by the low-pass, high-pass, band-pass, or band-stop filters.
<b>Lowpass</b>	Suppresses frequencies above a specified threshold. Frequencies below the threshold are transmitted.
<b>Highpass</b>	Suppresses frequencies below a specified threshold. Frequencies above the threshold are transmitted.
<b>Bandpass</b>	Suppresses all frequencies except those in a specified band.
<b>Bandstop</b>	Suppresses a band of frequencies from being transmitted through the filter. Frequencies above and below the band are transmitted.
<b>OK</b>	Applies the FFT algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

Figure 2. FFT dialog box (continued)

## OPTIONAL STEP: ATTENUATING FREQUENCY COMPONENTS USING THE PAINT TOOL

With the Paint tool you can attenuate frequency components to values whose magnitude is 1,000 times the minimum floating point value. The values are not simply attenuated to zero because this would result in the phase information being lost and cause phase discontinuities in the image. Frequency components whose magnitudes are less than or equal to 1,000 times the minimum floating point value are unaffected by the paint tool. To do this, complete the following steps:

- 1** Display the paint toolbar if it is not already visible. To do this, in the MIPAV window, select **Toolsbar > Paint toolbar**. The paint toolbar appears.
- 2** Select  **Paint Brush** to select the paint brush.

**3** Select the size of the paint brush tip. Choose one of the following:

- Small Tip—To select a tip that is one pixel in size.
- Medium Tip—To select a tip that is 16 pixels (4 x 4 square) in size.
- Large Tip—To select a tip that is 100 pixels (10 x 10 square) in size. As you draw, you can select one pixel at a time or drag to draw paint strokes on the image.

As you draw, you can select one pixel at a time or drag to draw paint strokes on the image.



**Note:** In painting complex images, the complex image is assumed to have been generated from real data.  $x(n_1, n_2) \text{ real} \leftrightarrow X(k_1, k_2) = X^*(-k_1, -k_2)$  for 2D and  $x(n_1, n_2, n_3) \text{ real} \leftrightarrow X(k_1, k_2, k_3) = X^*(-k_1, -k_2, -k_3)$ . Thus, real data yields a conjugate symmetric fourier transform. Therefore, when one pixel is painted in a complex image, its conjugate symmetric pixel is also painted. As you draw in 2D, the paint also appears on the opposite side of the image. As you draw in 3D, the paint also appears on the opposite side of the image on another slice. This mirroring technique ensures that, when the inverse fourier transform is applied to the frequency image, the spatial image that results is purely real.

**4** When finished, to return to normal mode, click  Normal Mode.

**5** You can erase the paint from the image's paint layer by completing the following steps. You can either erase all paint from the image or erase paint from a specific conjugate symmetric area.

To erase all paint in the image, click  Global Erase.

To erase space from a specific area, complete the following steps:

- Click Erase.
- Select the size of the eraser tip. Click  Small Tip to select a tip that is 1 pixel in size. Click  Medium Tip to select a tip that is 16 pixels (4 x 4 square) in size. Click  Large Tip to select a tip that is 100 pixels (10 x 10 square) in size.
- Use the mouse to begin erasing. You can select one pixel at a time or drag the mouse. If desired, use the magnification buttons to magnify

the image.

- When finished, to return to normal mode, click  Normal Mode.

**6** Click Commit to commit the changes (copy them permanently to an image).

---

### STEP 3: RECONVERTING THE IMAGE TO THE SPATIAL DOMAIN

To reconvert the image to the spatial domain, complete the following steps:

- 1** Select the fourier transform of a spatial image. The image appears in a default image window.
- 2** In the MIPAV window, select Algorithms > FFT. The FFT dialog box (Figure 2) opens.
- 3** Select the destination.
- 4** Select Inverse FFT.
- 5** If desired, click Display log magnitude.
- 6** Click OK. When complete processing is complete, the image appears in the spatial domain.

---

## Filters (Frequency)

Frequency filters process an image in the frequency domain. The image is first, Fourier transformed, then multiplied with the filter function, and then transformed back to the spatial domain. Attenuating high frequencies results in a smoother image in the spatial domain, attenuating low frequencies enhances the edges.

### Background

Frequency filtering is based on the Fourier Transform. The operator usually takes an image and a filter function in the Fourier domain. This image is then multiplied with the filter function in a pixel-by-pixel fashion:

EQUATION 1

$$G(k,l)=F(k,l)*H(k,l)$$

Where  $F(k,l)$  is the input image in the Fourier domain,  $H(k,l)$  is the filter function and  $G(k,l)$  is the result filtered image.

The form of the filter function determines the effects of the operator. There are basically four different kinds of filters: lowpass, highpass, bandpass and bandstop filters.

**A low-pass filter** attenuates high frequencies and retains low frequencies unchanged. The result in the spatial domain is equivalent to that of a smoothing filter; as the blocked high frequencies correspond to sharp intensity changes, i.e. to the fine-scale details and noise in the spatial domain image.

**A highpass filter** yields edge enhancement or edge detection in the spatial domain, because edges contain mostly high frequencies. In other hand, areas of rather constant gray level consist of mainly low frequencies and are, therefore, suppressed.

**A bandpass filter** attenuates very low and very high frequencies, but retains a middle range band of frequencies. Bandpass filtering is used to enhance edges (suppressing low frequencies) while reducing the noise at the same time (attenuating high frequencies).

The most simple lowpass filter is the ideal lowpass. It suppresses all frequencies higher than the cut-off frequency  $F_0$  and leaves smaller frequencies unchanged, which can be described as:

EQUATION 2

$$H(k,l)=1 \text{ if } (k^2+l^2)^{1/2} < F_0$$

and

EQUATION 3

$$H(k,l)=0 \text{ if } (k^2+l^2)^{1/2} > F_0$$

Where,  $H(k,l)$  is the filter function.

In most implementations,  $F_0$  is given as a fraction of the highest frequency represented in the Fourier domain image.

The drawback of this filter function is a ringing effect that occurs along the edges of the filtered spatial domain image. See Figure 1.

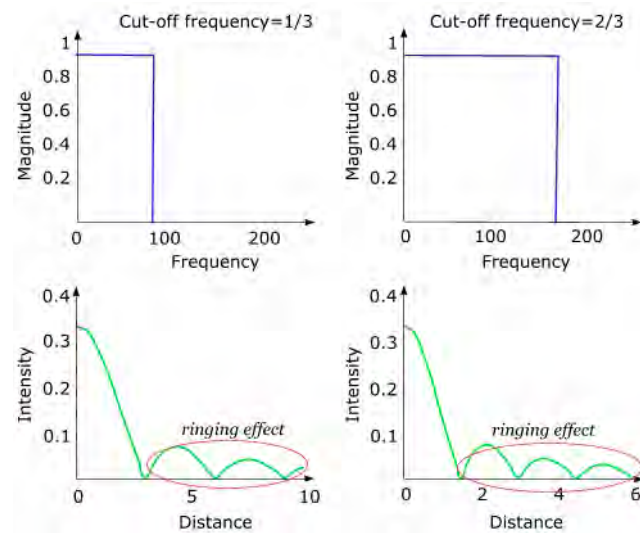


Figure 1. The ideal lowpass filter in frequency and spatial domains

Better results can be achieved with a Gaussian shaped filter function.

EQUATION 4

$$H(k,l)=\exp(-F^2(k,l)/2Fo^2)$$

Where,  $H(k,l)$  is the Gaussian lowpass filter function, and  $F_0$  is the cut-off frequency.

The advantage is that the Gaussian has the same shape in the spatial and Fourier domains and therefore does not incur the ringing effect in the spatial domain of the filtered image. See also Volume 2 Algorithms, “Gaussian filter”. See Figure 2.

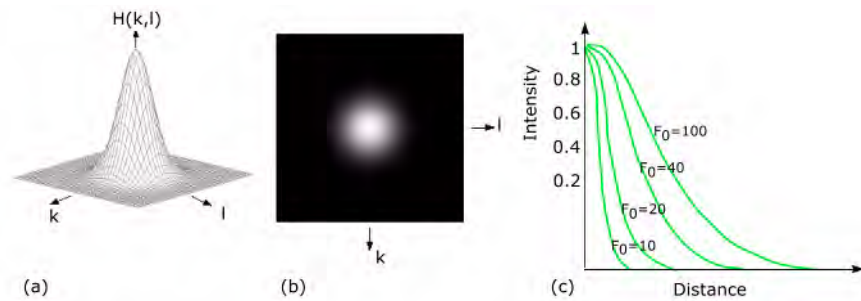


Figure 2. Gaussian low pass filter: (a) a plot of Gaussian function, (b) the inverse Fourier transform of Gaussian, (c) Frequency response of Gaussian with different  $F_0$

A commonly used discrete approximation to the Gaussian is the Butterworth filter. Applying this filter in the frequency domain shows a similar result to the Gaussian smoothing in the spatial domain. One difference is that the computational cost of the spatial filter increases with the size of the filter kernel, whereas the costs for a frequency filter are independent on the filter function. Hence, the spatial Gaussian filter is more appropriate for narrow lowpass filtering, while the Butterworth filter is a better implementation for wide lowpass filtering. See also “Gaussian and Butterworth filters”.

The same principles apply to highpass filters. A highpass filter function can be obtained by inverting the corresponding lowpass filter, e.g. an ideal highpass filter blocks all frequencies smaller than  $F_0$  and leaves the others unchanged.

A bandpass filter is a combination of both lowpass and highpass filters. It attenuates all frequencies smaller than a frequency  $F_{\min}$  and higher than a frequency  $F_{\max}$ , while the frequencies between the two cut-offs remain in the resulting output image.

A bandstop filter is a filter that passes most frequencies unaltered, but attenuates those in a specific range to very low levels.

---

## GAUSSIAN AND BUTTERWORTH FILTERS

In the frequency filtering step, both the real and imaginary parts of the fourier transform are scaled by the same formula, which depends only on the distance from the center of image.

The FIR filter must create two fourier transforms: one for the original image and one for the kernel. The Butterworth and Gaussian filters only need to create one fourier transform for the image since frequency scaling is done with a formula. Therefore, the FIR filter uses about twice as much memory as the Gaussian or Butterworth filters.

Centering is needed for the Gaussian and Butterworth filters since the scaling formulas are based on frequency magnitude increasing from the center of the image.

### Butterworth 2D and 3D filters

The following are the formulas for the Butterworth 2D and 3D filters, respectively

EQUATION 5

$$\begin{aligned}distsq &= (x - xcenter)^2 / (xcenter)^2 + (y - ycenter)^2 / (ycenter)^2 \\distsq &= (x - xcenter)^2 / (xcenter)^2 + (y - ycenter)^2 / (ycenter)^2 + \\&(z - zcenter)^2 / (zcenter)^2\end{aligned}$$

The real data is the product of the real data and the coefficient. The imaginary data is the product of the imaginary data and the coefficient.

### Butterworth low-pass, high-pass, band-pass, and band-stop filters

The following are the equations for the Butterworth low-pass, high-pass, band-pass, and band-stop filters.

Butterworth low-pass filter

$$\text{coefficient} = \frac{1.0}{1.0 + \left(\frac{\text{distsq}}{f1}\right)^{2 \cdot \text{butterworthorder}}}$$

### Butterworth high-pass filter

$$\text{coefficient} = \frac{1.0}{1.0 + \left(\frac{f1}{\text{distsq}}\right)^{2 \cdot \text{butterworthorder}}}$$

### Butterworth band-pass filter

$$\text{width} = f2 - f1, \text{centersq} = f1 \cdot f2$$

$$\text{num} = (\text{distsq} - \text{centersq})^{2 \cdot \text{butterworthorder}}$$

$$\text{coeff} = \frac{\text{num}}{(\text{num} + (\text{distsq} - \text{centersq})^{2 \cdot \text{butterworthorder}})}$$

### Band-stop filter

$$\text{width} = f2 - f1, \text{centersq} = f1 \cdot f2$$

$$\text{num} = (\sqrt{\text{distsq}} \cdot \text{width})^{2 \cdot \text{butterworthorder}}$$

$$\text{coeff} = \frac{\text{num}}{(\text{num} + (\sqrt{\text{distsq}} \cdot \text{width})^{2 \cdot \text{butterworthorder}})}$$

## Gaussian low- and high-pass filters

The following equations are for the Gaussian low- and high-pass filters.

### Gaussian low-pass (2D) filter

$$\text{coeff} = 1 - \text{gaussianhighpass}$$

### Gaussian low-pass (3D) filter

**Note:** The routine `bessj1` for determining a bessel function of the first kind and



$$\text{coefficient} = e^{-\frac{(x-x_{\text{center}})^2}{(2f_x^2 x_{\text{center}}^2)}} \cdot e^{-\frac{(y-y_{\text{center}})^2}{(2f_y^2 y_{\text{center}}^2)}} \cdot e^{-\frac{(z-z_{\text{center}})^2}{(2f_z^2 z_{\text{center}}^2)}}$$

first order is taken from "Numerical Recipes in C."

## WINDOWED FINITE IMPULSE RESPONSE

Window functions are time limited. This means there is always a finite integer  $N_w$  such that  $w(n)$  almost equal to 0 for all  $|n| > N_w$ . The final windowed impulse response is thus always time-limited, as needed for practical implementation. This window method designs a finite-impulse-response (FIR) digital filter based on the Hamming function.

Hamming function can be described as

EQUATION 6

$$w(n) = 0.53836 - 0.46164 \cdot \cos(2\pi n / N - 1)$$

The method consists of simply windowing a theoretically ideal filter impulse response  $h(n)$  by some suitably chosen window function  $w(n)$ , yielding the following equation:

EQUATION 7

$$h_w(n) = w(n) * h(n), \text{ where } n \text{ belongs to } Z$$

For example, the impulse response of the ideal lowpass filter is the *sinc* function.

EQUATION 8

$$h(n) = B * \text{sinc}(Bn) \text{ almost equal to } B * (\sin(\pi * Bn) / \pi * Bn), \text{ where } n \text{ belongs to } Z$$

where  $B = 2f_c$  is the normalized bandwidth of the lowpass filter and  $f_c$

denotes the cut-off frequency.

Since  $h(n)=\text{sinc}(BnT)$  decays away from time 0 as  $1/n$ , the method truncates it to the interval  $[-N, N]$  for some sufficiently large  $N$  defined by the user and as a result it obtains a finite response filter that approximates the ideal filter.

As for Hamming function,  $|N| > 3$  is sufficient for good approximation.

#### Outline of the method

The Filters Frequency method processes images by filtering in the frequency domain in three steps:

- 1 It performs a forward Fast Fourier Transform (FFT) to convert a spatial image into its frequency image.
- 2 It enhances some frequency components of the image and attenuates other frequency components of the image by using a lowpass, highpass, bandpass, or bandstop filter. Frequency filters may be constructed with 1 of 3 methods: Hamming windows, Gaussian filters, and Butterworth filters. However, for the Gaussian filters only lowpass and highpass filters are available.
- 3 It performs an inverse FFT to convert from the frequency domain back into the spatial domain.

This method performs all three steps in a single combined process.

Note: Since the core algorithm of this module is the FFT, it requires that all the dimensions of an  $N$ -dimensional dataset were powers of two. To be able to use this algorithm on datasets with arbitrary dimensions, the data is zero padded to powers of two before applying the forward FFT and stripped down to the original dimensions after applying the inverse FFT. The powers of two are not kept identical because symmetrical Fourier pictures are not required.

The module also creates a Gabor filter, which is essentially a tilted Gaussian with two unequal axes at an offset ( $\text{freqU}$ ,  $\text{freqV}$ ) from the origin. A Gabor filter only responds to a texture having both a particular frequency and a particular orientation. Note that a filter and its mirror image reflected across the  $u$  and  $v$  frequency axes produce identical frequency responses.

See also Volume 2 Algorithms, “Gabor Filter”.

The Frequency Filter algorithm also used in homomorphic filtering, for more information refer to Volume 2 Algorithms, “Homomorphic Filter”.

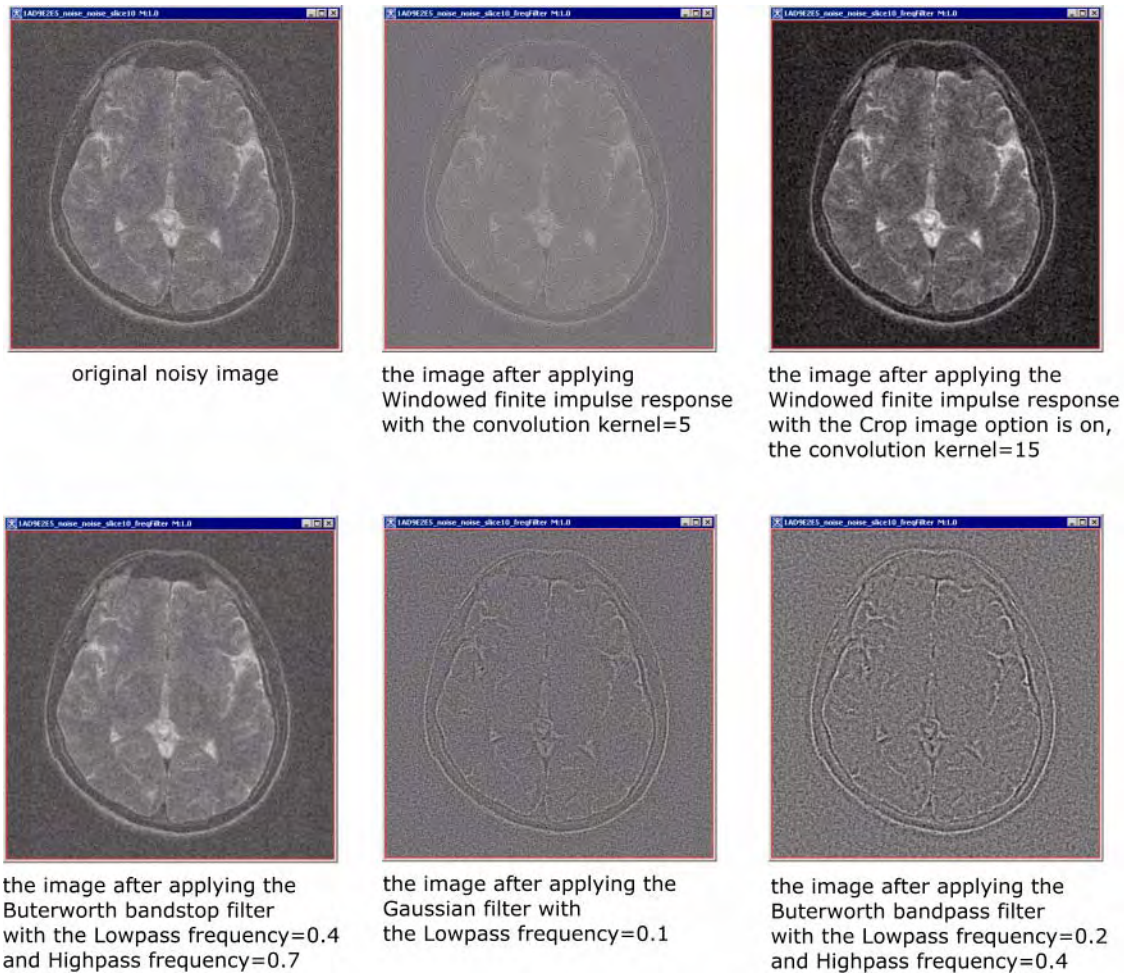


Figure 3. Applying the Frequency Filter algorithm with different parameters

## REFERENCES

Digital Image Processing Second Edition by Rafael C. Gonzalez and Richard E. Woods, Prentice-Hall, Inc., 2002, Chapter 4.5, pp. 191-194.

## IMAGE TYPES

2D and 3D grayscale images.

## Applying the Frequency Filter algorithm

To run this algorithm, complete the following steps:

- 1 Open an image of interest.
- 2 Select Algorithms > Filters Frequency.
- 3 The Frequency Filter dialog box appears.
- 4 Fill out the dialog box.
- 5 Specify the destination image frame, and then, press OK.

Depending on whether you selected the New Image or Replace Image option, the result appears in a new window or replaces the original image.

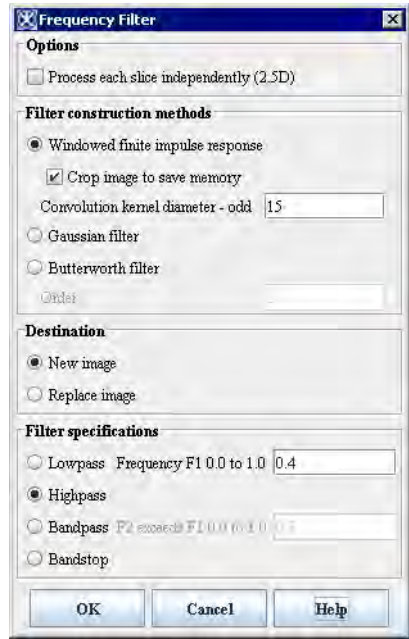
<b>Process each slice independently (2.5D)</b>	If true, processes each slice of a 3D image independently.	
<b>Windowed finite impulse response</b>	If selected, the method will use the Windowed finite impulse response algorithm.	
<b>Crop image to save memory</b>	If selected, this option crops image if the (largest image dimension + kDim - 1) value exceeds the smallest power of 2 number which is more or equal to the largest dimension.	
<b>Convolution kernel diameter—odd</b>	It must be an odd integer larger or equal than 3, see also “Windowed Finite Impulse Response” .	
<b>Gaussian filter</b>	If selected the method uses the Gaussian filter.	
<b>Butterworth filter</b>	If selected the method uses the Butterworth filter.	

Figure 4. Extract Object dialog box

<b>Order</b>	The order of the Butterworth filter.
<b>Destination</b>	
<b>New image</b>	The result image appears in a new image frame
<b>Replace image</b>	The result image appears in the same image frame, replacing the result image.
<b>Filter specifications</b>	
<b>In that part of the dialog box, first select the filter type. Choose among: Lowpass, Highpass, Bandpass and Bandstop option. Then. Enter the lowpass frequency value, and also the higher frequency cutoff value if the Bandpass option is chosen.</b>	
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes this dialog box.
<b>Help</b>	Displays online help for this dialog box.

**Figure 4. Extract Object dialog box (continued)**

## Filters (Spatial): Adaptive Noise Reduction

This algorithm reduces noise without blurring the edges by replacing a pixel value with a weighted sum of all local pixels reached by following a path with small pixel intensity values between neighboring pixels.

### Background

The Adaptive Noise Reduction algorithm is a port to Adaptive Noise Removal Filter, a software program. This program first transforms *RGB* color space into *YCrCb* color space. When the filtering is complete, it then transforms the *YCrCb* color space into *RGB* color space. Hence, all of the color operations occur in *YCrCb* space. However, the program does not transform black-and-white images into another space.

In “Frequently Asked Questions About Color,” Charles Poynton reveals that the *RGB-to-YCrCb* and *YCrCb-to-RGB* conversion equations used in Adaptive Noise Removal Filter differ significantly from standard ones. First, the program creates an edge graph or pixel neighbor graph using a weakly median-filtered version of the *YCrCb* space. A weakly median-filter version means that median filtering only occurs if the pixel is not on an edge in *Y*, *Cr*, or *Cb* space. The edge graph is an array of arrays with the first array having a length equal to the number of pixels in the image and the second array having eight edge weights. The eight edge weights are calculated from the intensity differences between the center pixel and one of the eight neighboring pixels:

$$E = \sqrt{((dYwY)^2 + (dRwR)^2 + (dBwB)^2)}$$

where *E* is the edge weight; *dY*, *dR*, and *dB* are the *Y*, *Cr*, and *Cb* differences; and *wY*, *wR*, and *wB* are the *Y*, *Cr*, and *Cb* weights derived from the *Y*, *Cr*, and *Cb* radiuses. A bigger relative radius results in a smaller weight for radiuses  $\geq 0.1$ , which is typically the case.

After the edge graph is created, the program filters the *Y*, *Cr*, and *Cb* spaces separately. It uses the selected pixel as the center of a square with  $L = 2R + 1$  where *L* = sides of length, *R* = range, and range = (int)(radius + 0.999).

The *enhancePixel* function returns the filtered pixel value at every pixel in the image. If neighboring pixels are connected to the center pixel via a low-cost path, queue elements corresponding to these neighboring pixels are added to the priority queue. If no queue element is retrieved from the queue, then the original pixel value is passed unchanged. If elements are retrieved from the queue, then those *Y*, *Cr*, or *Cb* pixel values that retrieved queue elements are put into a weighted sum. The weight is derived from an index into the filter array given by the queue element key times the *invFilterLen*. The filter array is generated with a Gaussian function.

Each queue element has a value, a *baseKey*, and a key. The *value* gives the location within the local  $(2R + 1)(2R + 1)$  local square where  $R = \text{range}$ . The *baseKey* has a sum of the *edgeGraph* values over the low-cost path from the center pixel to the selected pixel, and *key* is equal to the sum of *baseKey* and the *distMap* value for the selected pixel. The *distMap* value at a given position inside the local square is equal to the edge preservation strength times the distance from the center pixel. A key is only added to the queue if  $\text{key} < \text{maxDistance}$ :

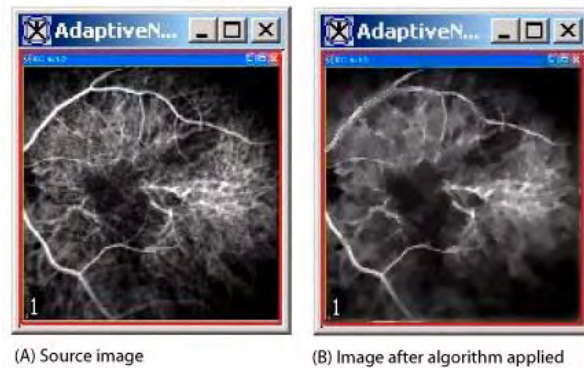
$$M = (\text{int})(Rd(S + 512))$$

where  $M = \text{maxDistance}$ ,  $Rd = \text{radius}$ , and  $S = \text{edge preservation strength}$ .

The priority queue is implemented as a binary heap and the binary heap is implemented as an array. A binary heap is a nearly full binary tree (only the bottom level may not be complete). This binary heap has the property that for every node other than a leaf, its key value is less than or equal to that of its children. In a binary heap the item of highest priority is always at the root of the tree or at node 0. When the highest priority item at the root is removed, the item of highest priority among the remainder moves into the root position. The children of node  $i$  are at  $2i + 1$  and  $2i + 2$ . The parent of

node  $i$  is at  $(\text{int})\left(\frac{i-1}{2}\right)$ .

The filter radius can be separately adjusted for each *Y*, *Cr*, and *Cb* channel for color images. For black-and-white images only, the *Y* filter radius can be adjusted. A larger radius removes more noise but loses more detail.



**Figure 1. Adaptive Noise Reduction processing**

The edge preservation strength can be adjusted. Larger values preserve more edges but sacrifice smoothness in low-contrast areas. Very large values may cause banding.

To speed the filtering on color images, you can select the Filter *Cr* and *Cb* at halved dimensions check box. When run, the algorithm shrinks the *Cr* and *Cb* spaces by a factor of 2, generates an edge table for the shrunken space, filters the image, and then expands the filtered *Cr* and *Cb* spaces to their original dimensions. Although the algorithm runs faster, it also loses some details.

The New image option causes MIPAV to place the resulting image in a new image window. The Replace image option causes MIPAV to place the changed image in the same window.

---

## IMAGE TYPES

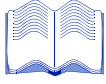
You can apply this algorithm to both color and black-and-white 2D images.

---

## SPECIAL NOTES

None.





---

## REFERENCES

Refer to the following references for more information:

Karlis Freivalds, "Adaptive Noise Removal Filter," a software program, University of Latvia, Institute of Mathematics and Computer Science, at <http://www.gradetools.com/karlisf>.

Charles Poynton, "Frequently Asked Questions About Color," at <http://www.poynton.com>.

## Applying the Adaptive Noise Reduction algorithm

To run this algorithm, complete the following steps:

- 1** Open an image.
- 2** Perform, as an option, any image processing, such as improving the contrast, on the image.
- 3** Select Algorithms > Filters (spatial) > Adaptive noise reduction. The Adaptive Noise Reduction dialog box opens.
- 4** Complete the information in the dialog box.
- 5** Click OK. The algorithm begins to run.

A pop-up window appears with the status. A series of status messages appear in the window.

When the algorithm finishes running, the pop-up window closes.

Depending on whether you selected New image or Replace image, the results appear in a new window or replace the image to which the algorithm was applied.

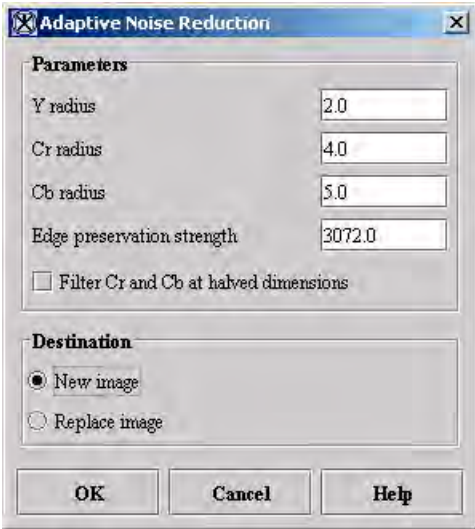
<b>Y radius</b>	<p>Specifies the local square used for <i>Y</i> space filtering, which is approximately <math>(2YRadius + 1)(2YRadius + 1)</math> in size.</p> <p>As the radius increases, the algorithm removes more noise, but loses more detail. The default value is 2.0.</p>	
<b>Cr radius</b>	<p>Specifies the local square used for <i>Cr</i> space filtering (only available for color), which is approximately in size.</p> <p><math>(2CrRadius + 1)(2CrRadius + 1)</math></p> <p>As the radius increases, the algorithm removes more noise, but loses more detail. The default value is 4.0.</p>	
<b>Cb radius</b>	<p>Specifies the local square used for <i>Cb</i> space filtering, which is approximately <math>(2CbRadius + 1)(2CbRadius + 1)</math> in size. As the radius increases, the algorithm removes more noise but loses more detail. The default value is 5.0.</p>	
<b>Edge preservation strength</b>	<p>Specifies a value for preserving edges. Larger values preserve more edges but sacrifice smoothness in low-contrast areas. Very large values may cause banding. For color images, the default value is 3072.0. For black-and-white images, the default value is <math>\left(\frac{12.0}{255.0}\right)(IMx - IMn)</math> where <i>IMx</i> is image maximum and <i>IMn</i> is image minimum.</p>	
<b>Filter Cr and Cb at halved dimensions</b>	<p>Causes the algorithm to run faster but with some loss of detail. For color images, the default value is enabled; for other images, the default value is disabled.</p>	
<b>New image</b>	Shows the results of the algorithm in a new image window.	
<b>Replace image</b>	Replaces the original source image with the results of the algorithm.	
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.	
<b>Cancel</b>	Disregards any changes you made in this dialog box and closes the dialog box.	
<b>Help</b>	Displays online help for this dialog box.	

Figure 2. Adaptive Noise Reduction dialog box

---

## Filters (Spatial): Adaptive Path Smooth

By replacing a pixel value with a weighted sum of the local pixels on the low-cost path, this algorithm reduces noise without blurring edges. It determines the weighted sum by following the path from the lowest-cost pixel at the edge of a local neighborhood back to the original pixel.

### Background

MIPAV uses a modified port of image noise removal software by Karlis Freivalds for this algorithm. Freivalds' program starts by transforming *RGB* color space into *YCrCb* color space. After filtering is complete, the program then transforms the *YCrCb* color space back into *RGB* color space. Hence, all of the color operations occur in *YCrCb* space.

"Frequently Asked Questions about Color" by Charles Poynton reveals that the *RGB*-to-*YCrCb* and *YCrCb*-to-*RGB* conversion equations used in Freivalds' program differ significantly from the standard ones. Grayscale images are not transformed into another space.

In the original program as ported in the Adaptive Noise Reduction algorithm, the weighted sum was applied to all pixels in the  $(2 \times \text{range} + 1)$  by  $(2 \times \text{range} + 1)$  local square where  $\text{range} = (\text{int})(\text{radius} + 0.999)$ . The program finds the lowest-cost pixel at the edge of the  $(2 \times \text{range} + 1)$  by  $(2 \times \text{range} + 1)$  local neighborhood. That pixel and the pixels along the low-cost path back to the original center pixel are the only pixels used in the weighted sum.

First, the program creates an edge graph or pixel neighbor graph. The 2D *edge graph* is an array of arrays with the first array having a length equal to the number of pixels in the image and the second array having eight edge weights. The eight edge weights are calculated from the intensity differences between the center pixel and one of the eight neighboring pixels.

In 2D and 2.5D images, the edge graph encompasses the entire image. In 3D with 26 nearest neighbors, an edge graph of size

$$(2 \times \text{range} + 1) \times (2 \times \text{range} + 1) \times (2 \times \text{range} + 1) \times 26$$

is separately created for every voxel in the image to conserve memory.

For color images:

$$EW = \sqrt{(dy \times wy)^2 + (dr \times wr)^2 + (dB \times wB)^2}$$

where

*EW* = edge weight

*dY*, *dR*, and *dB* = *Y*, *Cr*, and *Cb* differences

*wY*, *wR*, and *wB* = *Y*, *Cr*, and *Cb* weights derived from the *Y*, *Cr*, and *Cb* radiuses

A bigger relative radius results in a smaller weight for radiuses  $\geq 0.1$ , which should typically be the case. For grayscale images, edge weights are simply the absolute values of the differences between two pixels.

After the edge graph is created, the *Y*, *Cr*, and *Cb* spaces are separately filtered. The filter uses the selected pixel as the center of a square with sides of  $length = 2 \times range + 1$ .

The function *enhancePixel* returns the filtered pixel value at every pixel in the image. If neighboring pixels are connected to the center pixel via a low-cost path, queue elements corresponding to these neighboring pixels are added to the priority queue. If no queue element is retrieved from the queue, then the original pixel value is passed unchanged. If elements are retrieved from the queue, then those *Y*, *Cr*, or *Cb* pixel values that retrieved queue elements in the path from the low-cost edge pixel to the center pixel are put into a weighted sum. The weight, derived from the *distMap* array, is simply the Gaussian function of the distance from the center pixel.

Each queue element has a value, a key, and an origin as indicated in the following:

- *value* = the location within the local  $(2 \times \text{range} + 1) * (2 \times \text{range} + 1)$  local square
- *key* = sum of the edge graph values on the low-cost path from the center pixel to the selected pixel
- *origin* = the value of the queue element that provides the neighboring pixel on the low-cost path back to the center pixel with the *center pixel origin* = -1

A key is only added to the queue if an edge weight is less than threshold.

The priority queue is implemented as a binary heap and the binary heap is implemented as an array. A binary heap is a “nearly full” binary tree (only the bottom level may not be complete). This binary heap has the property that, for every node other than a leaf, its key value is less than or equal to that of its children. In a binary heap, the item of highest priority is always at the root of the tree or at node 0. When the highest priority item at the root is removed, the item of highest priority among the remainder moves into the root position. The children of node  $i$  are at  $2 \times i + 1$  and  $2 \times i + 2$ . The parent of node  $i$  is at  $(int)\frac{(i-1)}{2}$ .



**Figure 1. Example of adaptive path smooth processing**

On the Adaptive Path Smooth dialog box for color images, you can separately adjust the filter radius for each *Y*, *Cr*, and *Cb* channel for color images. On the Adaptive Path Smooth dialog box for grayscale images, you can only adjust a single radius.



**Tip:** A larger radius removes more noise from the image but also loses more detail from the image.

You can also adjust the threshold for both color and grayscale images. Again, larger values preserve more edges but sacrifice smoothness in low-contrast areas. In addition, banding may appear for very large values.

For color images only, there is an option to filter *Cr* and *Cb* at halved dimensions. If this option is selected, the *Cr* and *Cb* spaces are shrunk by a factor of 2, an edge table is generated for the shrunk space, the filtering is performed, and the filtered *Cr* and *Cb* spaces are expanded back up to their original dimensions. The algorithm runs faster, but the image loses detail.

On both dialog boxes, if you choose New image, MIPAV then places the resulting image in a new image window. Selecting the Replace image option causes MIPAV to place the changed image in the same window.

---

## IMAGE TYPES

You can apply this algorithm to both color and grayscale 2D and 3D images.

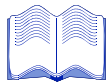
---

## SPECIAL NOTES

None.

---

## REFERENCE



Refer to the following references for more information about this algorithm:

Image noise removal filter software by Karlis Freivalds at <http://www.gradetools.com/karlisf>

Frequently Asked Questions About Color by Charles Poynton at <http://www.poynton.com/colorFAQ.html>.

## Applying the Adaptive Path Smooth algorithm

To run this algorithm, complete the following steps:

- 1** Open an image.
- 2** Select Algorithms > Filter(spatial) > Adaptive path smooth.

Depending on whether the image is grayscale or color, the Adaptive Path Smooth dialog box for grayscale images (Figure 2) or the Adaptive Path Smooth dialog box for color images opens. If the image is 3D, a check box to process each slice independently appears (Figure 3).

**3** Complete the information in the dialog box.

**4** Click OK.

The algorithm begins to run, and a progress bar appears with the status. When the algorithm finishes running, the progress bar disappears, and the results appear either in a new window or they replace the original source image.

<b>Radius</b>	The local square used is approximately $(2 \times \text{Radius} + 1)$ times $(2 \times \text{Radius} + 1)$ in size. As the radius increases the algorithm removes more noise but loses more details from the image.  The default value is 2.0.	
<b>Threshold</b>	Larger values preserve more edges but sacrifice smoothness in low-contrast areas. For very large values, banding may appear.  The default value is $0.1 \times (\text{image maximum} - \text{image minimum})$ .	
<b>Include neighbors of low cost path</b>	Adds nearest neighbors of path pixels to the queue if they differ from the path pixels by less than threshold. However, the nearest neighbor weight is only half that of a path pixel. The default value is <i>not</i> selected.	
<b>Process each slice independently</b>	Filters each slice of the dataset independently of adjacent slices. The default value is <i>not</i> selected. (This check box only appears for 3D images.)	
<b>New image</b>	Shows the results of the algorithm in a new image window (default choice).	
<b>Replace image</b>	Replaces the current active image with the new image produced by the algorithm.	
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.	
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes this dialog box.	
<b>Help</b>	Displays online help for this dialog box.	

**Figure 2. Adaptive Path Smooth dialog box for grayscale images**



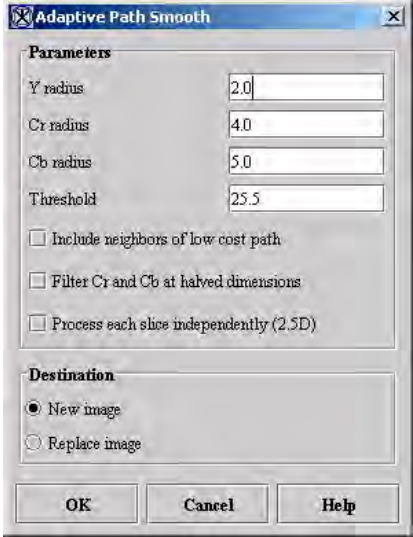
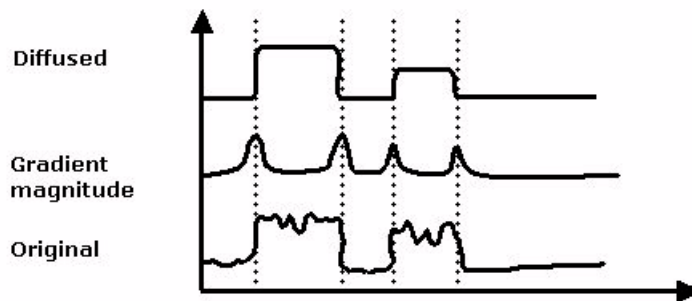
<b>Y radius</b>	The local square used for <i>Y</i> space is approximately $(2 \times YRadius + 1)$ times $(2 \times YRadius + 1)$ in size. As the radius increases the algorithm removes more noise but loses more details from the image. The default value is 2.0.	
<b>Cr radius</b>	The local square used for <i>Cr</i> space filtering is approximately $(2 \times CrR + 1) \times (2 \times CrR + 1)$ in size. As the radius increases, the algorithm removes more noise but loses more details from the image. The default value is 4.0.	
<b>Cb radius</b>	The local square used for <i>Cb</i> space filtering is approximately $(2 \times CbR + 1) \times (2 \times CbR + 1)$ in size. As the radius increases, the algorithm removes more noise but loses more detail from the image. The default value is 5.0.	
<b>Threshold</b>	Larger values preserve more edges but sacrifice smoothness in low-contrast areas. For very large values, banding may appear. The default value is $0.1 \times (\max(\text{red max}, \text{green max}, \text{blue max}) - \min(\text{red min}, \text{green min}, \text{blue min}))$ .	
<b>Include neighbors of low cost path</b>	Adds nearest neighbors of path pixels to the queue if they differ from the path pixels by less than threshold. However, the nearest neighbor weight is only half that of a path pixel. The default value is <i>not</i> selected.	
<b>Filter Cr and Cb at halved dimensions</b>	When this option is selected, the algorithm runs faster but loses some detail. The default value is <i>not</i> selected.	
<b>Process each slide independently</b>	Filters each slice of the dataset independently of adjacent slices. The default value is <i>not</i> selected. (This check box only appears for 3D images.)	
<b>New image</b>	Shows the results of the algorithm in a new image window (default choice).	
<b>Replace image</b>	Replaces the current active image with the new image produced by the algorithm.	
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.	
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes this dialog box.	
<b>Help</b>	Displays online help for this dialog box.	

Figure 3. Adaptive Path Smooth dialog box for 3D color images

## Filters (Spatial): Anisotropic Diffusion

This algorithm anisotropically diffuses an image. That is, it blurs over regions of an image where the gradient magnitude is relatively small (homogenous regions) but diffuses little over areas of the image where the gradient magnitude is large (i.e., edges). Therefore, objects are blurred, but their edges are blurred by a lesser amount.

Figure 1 illustrates the relationship between anisotropic diffusion and gradient magnitude. In the figure, the original signal is noisy but two major peaks are still visible. Applying the diffusion process to the original signal and using the gradient magnitude to attenuate the diffusion process at places where the edge is strong produces a signal in which the overall signal-to-noise ratio has improved.



**Figure 1. A 1D example of anisotropic diffusion**

*Note how the noise of the diffused signal is removed while only blurring the edge a small amount. The diffusion at the edge of the signals is attenuated by using a function of the gradient magnitude.*

## Background

The following is the anisotropic diffusion equation:

EQUATION 1

$$I_t = \text{div}(c(\hat{x},t)\nabla I) = c(\hat{x},t)\nabla^2 I + \nabla c \cdot \nabla I$$

$$I(\hat{x},0) = I_0(\hat{x})$$

$$c(\hat{x},t) = e^{-\left(\frac{\|\nabla I\|}{k}\right)^2} \quad \text{or}$$

$$c(\hat{x},t) = \frac{1}{1 + \left(\frac{\|\nabla I\|}{k}\right)^2}$$

where  $c(\hat{x},t)$  is the variable conductance term. Typically, a function of the gradient magnitude  $\|\nabla I\|$ , if  $c(\hat{x},t)$  is a constant, then  $I_t = c\nabla^2 I$  which is the isotropic heat diffusion equation (Gaussian blurring).

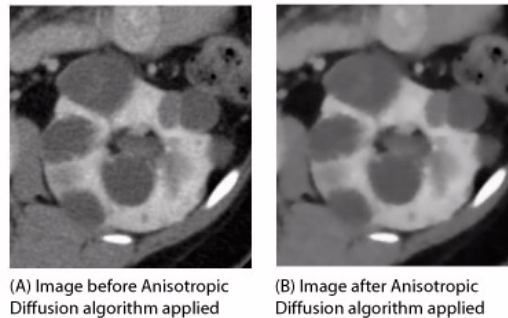
In simple terms, the image  $I$  at some time  $t$  is given by the diffusion of the image where conductance term limits the diffusion process. This program uses a conductance term calculated by dividing a constant ( $k$ ) divided by the square root of the sum of the squares of the normalized first order derivatives of the Gaussian function as an approximate gradient function. Figure 2 shows the result of applying this technique. The image on the left is a CT image before processing. The image on the right of it was diffused. Note the image was diffused within object boundary but to a much lesser

amount near the boundaries of the objects.

---

## IMAGE TYPES

You can apply this algorithm just to gray-scale images.

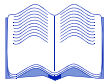


**Figure 2. An image before and after applying the Anisotropic Diffusion algorithm**

---

## SPECIAL NOTES

Refer to the section on “Filters (Spatial): Nonlinear Noise Reduction” on page 279.



## REFERENCES

Refer to the following references for more information:

P. Perona and J. Malik. “Scale-Space and Edge Detection Using Anisotropic Diffusion,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, 12(7):629-639, July 1990.

Bart M. Ter Romeney, ed. *Geometry-Driven Diffusion in Computer Vision*. (Dordrecht, The Netherlands: Kluwer Academic Publishers, 1994).

---

## Applying the Anisotropic Diffusion algorithm

To run this algorithm, complete the following steps:

- 1** Open an image.
- 2** Perform, as an option, any image processing, such as improving the contrast, eliminating noise, etc., on the image.
- 3** Select Algorithms > Filters (spatial) > Anisotropic Diffusion. The Anisotropic Diffusion dialog box (Figure 3) opens.
- 4** Complete the information in the dialog box.

---

**Note:** Diffusing an image is a lengthy process. Give some consideration to the values you type into the Iterations and  $k$  boxes. The higher the number of iterations slows processing speed as does a small  $k$  value.

---

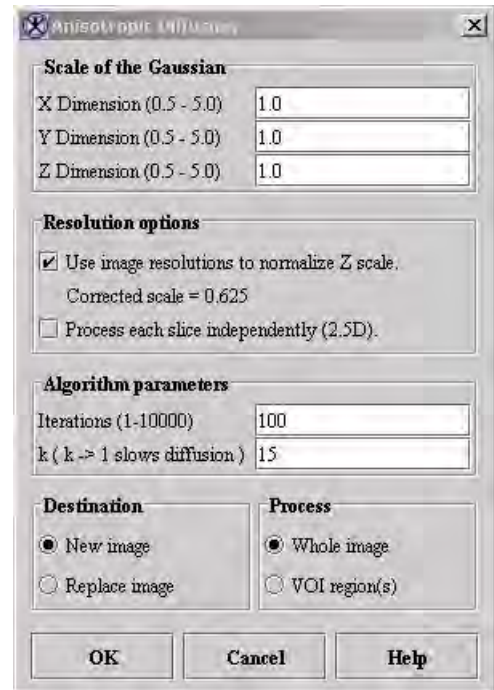
- 5** Click OK. The algorithm begins to run.

A pop-up window appears with the status. The following message appears: “Diffusing image.”

When the algorithm finishes running, the pop-up window closes.

Depending on whether you selected New image or Replace image, the results appear in a new window or replace the image to which the algorithm was applied.

<b>X dimension (0.5 - 5.0)*</b>	Indicates the standard deviation (SD) of Gaussian in the X direction.
<b>Y dimension (0.5 - 5.0)*</b>	Indicates the SD of Gaussian in the Y direction.
<b>Z dimension (0.5 - 5.0)*</b>	Indicates the SD of Gaussian in the Z direction.
<b>Use image resolutions to normalize Z scale</b>	Normalizes the Gaussian to compensate for the difference if the voxel resolution is less between slices than the voxel resolution in-plane. This option is selected by default.
<b>Process each slice independently (2.5D)</b>	Blurs each slice of the dataset independently.
<b>Iterations (1-1000)</b>	Specifies the number of times the process is run.
<b>k (k &gt; 1 slows diffusion)</b>	Specifies the constant used to control the blurring rate. A small value slows the diffusion process (especially across edges) per iteration. A large value speeds the diffusion process but also allows for diffusion (blurring across edges); it also makes this function act similar to Gaussian blurring. See equation above.
<b>New image</b>	Shows the results of the algorithm in a new image window.
<b>Replace image</b>	Replaces the current active image with the results of the algorithm.
<b>Whole image</b>	Applies the algorithm to the whole image.
<b>VOI regions</b>	Applies the algorithm to the volumes (regions) delineated by the VOIs.
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.



\*An SD greater than 1 attenuates small scale edges typically caused by noise, but reduces edge accuracy.

Figure 3. Anisotropic Diffusion dialog box

---

## Filters (Spatial): Coherence-Enhancing Diffusion

*Coherence-enhancing filtering is useful for filtering relatively thin, linear structures such as blood vessels, elongated cells, and muscle fibers.*

*Coherence-enhancing filtering is a specific technique within the general classification of diffusion filtering. Diffusion filtering, which models the diffusion process, is an iterative approach of spatial filtering in which image intensities in a local neighborhood are utilized to compute new intensity values.*

Two major advantages of diffusion filtering over many other spatial domain filtering algorithms are:

- *A priori* image information can be incorporated into the filtering process
- The iterative nature of diffusion filtering allows for fine grain control over the amount of filtering performed

There is not a consistent naming convention in the literature to identify different types of diffusion filters. This documentation follows the approach used by Weickert (see “References” on page 235). Specifically, since the diffusion process relates a concentration gradient with a flux, *isotropic diffusion* means that these quantities are parallel. *Regularized* means that the image is filtered prior to computing the derivatives required during the diffusion process. In linear diffusion the filter coefficients remain constant throughout the image, while *nonlinear* diffusion means the filter coefficients change in response to differential structures within the image. Coherence-enhancing filtering is a regularized *nonlinear* diffusion that attempts to smooth the image in the direction of nearby voxels with similar intensity values.

---

For more information about the algorithm, refer to MIPAV Volume 2 User Guide on the MIPAV web site: {<http://mipav.cit.nih.gov/>}

---

## Background

All diffusion filters attempt to determine the image  $I(x, y)$  that solves the well-known diffusion equation, which is a second-order partial differential defined as

$$\partial_t I = \text{div}(D\nabla I)$$

where

$\partial_t I$  = The time derivative of the image

$\text{div}$  = The divergence operator

$D$  = The diffusion tensor (which may be a scalar value)

$\nabla I$  = The gradient of the image

The quantity that distinguishes different diffusion filters is primarily the diffusion tensor also called the diffusivity.

In *homogeneous linear diffusion* filtering, the diffusivity,  $D$ , is set to 1 and the diffusion equation becomes:

$$\partial_t I = \partial_{xx} I + \partial_{yy} I$$

In *isotropic nonlinear diffusion* filtering, the diffusivity term is a monotonically decreasing scalar function of the gradient magnitude squared, which encapsulates edge contrast information. In this case, the diffusion equation becomes:

$$\partial_t I = \text{div}(D(|\nabla I|^2)\nabla I)$$

It is well known that derivative operations performed on a discrete grid are an ill-posed problem, meaning derivatives are overly sensitive to noise. To convert derivative operations into a well-posed problem, the image is low-pass filtered or smoothed prior to computing the derivative. *Regularized isotropic (nonlinear) diffusion* filtering is formulated the same as the *isotropic nonlinear diffusion* detailed above; however, the image is smoothed prior to computing the gradient. The diffusion equation is modified slightly to indicate regularization by including a standard deviation term in the image gradient as shown in the following equation:



$$\partial_t I = \text{div}\left(D\left(|\nabla I_\sigma|^2\right)\nabla I_\sigma\right)$$

The smoothing to regularize the image is implemented as a convolution over the image and therefore this filtering operation is linear. Since differentiation is also a linear operation, the order of smoothing and differentiation can be switched, which means the derivative of the convolution kernel can be computed and convolved with the image resulting in a well-posed measure of the image derivative.

In *edge-enhancing anisotropic diffusion*, the diffusivity function allows more smoothing parallel to image edges and less smoothing perpendicular to these edges. This variable direction smoothing means that the flux and gradient vectors no longer remaining parallel throughout the image, hence the inclusion of the term *anisotropic* in the filter name. Directional diffusion requires that the diffusivity function provide more information than a simple scalar value representing the edge contrast. Therefore, the diffusivity function generates a *matrix tensor* that includes directional information about underlying edges. We refer to a tensor-valued diffusivity function as a *diffusivity tensor*. In *edge-enhancing anisotropic diffusion*, the diffusivity tensor is written as the following:

$$D(\nabla I_\sigma) = D\left(\nabla I_\sigma \nabla I_\sigma^T\right)$$

Directional information is included by constructing an orthonormal system of eigenvectors  $v_1, v_2, \dots, v_n$  of the diffusivity tensor so that  $v_1 \parallel \nabla I_\sigma$  and  $v_2 \perp \nabla I_\sigma, \dots, v_n \perp \nabla I_\sigma$ .

*Coherence-enhancing anisotropic diffusion* is an extension of *edge-enhancing anisotropic diffusion* that is specifically tailored to enhance line-like image structures by integrating orientation information. The diffusivity tensor in this case becomes:

$$D(\nabla I_\sigma) = D\left(K_\rho \otimes \nabla I_\sigma \nabla I_\sigma^T\right)$$

In this diffusivity tensor,  $K_\rho$  is a Gaussian kernel of standard deviation  $\rho$ , which is convolved ( $\otimes$ ) with each individual component of the  $\nabla I_\sigma \nabla I_\sigma^T$  matrix. Directional information is provided by solving the eigensystem of the diffusivity tensor without requiring  $v_1 \parallel \nabla I_\sigma$  and  $v_2 \perp \nabla I_\sigma, \dots, v_n \perp \nabla I_\sigma$ .

Figure 2 shows the results of applying the coherence-enhancing anisotropic diffusion filter to an example MR knee image.

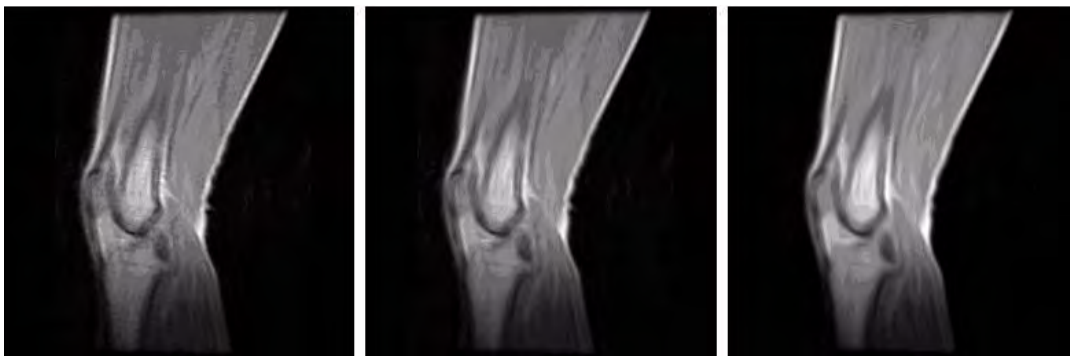


Figure 1. Example of coherence-enhancing diffusion

---

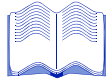
## IMAGE TYPES

You can apply this algorithm to all data types except complex and to 2D, 2.5D, and 3D images.

---

## SPECIAL NOTES

The resulting image is, by default, a float image.



## REFERENCES

Refer to the following references for more information about diffusion filtering in general and this algorithm in particular.

Weickert, Joachim. "Nonlinear Diffusion Filtering," in *Handbook of Computer Vision and Applications*, Volume 2, eds. Bernd Jahne, Horst Haussecker, and Peter Geissler. (Academic Press, April 1999), 423–450.

Weickert, Joachim. *Anisotropic Diffusion in Image Processing* (Stuttgart, Germany: Teubner, 1998).

## Applying the Coherence-Enhancing Diffusion algorithm

To run this algorithm, complete the following steps:

- 1 Select Algorithms > Filter > Coherence-Enhancing Diffusion. The Coherence-Enhancing Diffusion dialog box opens (Figure 2).

<b>Number of iterations</b>	Specifies the number of iterations, or number of times, to apply the algorithm to the image.
<b>Diffusivity denominator</b>	Specifies a factor that controls the diffusion elongation.
<b>Derivative scale space</b>	Specifies the standard deviation of the Gaussian kernel that is used for regularizing the derivative operations.
<b>Gaussian scale space</b>	Specifies the standard deviation of the Gaussian filter applied to the individual components of the diffusivity tensor.
<b>Process each slice separately</b>	Applies the algorithm to each slice individually. By default, this option is selected.
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

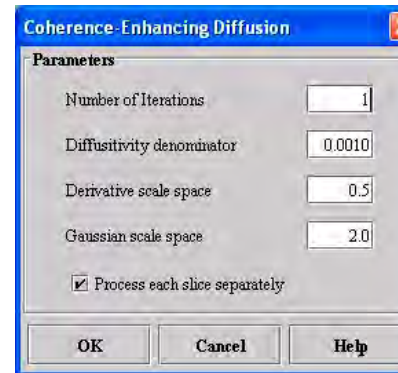
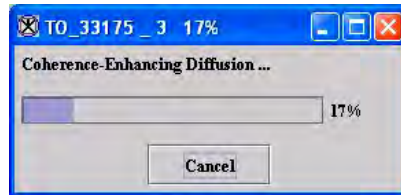


Figure 2. Coherence Enhancing Diffusion dialog box

- 2 Complete the fields in the dialog box.

**3** When complete, click OK.

The algorithm begins to run, and a status window appears. When the algorithm finishes, the resulting image appears in a new image window.



**Figure 3.** Status message that appear when the Coherence-Enhancing Diffusion algorithm is running

## Filters (Spatial): Gaussian Blur

This algorithm blurs an image or the VOI of the image with a Gaussian function at a user-defined scale sigma (standard deviation [SD]). In essence, convolving a Gaussian function produces a similar result to applying a low-pass or smoothing filter. A *low-pass filter* attenuates high-frequency components of the image (i.e., edges) and passes low-frequency components. This results in the blurring of the image. *Smoothing filters* are typically used for noise reduction and for blurring. The standard deviation of the Gaussian function controls the amount of blurring. A large standard deviation (i.e.,  $> 2$ ) significantly blurs, while a small standard deviation (i.e., 0.5) blurs less. If the objective is to achieve noise reduction, a rank filter (median) might be more useful in some circumstances.

Advantages to convolving the Gaussian function to blur an image include:

- Structure is not added to the image.
- It, as well as the Fourier Transform of the Gaussian, can be analytically calculated.
- By varying the SD, a Gaussian scale space can easily be constructed.

## Background

The radially symmetric Gaussian, in an  $n$ -dimensional space, is:

$$G(\hat{x}, \sigma) = \frac{1}{(2\pi\sigma)^{n/2}} e^{-\hat{x} \cdot \hat{x} / (2\sigma^2)}$$

EQUATION 1

where  $\sigma$  (referred to as the *scale*), is the standard deviation of the Gaussian and is a free parameter that describes the width of the operator (and thus the degree of blurring) and  $n$  indicates the number of dimensions.

Geometric measurements of images can be obtained by spatially convolving (shift-invariant and linear operation) a Gaussian neighborhood operator with an image,  $I$ , where  $I: \mathbb{R}^n \rightarrow \mathbb{R}$  represents an  $n$ -dimensional image. The convolution process produces a weighted average of a local neighborhood where the width of the neighborhood is a function of the scale of the

Gaussian operator. The scale of the Gaussian controls the resolution of the resultant image and thus the size of the structures that can be measured. A scale-space for an image  $I(\bar{x})$  of increasingly blurred images can be defined by  $L: \mathbb{R}^n \times \mathbb{R} \rightarrow \mathbb{R}^n$  and  $L(\bar{x}, \sigma) = G(\bar{x}, \sigma) \otimes I(\bar{x})$  where  $\otimes$  represents convolution.

The following is the calculation of the full-width half max (FWHM) of the Gaussian function.

$$G(x, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

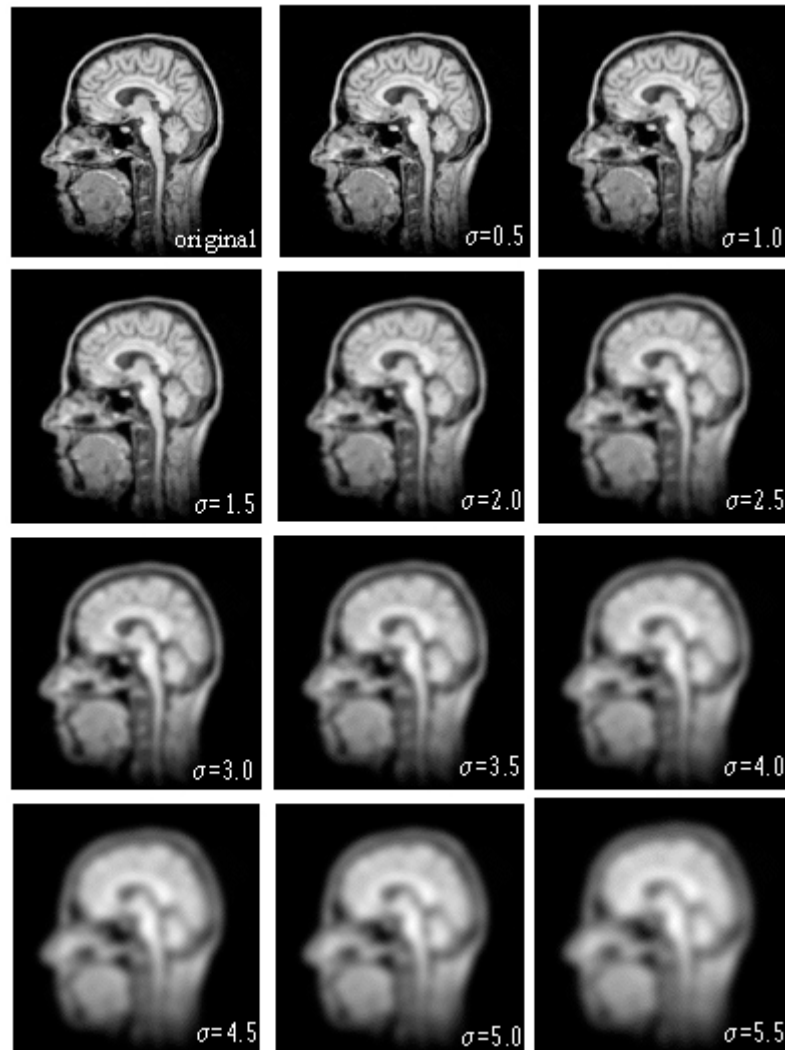
$$1n0.5 = \frac{1}{\sqrt{2\pi}\sigma} = 1n \left( \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x^2)}{2\sigma^2}} \right)$$

$$1n0.5 + 1n \left( \frac{1}{\sqrt{2\pi}\sigma} \right) = 1n \left( \frac{1}{\sqrt{2\pi}\sigma} \right) + 1n \left( e^{-\frac{(x^2)}{2\sigma^2}} \right)$$

$$-0.693 = -\frac{x^2}{2\sigma^2}$$

$$1.18\sigma = x$$

Figure 2 shows the Gaussian function. The dotted lines indicate the standard deviation of 1. The FWHM is indicated by the solid lines. For a Gaussian with a standard deviation equal to 1 ( $\sigma = 1$ ), the FWHM is 2.36.



**Figure 1. Gaussian Blur algorithm processing**

Figure 1 shows a Gaussian scale-space of the sagittal view of a MR head image. The original image is shown in the upper left ( $\sigma = 1$ ). Varying  $\sigma$  from 0.5 to 5.5 in steps of 0.5 blurs the other images (from left to right and top to bottom).

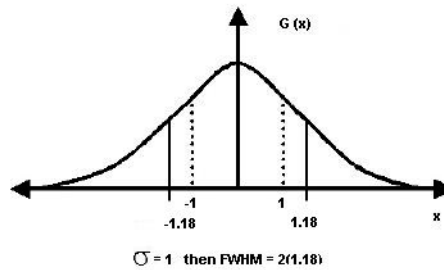


Figure 2. Gaussian function

Note that as the scale increases, small-scale features are suppressed. For example, at small scales the individual folds of the brain are quite defined, but, as the scale increases, these folds diffuse together resulting in the formation of a region one might define as the brain.

## IMAGE TYPES

You can apply this algorithm to all image data types except Complex and to 2D, 2.5D, 3D, and 4D images.

- By default, the resultant image is a float type image.
- By selecting Process each slide independently (2.5D) in the Gaussian Blur dialog box, you can achieve 2.5D blurring (each slice of the volume is blurred independently) of a 3D dataset (Figure 1).

## SPECIAL NOTES

None.



---

## REFERENCES

See the following references for more information about this algorithm:

J. J. Koenderink, “The Structure of Images,” *Biol Cybern* 50:363–370, 1984.

J. J. Koenderink and A. J. van Doorn, “Receptive Field Families,” *Biol Cybern* 63:291–298, 1990.

Tony Lindeberg, “Linear Scale-Space I: Basic Theory,” *Geometry-Driven Diffusion in Computer Vision*, Bart M. Ter Har Romeney, ed. (Dordrecht, The Netherlands: Kluwer Academic Publishers, 1994), pp. 1–38.

R. A. Young, “The Gaussian Derivative Model for Machine Vision: Visual Cortex Simulation,” *Journal of the Optical Society of America*, GMR-5323, 1986.

## Applying the Gaussian Blur algorithm

To run this algorithm, complete the following steps:

- 1** Select Algorithms > Filter > Gaussian Blur. The Gaussian Blur dialog box opens (Figure 3).
- 2** Complete the information in the dialog box.
- 3** Click OK.

The algorithm begins to run, and a pop-up window appears with the status. The following message appears: “Blurring Image.”

When the algorithm finishes running, the pop-up window closes, and the results appear in either a new window or replace the image to which the algorithm was applied.

<b>X Dimension</b>	Indicates the standard deviation (SD) of Gaussian in the X direction.
<b>Y Dimension</b>	Indicates the SD of Gaussian in the Y direction.
<b>Z Dimension</b>	Indicates the SD of Gaussian in the Z direction.
<b>Use image resolutions to normalize Z scale</b>	Normalizes the Gaussian to compensate for the difference if the voxel resolution is less between slices than the voxel resolution in-plane. This option is selected by default.
<b>Process each slice independently (2.5D)</b>	Blurs each slice of the dataset independently of adjacent slices.
<b>Process red channel</b>	Applies the algorithm to the red channel only.
<b>Process green channel</b>	Applies the algorithm to the green channel only.
<b>Process blue channel</b>	Applies the algorithm to the blue channel only.
<b>New image</b>	Shows the results of the algorithm in a new image window.
<b>Replace image</b>	Replaces the current active image with the results of the algorithm.
<b>Whole image</b>	Applies the algorithm to the whole image.
<b>VOI region(s)</b>	Applies the algorithm to the volumes (regions) delineated by the VOIs.
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

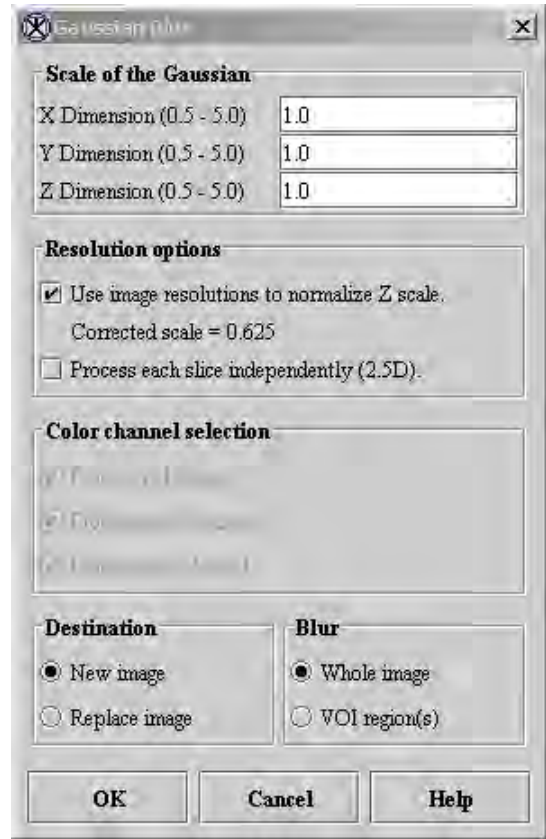


Figure 3. Gaussian Blur dialog box

## Filters (Spatial): Gradient Magnitude

Edge detection is the identification of meaningful discontinuities in gray level or color images. Edges are formed between two regions that have differing intensity values. This algorithm calculates the gradient magnitude of an image (or VOI of the image) using the first derivatives ( $G_x$ ,  $G_y$ , and  $G_z$  [3D]) of the Gaussian function at a user-defined scale sigma (standard deviation, or SD) and convolving it with image. The convolution of the first derivatives of the Gaussian with an image is a robust method of extracting edge information. By varying the SD, a scale-space of edges can easily be constructed.

### Background

Figure 1 shows the 1D signal of an object, the first derivative of the object, and, last, the gradient magnitude (defined later). Note that the maximum responses of the gradient magnitude result correspond to the edges of the object.

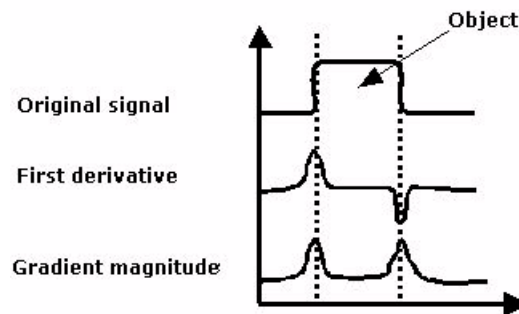


Figure 1. Edge detection using derivative operators

At location  $(x,y)$  of a 2D image, the gradient vector is

:

EQUATION 1

$$\nabla(x,y) = \begin{bmatrix} I_x \\ I_y \end{bmatrix} = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix}$$

and points in the direction of the maximum rate of change of the image at that point.

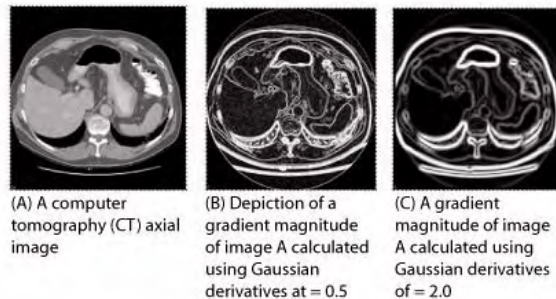
The gradient magnitude is defined as

$$mag(\nabla I) = \left[ I_x^2 + I_y^2 \right]^{1/2} \quad \text{EQUATION 2}$$

and is the maximum rate of increase of  $I(x,y)$  in the direction of  $\nabla I$ .

An efficient method of calculating the gradient magnitude is to analytically calculate the first derivative of Gaussian and convolve it with the image. The equation for this calculation follows

$$I_x(x,y,\sigma) = \frac{\partial}{\partial x} G(x,y,\sigma) \otimes I(x,y) \quad \text{EQUATION 3}$$



**Figure 2. Gradient Magnitude algorithm processing**

*A shows a computer tomography (CT) axial image of an abdomen. Image B depicts a gradient magnitude of image A calculated using Gaussian derivatives at = 0.5, and image C shows a gradient magnitude of image A calculated using Gaussian derivatives at = 2.0.*

## IMAGE TYPES

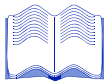
You can apply this algorithm to all image data types except complex and to 2D, 2.5D, 3D, and 4D images.

---

## SPECIAL NOTES

The following notes apply:

- The resultant image is, by default, a float type image;
- To achieve 2.5D blurring (each slide of the volume is processed independently), select Process each slide independently (2.5D) in the Gradient Magnitude dialog box.



## REFERENCES

Refer to the following references for more information:

Raphael C. Gonzalez and Richard E. Woods, *Digital Image Processing* (Boston: Addison-Wesley, 1992).

J. J. Koenderink, "The Structure of Images," *Biol Cybern* 50: 363–370, 1984.

Tony Lindeberg, "Linear Scale-Space I: Basic Theory," *Geometry-Driven Diffusion in Computer Vision*, Bart M. Ter Har Romeney, ed. (Dordrecht, The Netherlands: Kluwer Academic Publishers, 1994), pp. 1–38.

## Applying the Gradient Magnitude algorithm

To run this algorithm, complete the following steps:

- 1** Select Algorithms > Filter > Gradient Magnitude. The Gradient Magnitude dialog box opens (Figure 4).
- 2** Complete the information in the dialog box.
- 3** When complete, click OK. The algorithm begins to run.

A pop-up window appears with the status. The following message appears: "Calculating the Gradient Magnitude." When the algorithm finishes running, the pop-up window closes.

Depending on whether you selected New Image or Replace Image, the result appears in a new window or replaces the image to which the algorithm was applied.

<b>X Dimension</b>	Indicates the standard deviation (SD) of Gaussian in the X direction.
<b>Y Dimension</b>	Indicates the SD of Gaussian in the Y direction.
<b>Z Dimension</b>	Indicates the SD of Gaussian in the Z direction.
<b>Use image resolutions to normalize Z scale</b>	Normalizes the Gaussian to compensate for the difference if the voxel resolution is less between slices than the voxel resolution in-plane. This option is selected by default.
<b>Process each slice independently (2.5D)</b>	Blurs each slice of the dataset independently.
<b>Process red channel</b>	Applies algorithm to the red channel only.
<b>Process green channel</b>	Applies algorithm to the green channel only.
<b>Process blue channel</b>	Applies algorithm to the blue channel only.
<b>New image</b>	Shows the results of the algorithm in a new image window.
<b>Replace image</b>	Replaces the current active image with the results of the algorithm.
<b>Whole image</b>	Applies the algorithm to the whole image.
<b>VOI region(s)</b>	Applies the algorithm to the volumes (regions) delineated by the VOIs.
<b>OK</b>	Applies the Gradient Magnitude algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

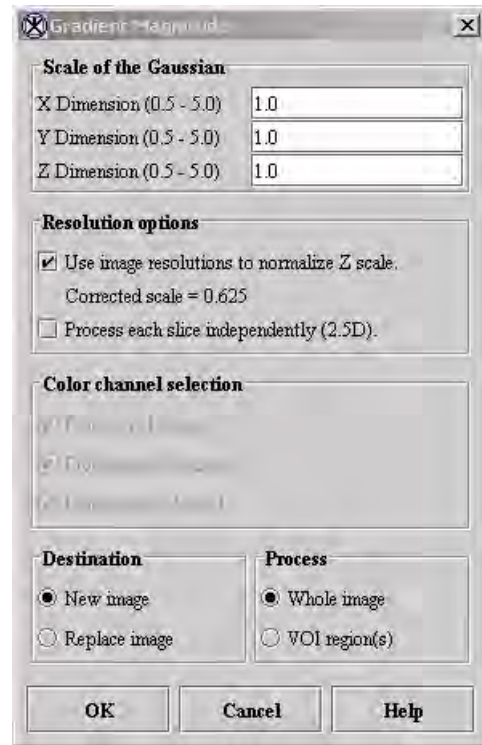


Figure 3. Gradient Magnitude dialog box

---

## Filters (Spatial): Haralick Texture

*The Haralick texture features are used for image classification. These features capture information about the patterns that emerge in the image texture. They are calculated by construction a co-occurrence matrix. Once the co-occurrence matrix has been constructed, calculations of the 12 features begin.*

---

For more information about the method, refer to the MIPAV web site {<http://mipav.cit.nih.gov/documentation/HTML%20Algorithms/FiltersSpatialHaralickTexture.htm>}

---

### Background

The algorithm is based on the material in the GLCM Texture Tutorial by Mryka Hall-Beyer at <http://www.fp.ucalgary.ca/mhallbey/tutorial.html>.

Haralick texture feature calculation can be divided into two parts

- 1 Construction of the co-occurrence matrices;
- 2 Calculation of the 12 texture features based on the co-occurrence matrices.

---

### OUTLINE OF THE ALGORITHM

#### **Construction of gray-level co-occurrence matrix (GLCM)**

The gray-level co-occurrence matrix, GLCM is used to describe the patterns of neighboring pixels in an image at a given offset distance,  $d$  usually chosen to be 1. In the calculation of the texture features, five such matrices are needed to describe different orientations or direction possibilities. The five direction possibilities are North-South, Northeast-Southwest, East-West, Southeast-Northwest, and spatially invariant. Spatially invariant is created by counting in the four directions and summing the counts.

The co-occurrence matrix is a square with dimension  $N_g$ , where  $N_g$  is the number of rescaled gray levels in the image.  $N_g$  has a default value = 32 and

is required to be between 8 and 64. There are 2 reasons for compressing the data:

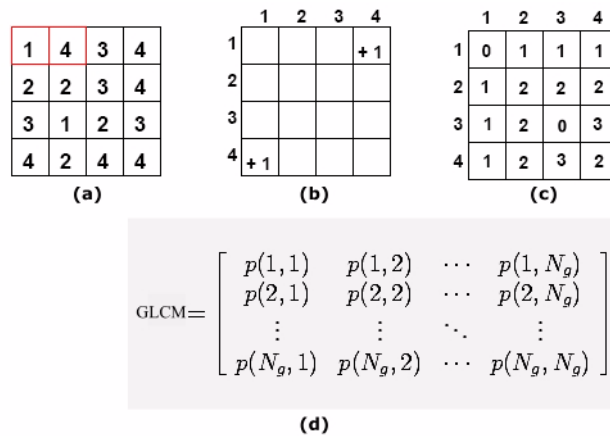
- 1** The matrix size would otherwise be excessively large. 16 bit data would give a matrix of size  $65536 \times 65536 = 4,294,967,296$  cells.
- 2** If all cells in even 8 bit  $256 \times 256$  data were used, there would be many cells filled with zeros (because that combination of gray levels simply does not occur). The GLCM approximates the joint probability distribution of the two pixels. Having many zero filled cells creates a very bad approximation. If the number of gray levels is reduced, the number of zeros is decreased, and the statistical validity is greatly improved.

Float, double, and ARGB\_FLOAT data will always be rescaled. Other data will be rescaled if (image maximum - image minimum) is more than 64.

Element [i,j] of the matrix is generated by counting the number of times a pixel with value i is adjacent to a pixel with value j. The adjacency is tested for 1 of 4 direction possibilities: east and west, northeast and southwest, northwest and southeast, or north and south. Always summing the counts from 2 opposite directions ensures that the GLCM is symmetric. Then, each element [i, j] is divided by the sum of all the elements to turn the matrix elements into probabilities.

Co-occurrence matrix texture considers the relation between two pixels at a time, which called the reference and neighbor pixel. The frequency of occurrence is counted over a square window which is placed over each pixel interior to a band of width (window size-1)/2 at the edge of the image, since results are only calculated for those pixels over which the entire square window can be placed. The values in this outer band have been left at zero.





**Figure 1. Construction of the east-west co-occurrence matrix for pixel distance equals 1. (a) – the original image with pixel image intensities showing as numbers; (b) shows the incremental stage that occurs when the outlined neighboring pixels in (a) are examined; (c) shows the final result of the co-occurrence matrix and (d) represents the matrix calculation**

## IMAGE TYPES

Any noncomplex 2D image.

## REFERENCES

Refer to the following references for more information:

This code is based on the material in the GLCM Texture Tutorial by Mryka Hall-Beyer, which can be found at <http://www.fp.ucalgary.ca/mhallbey/tutorial.htm>

“An Analysis of co-occurrence texture statistics as a function of grey level quantization” by David A. Clausi, *Can J. Remote Sensing*, Vol. 28, No. 1, 2002, pp. 45-62.

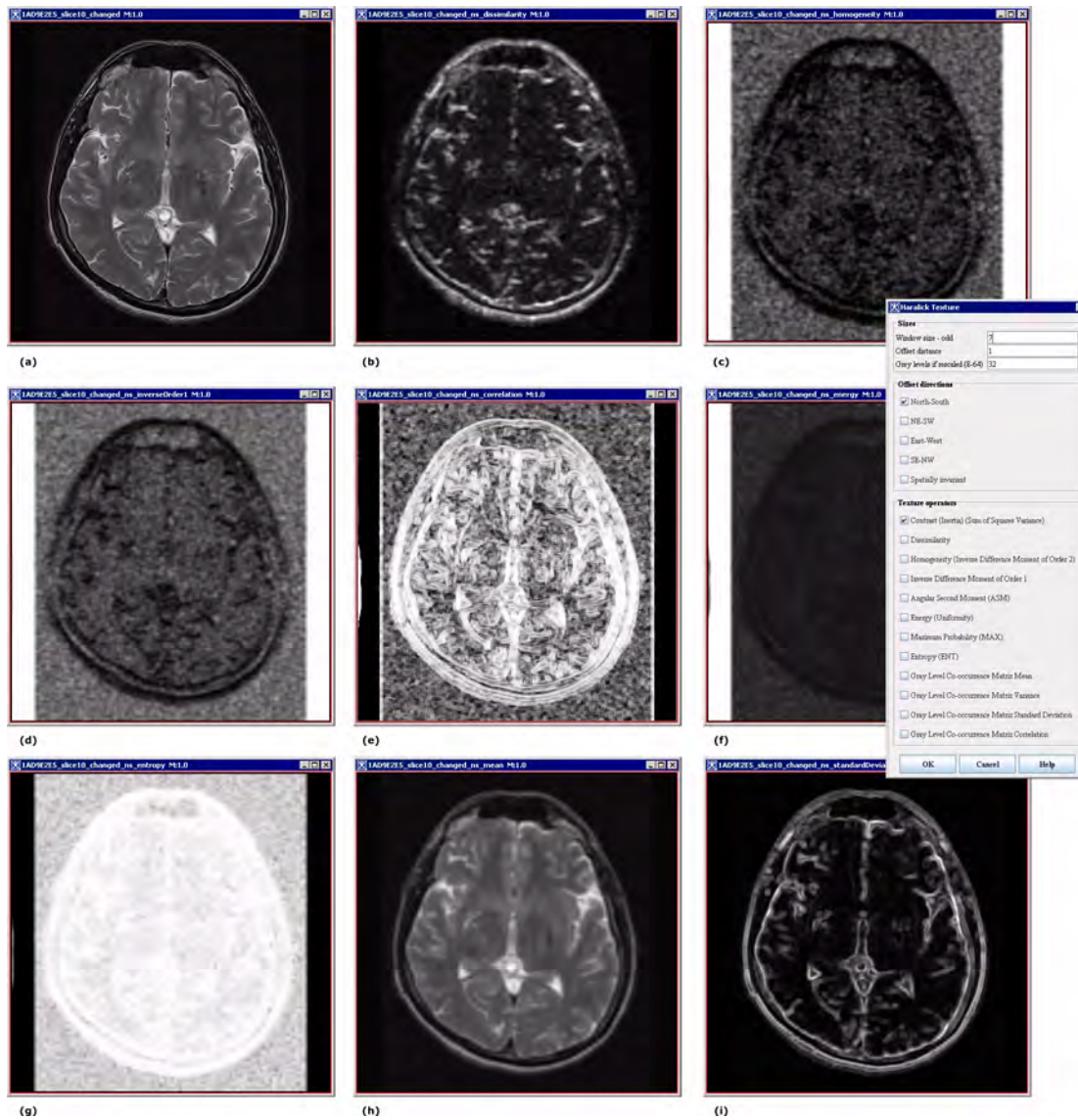


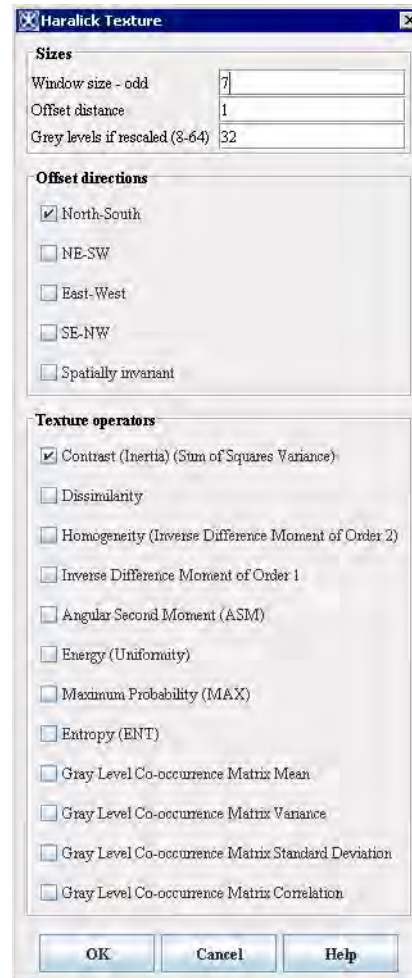
Figure 2. Applying the Haralick Texture algorithm: (a) – the original image, (b)–(i) result images

## Applying the algorithm

- 1 Open an image of interest.
- 2 Navigate to Algorithms > Filters Spatial>Haralick Texture.
- 3 The Haralick Texture dialog box appears, see Figure 3.

**4** Complete the dialog box and press OK. Use the information from section “Calculation of the 12 texture features” .  
The result images appear in new frames, see Figure 2.

## CALCULATION OF THE 12 TEXTURE FEATURES



**Figure 3. Haralick texture dialog box**

Texture features are calculated as weighted averages of the normalized co-occurrence matrix cell contents. Texture features are listed in the dialog box, see Figure 3.

Note: a weighted average multiplies each value to be used by a factor or a weight that expresses the relative importance of the value before summing all the weighted values and dividing by the number of values.

### Contrast

Calculation:

$$\sum_{i,j=0}^{N-1} P_{i,j} (i-j)^2$$

(EQ 1)

When  $i$  and  $j$  are equal, the cell is on the diagonal and  $(i-j)=0$ . These values represent pixels entirely similar to their neighbor, so they are given a weight of 0. If  $i$  and  $j$  differ by 1, there is a small similarity, and the weight is 1. If  $i$  and  $j$  differ by 2, contrast is increasing and the weight is 4. The weights continue to increase exponentially as  $(i-j)$  increases.

### Dissimilarity

While in Contrast calculation weights increasing exponentially (0, 1, 4, 9, etc.) as one moves steps away from the diagonal, the dissimilarity weights increase linearly (0, 1, 2, 3 etc.).

$$\sum_{i,j=0}^{N-1} P_{i,j} |i-j|$$

(EQ 2)

### Homogeneity (Inverse Difference Moment)

Homogeneity weights values by the inverse of the Contrast weight, with weights decreasing exponentially away from the diagonal.

(EQ 3)

$$\sum_{i,j=0}^{N-1} \frac{P_{i,j}}{1 + (i-j)^2}$$

### Angular Second Moment (ASM)

(EQ 4)

$$\sum_{i,j=0}^{N-1} P_{i,j}^2$$

### Energy (Uniformity)

(EQ 5)

$$\text{Energy} = \sqrt{ASM}$$

Note that ASM and Energy use each  $P_{ij}$  as a weight for itself. High values of ASM or Energy occur only when the texture patterns window is very orderly organized.

### Entropy

(EQ 6)

$$\sum_{i,j=0}^{N-1} P_{i,j} (-\ln P_{i,j})$$

Notes about the entropy measure:

- $P_{ij}$  is a probability measure is always between 0 and 1; therefore,  $\ln(P_{ij})$  will always be zero or negative;

- Smaller values of  $P_{ij}$  mean that the appearance of a given pixel combination is the less common;
- The smaller the value of  $P_{ij}$ , the larger is the absolute value of  $\ln(P_{ij})$ ;
- The (-1) multiplier in the entropy Equation 6 makes it positive;
- Therefore, the smaller the  $P_{ij}$  value, the greater the weight, and the greater the value of  $-(P_{ij} * \ln(P_{ij}))$ .

Since  $\ln(0)$  is undefined, assume that  $0 * \ln(0) = 0$  and  $1 * \ln(1) = 0$ . So the entropy is zero at  $p = 0$  and  $p = 1$  with a maximum value somewhere between. The maximum entropy occurs when the derivative of  $(p * \ln(p))$  with respect to  $p$  equals zero.  $p = 1/e$  gives a maximum entropy value of  $1/e$ , which is about 0.378.

### Gray Level Co-occurrence Matrix (GLCM) Mean

The GLCM Mean is expressed in terms of the gray level co-occurrence matrix. Consequently, the pixel value is weighted not by its frequency of occurrence by itself (as in common mean expression), but by the frequency of its occurrence in combination with a certain neighbor pixel value.

$$\mu_i = \sum_{i,j=0}^{N-1} i(P_{i,j}) \quad \mu_j = \sum_{i,j=0}^{N-1} j(P_{i,j}) \quad (\text{EQ 7})$$

Note that there are two equations here:

- The left hand equation calculates the mean based on the reference pixels,  $\mu_i$ .
- The right hand equation calculates the mean using the neighbor pixels,  $\mu_j$ .

For the symmetrical GLCM, where each pixel in the window is counted once as a reference and once as a neighbor, these two values are identical.

### Gray Level Co-occurrence Matrix (GLCM) Variance

Variance in texture performs the same task as does the common descriptive statistic called variance. It relies on the mean, and the dispersion around the mean, of cell values within the GLCM.

$$\sigma_i^2 = \sum_{i,j=0}^{N-1} P_{i,j} (i - \mu_i)^2 \quad \sigma_j^2 = \sum_{i,j=0}^{N-1} P_{i,j} (j - \mu_j)^2 \quad (\text{EQ 8})$$

### Gray Level Co-occurrence Matrix (GLCM) Standard Deviation

$$\sigma_i = \sqrt{\sigma_i^2} \quad \sigma_j = \sqrt{\sigma_j^2} \quad (\text{EQ 9})$$

### Gray Level Co-occurrence Matrix (GLCM) Correlation

Correlation between pixels assumes that there are predictable and linear relationships between the pixel intensities, which can be expressed by the correlation equation.

$$\sum_{i,j=0}^{N-1} P_{i,j} \left[ \frac{(i - \mu_i)(j - \mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}} \right] \quad (\text{EQ 10})$$

### Properties of Correlation

- Single objects usually have a higher correlation value within them than between adjacent objects.
- Pixels are usually more highly correlated with pixels nearby than with more distant pixels.
- Therefore, smaller window sizes will usually have a higher correlation value than larger windows. For that reason, it is useful to use the following approach in calculating Correlation – if Correlation is calculated for successively larger window sizes, the size at which the Correlation value declines may be taken as the size of definable objects within an image. Note that this works only if all objects in the image are about the same size.

Note that Correlation uses quite a different approach in calculation than the other texture measures described above. As a result, it provides different information and can often be used profitably in combination with other texture measures. If all the pixels have identical values, then the denominator variances are zero and the correlation equation would give an undefined result. If the variance is zero, then the equation is not used, and the correlation is set to 1 to reflect identical pixels.



## Filters (Spatial): Laplacian

*Edge detection is the identification of meaningful discontinuities in gray level or color images. Edges are formed between two regions that have differing intensity values. This algorithm calculates the laplacian of an image (or VOI of the image) using the second derivatives ( $G_{xx}$ ,  $G_{yy}$ , and  $G_{zz}$  [3D]) of the Gaussian function at a user-defined scale sigma [standard deviation (SD)] and convolving it with image. The convolution of the second derivatives of the Gaussian with an image is a robust method of extracting edge information. By varying the SD, a scale-space of edges can easily be constructed.*

For more information about the algorithm, refer to MIPAV Volume 2 User Guide on the MIPAV web site: {<http://mipav.cit.nih.gov/>}

### Background

Figure 1 shows the 1D signal of an object. The first derivative of the object is shown next. Last, the Laplacian (defined later) is shown. Note that the zero-crossing of the second derivative corresponds to the edges of the object.

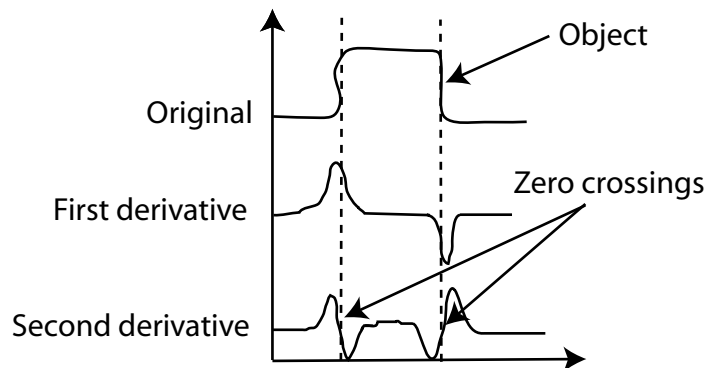


Figure 1. Edge detection using laplacian operators

The laplacian function is defined as:

$$\nabla^2 G(x,y) = \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2}$$

An efficient method of calculating the laplacian is to analytically calculate the second derivatives of Gaussian and convolve the sum with the image.

The equation for this calculation follows:

$$\nabla^2 I(x,y) = \left( \frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2} \right) \otimes I(x,y)$$



Figure 2. (A) Original MR image; (B) laplacian results; and (C) extraction of the zero crossing of the laplacian (object edges)

## IMAGE TYPES

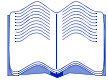
You can apply this algorithm to all image data types except complex and to 2D, 2.5D, and 3D images.

## SPECIAL NOTES

The resulting image is, by default, a float type image.

To achieve 2.5D blurring (each slice of the volume is processed

independently) of a 3D dataset, select Process each slice independently (2.5D) in the Laplacian dialog box (Figure 1).



---

## REFERENCES

Refer to the following references for more information about this algorithm:

Tony Lindeberg, “Linear Scale-Space I: Basic Theory,” *Geometry-Driven Diffusion in Computer Vision*, Bart M. Ter Har Romeney, ed. (Dordrecht, The Netherlands: Kluwer Academic Publishers, 1994), pp. 1–38.

J. J. Koenderink, “The Structure of Images,” *Biol Cybern* 50:363–370, 1984.

Raphael C. Gonzalez and Richard E. Woods, *Digital Image Processing* (Boston: Addison-Wesley, 1992).

## Applying the Laplacian algorithm

To run this algorithm, complete the following steps:

- 1** Select Algorithms > Filter > Laplacian. The Laplacian dialog box opens (Figure 3).
- 2** Complete the fields in the dialog box.
- 3** When complete, click OK. The algorithm begins to run.

A pop-up window appears with the status. The following message appears: “Calculating the Laplacian.”

When the algorithm finishes, the pop-up window closes. Depending on whether you selected New Image or Replace Image, the results either appear in a new window or replace the image to which the algorithm was applied.

<b>X Dimension</b>	Specifies the standard deviation (SD) of Gaussian in the X direction.
<b>Y Dimension</b>	Specifies the SD of Gaussian in the Y direction.
<b>Z Dimension</b>	Specifies the SD of Gaussian in the Z direction.
<b>Amplification factor</b>	Applies, by default, the classical laplacian factor of 1.0; values greater than 1.0 and less than 1.2 enable the laplacian to act as a high-pass, or high-boost, filter.
<b>Use image resolutions to normalize Z scale</b>	Normalizes the Gaussian to compensate for the difference if the voxel resolution is less than the voxel resolution inplane.
<b>Process each edge independently (2.5D)</b>	Blurs each slice of the dataset independently.
<b>Output edge image</b>	Produces a binary image that represents the edges defined by the 0 crossings of the laplacian
<b>Threshold edge noise between &lt;_&gt; and &lt;_&gt;</b>	Limits the threshold edge noise to the range that you specify between the 0 crossings of the laplacian; the default range is -10 to 10.
<b>New image</b>	Shows the results of the algorithm in a new image window.
<b>Replace image</b>	Replaces the current active image with the results of the algorithm.
<b>Whole image</b>	Applies the algorithm to the whole image.
<b>VOI region(s)</b>	Applies the algorithm to the volumes (regions) delineated by the VOIs.
<b>OK</b>	Applies the laplacian algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

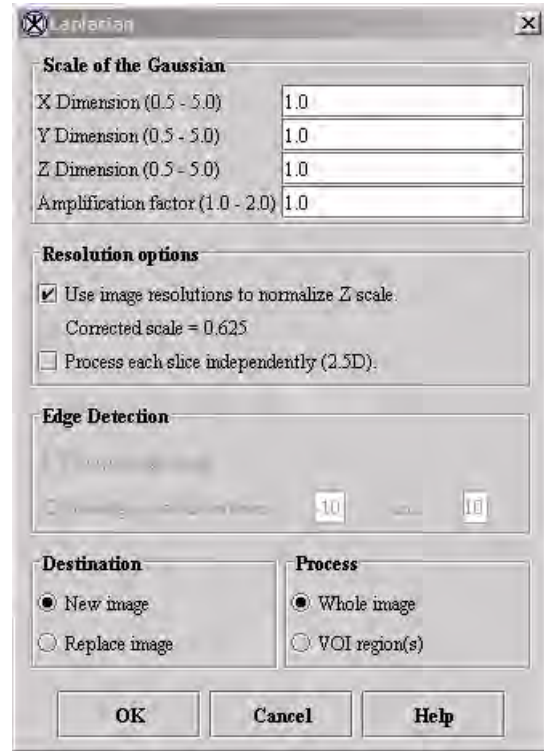


Figure 3. Laplacian dialog box

---

## Filters (Spatial): Local Normalization TBD

TBD.

### Background

TBD. BEFORE AND AFTER GRAPHIC (WHAT HAPPENS WHEN ALGORITHM IS RUN)?

Figure 1. Local Normalization algorithm processing

---

### IMAGE TYPES

TBD.

---

### SPECIAL NOTES

TBD.

---

### REFERENCES

See the following references for more information about this algorithm:

TBD.

### Applying the Local Normalization algorithm

To run this algorithm, complete the following steps:

- 1 [tbd]
- 2 [tbd]
- 3 [tbd]

## Filters (Spatial): Mean

This algorithm provides a simple way of reducing noise either in an entire image or in a delineated VOI in the image where

Mean = sum of all pixels in the kernel / total number of pixels in the kernel

This is one method of low-pass filtering, which is sometimes referred to as *neighborhood averaging*.

### Background

In applying this algorithm, MIPAV first defines a region, either in the whole image or in the delineated VOI, over which it replaces a target pixel value with the average value over the kernel of pixels in which the target pixel is at the center. That is, for the target pixel:

$$mean = \frac{1}{s} \sum_{i=1}^{i=s} I_i \cdot k_i$$

where  $k$  = kernel,  $s$  = kernel size, and  $I$  = image.

For *slice averaging* (Figure 1 and Figure 2), the kernels are odd-length squares (for example, 3 x 3). The processing time increases with the square of the kernel length. For *volume averaging* (Figure 2), the kernels are odd-length cubes (such as 5 x 5 x 5), and the processing time increases with the cube of the kernel length. Refer to Figure 2 for examples of slice and volume averaging.

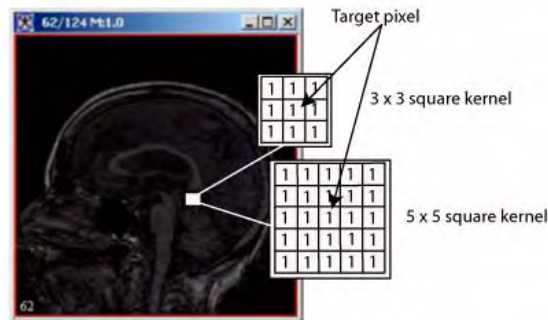
Note that the term *average* does not always represent the mean. In fact, there are three types of average: the commonly used mean, the median, and the infrequently used mode. The *mean* is defined as the sum of the values in a group divided by the number of values in the group. The *median* value is the value that has the same number of values greater and less than it. The *mode* is the value with the greatest number of occurrences.

If part of the neighborhood kernel falls outside the image when the pixel is on or near the boundary, MIPAV only uses that portion of the kernel inside the image.



**Example:** Consider the upper right corner pixel in a 2D image if a 3 x 3 kernel is used. In this case, when the algorithm is applied, MIPAV uses only the target pixel and its lower, left, and diagonal left lower neighbors. It then divides the sum of these four pixels by 4.

If you select the New image option, MIPAV places the resulting image in a new image window. If you select the Replace image option, MIPAV returns the changed picture to the same image window. If you delineate a VOI on the image and select VOI regions, then MIPAV replaces the original value of a pixel inside the VOI with the mean value of the kernel that contains the pixel at its center. The pixel values outside the VOI remain unchanged.



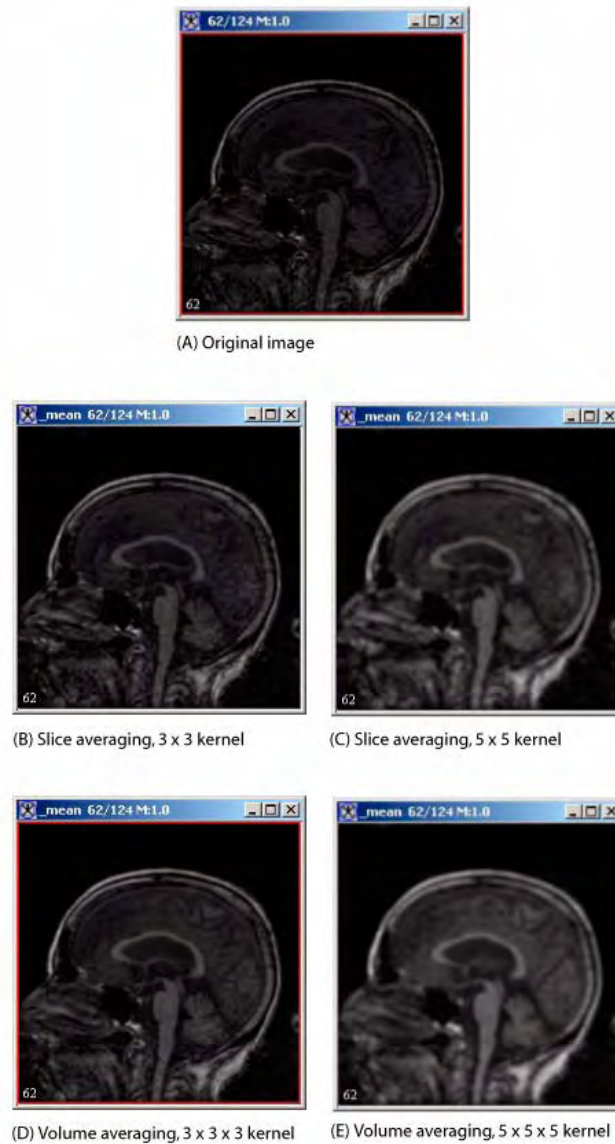
**Figure 1. Slice averaging**

*In slice averaging, MIPAV defines a region and replaces the target pixel value with the average value over the kernel of pixels in which the target pixel is at the center.*

Because 2D images contain only one slice, the only option possible under Kernel Thickness in the Mean Filter dialog box is slice averaging—the Apply slice kernel option. The Apply volume kernel option, which performs volume averaging, is dimmed. However, both slice averaging and volume averaging are available for 3D images. Although slice averaging is the default value, users can choose volume averaging instead by selecting the Apply volume kernel option.

For slice averaging, the default kernel size is 3 x 3, but the following kernel sizes are available: 5 x 5, 7 x 7, 9 x 9, or 11 x 11. For volume averaging, the default kernel size is 3 x 3 x 3. However, you can select one of the following sizes instead: 5 x 5 x 5, 7 x 7 x 7, 9 x 9 x 9, or 11 x 11 x 11.

For color images the default is to obtain the means of the red, green, and



**Figure 2. Image processing using the Mean algorithm**

*This figure depicts slice averaging and volume averaging applied to the whole image with various size kernels as noted.*

blue channels. However, if users clear the Red channel, Green channel, or Blue channel check boxes, MIPAV only performs mean averaging on the color channels that are selected.



---

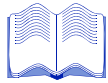
## IMAGE TYPES

You can apply this algorithm to all image data types and to 2D, 2.5D, and 3D images.

---

## SPECIAL NOTES

None.



---

## REFERENCES

Refer to the following reference for more information about this algorithm:

Raphael C. Gonzalez and Richard E. Woods. *Digital Image Processing, Second Edition* (Upper Saddle River, New Jersey: Prentice-Hall, Inc., 2002), pp. 116–123.

## Applying the Mean algorithm

To run this algorithm, complete the following steps:

- 1** Open an image. As an option, delineate a VOI.
- 2** Select Algorithms > Filter (spatial) > Mean. The Mean Filter dialog box (Figure 3) opens.
- 3** Complete the information in the dialog box.
- 4** Click OK.

The algorithm begins to run, and a pop-up window appears with the status. The following message appears: “Filtering image.”

When the algorithm finishes running, the pop-up window closes, and the results either appear in a new window or replace the image to which the algorithm was applied.

<b>Kernel size</b>	Specifies the size of the $k \times k$ neighborhood that extends in all directions by $(k - 1)/2$ from the center pixel.
<b>Red channel</b>	Applies the algorithm to the red channel.
<b>Green channel</b>	Applies the algorithm to the green channel.
<b>Blue channel</b>	Applies the algorithm to the blue channel.
<b>New image</b>	Shows the results of the algorithm in a new image window.
<b>Replace image</b>	Replaces the current active image with the newly calculated mean image.
<b>Whole image</b>	Applies the algorithm to the whole image.
<b>VOI regions</b>	Applies the algorithm to the volumes (regions) delineated by the VOIs.
<b>Apply slice kernel</b>	Applies the kernel to only the specific slice in the dataset
<b>Apply volume kernel</b>	Applies the kernel to all of the slices—that is, the entire volume—in the dataset.
<b>OK</b>	Applies the mean algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

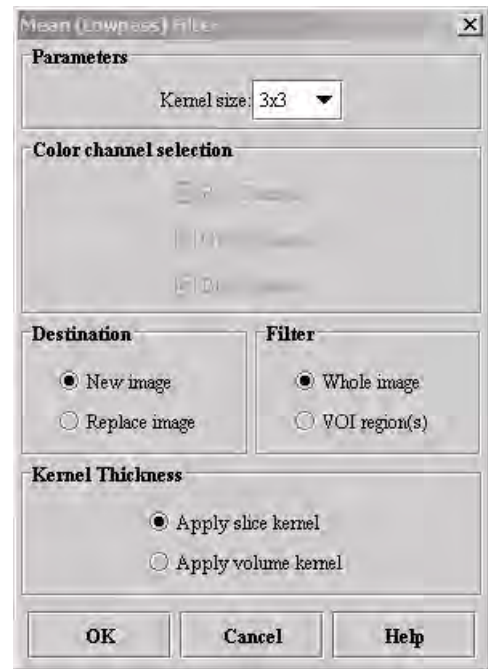


Figure 3. Mean Filter dialog box

---

## Filters (Spatial): Median

*The Median filter algorithm replaces a target pixel's value with the median value of the neighboring pixels. The designation of neighbor is determined by the size and shape of the kernel. A kernel is much like a mask; the pixels in the masked area are part of the neighborhood. The kernel's size and shape determines the number of pixel values used to calculate the median. You can adjust this algorithm's parameters to change the kernel's size and shape, apply selective filtering, and repetitively apply the selected filtering for any portion of the image you select.*

The median filter algorithm is most useful for removing *shot noise*, which is noise that appears as random spots in an image. Shot noise can be caused by dust or a bad detector in image acquisition or by errors in data transfer (Lyon, Russ). Note that because there is no inverse function for the median filter algorithm, the original image cannot be reconstructed. This algorithm is also called a *rank-type filter*, because the list of values must be sorted before a useful value can be found.

---

For more information about the algorithm, refer to MIPAV Volume 2 User Guide on the MIPAV web site: {<http://mipav.cit.nih.gov/>}

---

## Background

When you apply the Median filter algorithm, the target pixel is identified and the values of the surrounding neighborhood are collected. Next, the system arranges the values of the target and neighborhood pixels in a list and sorts them; then it replaces the value of the target pixel with the list's median value (Figure 1). Given that the list has  $n$  items and  $n$  is an even number, the median is the item at  $n/2$ . When  $n$  is odd, the median is defined as an average of the items at position  $(n - 1)/2$  and at position  $(n + 1)/2$ .

---

**Example:** You cannot apply this algorithm if the target pixel is near the edge of a window. This algorithm is applied to all pixels in the kernel. If the target pixel, which is usually at the center of the kernel, is near the edge of the image window, the kernel probably contains undefined regions because the kernel is always symmetrical. Since the median filter algorithm cannot be calculated

reliably if the undefined regional data is present, if the target pixel is at the edge of the image, it is not processed.

Neighborhood pixels are determined by the shape and size of the kernel. (The selection of pixels using the kernel is sometimes referred to as the *sliding window* approach.) The most common kernel shapes are the square and cross. Square-shaped kernels are defined as all pixels in the  $k(k)$  region surrounding the target pixel, where  $k$  refers to the window size. Cross-shaped kernels are defined as the central row and column of the  $k(k)$  region. The kernel is usually centered on the target pixel and is symmetrical; pixels chosen are symmetrically located about the target pixel. Thus, pixels on opposite sides are equally weighted. The kernel varies in size, although commonly, a square kernel can range from 3 to 11 pixels per side. Other kernel shapes and sizes could be selected, but choosing the region of pixels to include always involves juggling the elimination of noise and increasing fuzziness and posterization of an image.

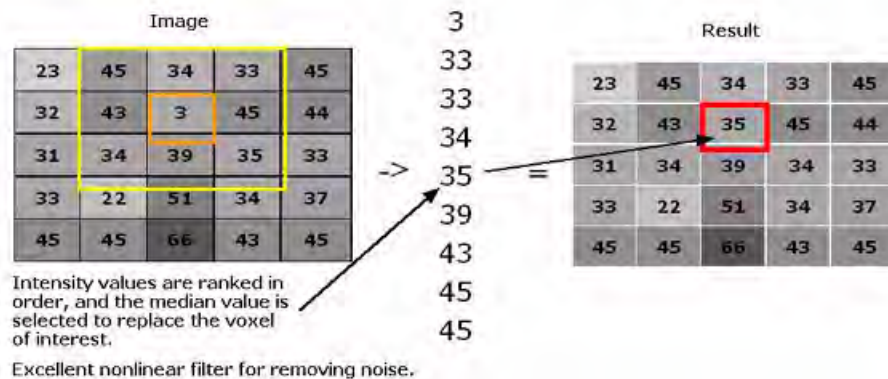
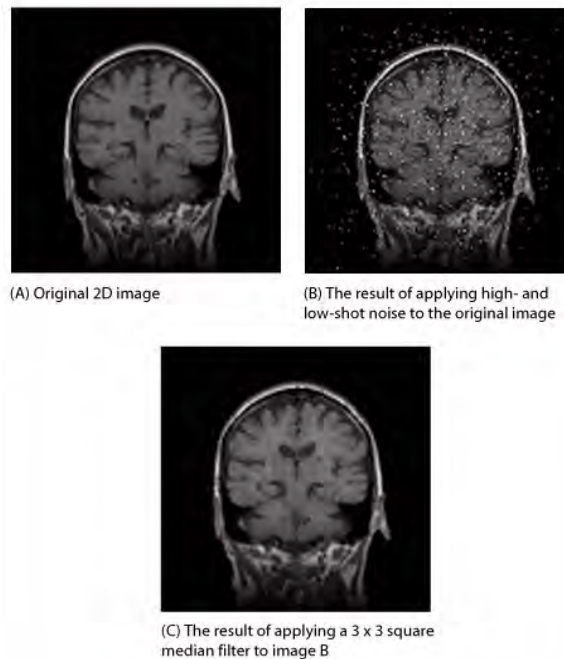


Figure 1. An example of median filter processing using a square 3 x 3 kernel

Pixel values are sorted in increasing order. Any sorting method is acceptable, although, because the number of pixels can be large, a fast sorting algorithm is preferable. The best sorting algorithm is one that can efficiently accommodate both small and large lists. Generally, when considering the number of items to sort, the numbers can range from 5 (in a 3 x 3 cross) to 121 (in an 11 x 11 square). This algorithm implements the Shell-sort because it operates very efficiently for the sizes of the lists expected.

As the kernel size increases, computation also increases. Because the median filter algorithm is already computationally intensive, this algorithm incorporates adaptive filtering to improve the responsiveness of the filter and to reduce the number of computations. Rather than using one specific method, adaptive filtering involves any number of novel approaches. A simple approach is to verify whether the target pixel is within some percentage of the median. This verifies that the pixel value is not outside the range of expected values of a pixel. For example, shot noise is generally a value that much larger or smaller than the average of the surrounding pixel values. Figure 2 illustrates how the median filter algorithm is used to remove shot noise.



**Figure 2. Removing shot noise with the Median Filter algorithm**

Filtering a color image is very similar to filtering a monochrome image. However, each color must be filtered separately because each color is a part of its own band or channel; each channel forms a monochrome image. This means that all red pixels are filtered, then all green, and then all blue, but never red and green pixels at the same time. Values in the alpha channel, which indicate pixel opacity levels, are not filtered.

---

## REFERENCES

See the following references for more information about this algorithm:

Douglas A. Lyon, *Image Processing in Java*, Upper Saddle River, New Jersey: Prentice-Hall, 1999, pp: 201–214.

John C. Russ, *The Image Processing Handbook*, 3rd Edition Boca Raton, Florida: CRC Press, 1999, pp: 174–182.

Bernde Jahne, *Practical Handbook on Image Processing for Scientific Applications*, Boca Raton, Florida: CRC Press, 1997, pp: 331, 332.

---

## IMAGE TYPES

You can apply this algorithm to color and gray-scale 2D, 2.5D, and 3D images. However, you cannot apply this algorithm to complex images.

---

## SPECIAL NOTES

The following special notes apply:

- When this algorithm is applied to 3D datasets, you can apply the filter to either a single slice or volume kernel. If you choose to apply it to a volume kernel, MIPAV finds the median in a list of values from pixels in all directions along the  $x$ -,  $y$ -, and  $z$ -axes. Note that the kernel-size must be smaller than the number of slices in the image.
- The resultant image is, by default, a float image.
- An outlier detector is available for use in this algorithm. The outlier detector replaces the target pixel only if its magnitude exceeds the mean of the list by a user-inputted fraction of the standard deviation.
- Currently the algorithm allows square and cross kernel shapes for 2D images and cube and axis kernel shapes for 3D images. Future plans include the addition of other kernel shapes.
- Color images may be filtered across any combination of the color channels (red, green, or blue). Alpha values—those values relating to how opaque a particular pixel is—are unaffected.

---

## Applying the Median Filter algorithm

To run this algorithm, complete the following steps:

- 1** Select Algorithms > Filter > Median Filter. The Median Filter dialog box (Figure 3) appears.
- 2** Complete the fields in the dialog box. Keep the following in mind:
  - To filter a VOI region only, select the VOI and then select VOI Region(s).
  - Pixels that are too close to the edge of an image are not processed.
  - To apply a median filter to a color image, select the channels to filter. By default, MIPAV filters all three channels (red, green, and blue).
- 3** Click OK when complete. The algorithm begins to run.

A pop-up window appears with the status, and the following message appears: “Filtering Image (Pass 1 of 1).” (Depending on what you entered in Number of iterations on the Median Filter dialog box, the number of passes varies.) When the algorithm finishes running, the pop-up window closes.

Depending on whether New image or Replace image was selected, results appear in a new window or replace the image to which the algorithm was applied.

<b>Number of iterations</b>	Indicates the number of times to filter the image with a median filter of the selected deviation, kernel size, and kernel shape.	
<b>Maximum standard deviation</b>	The maximum allowable magnitude of the pixel value as a function of the standard deviation. The system replaces any pixel whose value is outside this function of the standard deviation from the mean of the neighborhood (as selected by the kernel-size and shape) by the median of the neighborhood. A value of 0.0 is equivalent to replacing every pixel with the median of its neighborhood. A value of 1.0 limits the filtering of the image. The system processes only target pixels whose values are greater than 1.0 standard deviation from its neighbors.	
<b>Kernel size</b>	A $k \times k$ region selects a neighborhood that extends in all directions by $(k-1)/2$ , placing the target pixel in the center of the kernel.	
<b>Kernel shape</b>	Selects which pixels in the kernel size box are included as a part of the median's neighborhood.	
<b>Kernel size adaptively changes</b>	This checkbox indicates whether median filter should be run via adaptive filtering. Adaptive filtering has 3 main purposes: <ul style="list-style-type: none"> <li>1. To remove salt-and-pepper (impulse) noise;</li> <li>2. To provide smoothing of other noise that may not be impulsive;</li> <li>3. And to reduce distortion, such as excessive thinning or thickening of object borders.</li> </ul>	
<b>Maximum Kernel Size</b>	This is the maximum kernel size that the window size can get increased to under the algorithm.	
<b>Red channel</b>	Filters the red channel (for use with RGB images).	
<b>Green channel</b>	Filters the green channel (for use with RGB images).	
<b>Blue channel</b>	Filters the blue change (for use with RGB images).	

Figure 3. Median Filter dialog box



<b>New image</b>	Indicates where the results of the algorithm appear. If you select this option, the results appear in a new image window.
<b>Replace image</b>	Indicates where the results of the algorithm appear. If you select this option, the results replace the current active image.
<b>Whole image</b>	Applies the algorithm to the whole image.
<b>VOI region(s)</b>	Applies the algorithm to the volumes (regions) delineated by the VOIs.
<b>Apply slice kernel</b>	Filters each slice of the image independently of other slices, finding neighboring pixels from the slice.
<b>Apply volume kernel</b>	Gathers pixel values from the slice as well as from neighboring slices.
<b>OK</b>	Applies the Median algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

**Figure 3. Median Filter dialog box (continued)**

---

## Filters (Spatial): Mode

Mode filtering is a nonlinear spatial filtering technique that replaces a target pixel with the mode of neighboring pixels. The *mode* of a set of values is the value that occurs most frequently within the set. If all values are unique—that is, all numbers occur only once in the set—the value of the target pixel stays the same. Mode filtering is similar in philosophy to median filtering.

Mode filtering is currently thought to be useful in filtering a segmented image to eliminate small clusters of pixels labeled one value that are completely within a large cluster of pixels of a different value. Mode filtering is currently being investigated in improving a C-means classification of brain images, which were first segmented using the brain extraction tool (BET). Mode filtering can be applied to grayscale images to remove some noise; however, a 3 x 3 x 3 cube kernel significantly blurs the image.

### Background

When you apply the Mode algorithm, MIPAV identifies the target pixel and collects the values of the surrounding neighborhood. Next, it arranges the values of the target and neighborhood pixels in a list and sorts them. The value of the target pixel is replaced with the mode of the numbers in the list. Determining the mode of a sorted list of numbers is done in a single pass through the list by incrementing a count variable each time adjacent values are the same and maintaining the maximum count along with the corresponding value.

Neighborhood pixels are determined by the shape and size of the kernel. (The selection of pixels using the kernel is sometimes referred to as the *sliding window approach*.) The most common kernel shapes are the square and cross.

Square-shaped kernels are defined as all pixels in the  $k(k)$  region surrounding the target pixel, where  $k$  refers to the window size. Cross-shaped kernels are defined as the central row and column of the  $k(k)$  region.

The kernel is usually centered on the target pixel and is symmetrical; pixels chosen are symmetrically located about the target pixel. Thus, pixels on opposite sides are equally weighted. The kernel varies in size, although commonly a square kernel can range from 3 to 11 pixels per side. Other

kernel shapes and sizes could be selected, but choosing the region of pixels to include always involves juggling the elimination of noise and increasing fuzziness and posterization of an image.

Pixel values are sorted in increasing order. Any sorting method is acceptable, although, because the number of pixels can be large, a fast-sorting algorithm is preferable. The best sorting algorithm is one that can efficiently accommodate both small and large lists. Generally, when considering the number of items to sort, the numbers can range from 5 (in a 3 x 3 cross) to 121 (in an 11 x 11 square). This algorithm implements the Shell-sort because it operates very efficiently for the sizes of the lists expected.

The left image in Figure 1 shows a segmented image that results after applying the C-means classification algorithm to a brain image produced with the brain extraction tool. The image on the right shows the result of applying the Mode filtering algorithm with a 3 x 3 x 3 local neighborhood to the image on the left. Clearly, the image on the right has fewer small isolated classes than the image on the left.



Figure 1. Examples of (A) a segmented image and (B) the results of applying the Mode algorithm to the image

The Mode algorithm processes all pixels in the input image. The algorithm uses a symmetric neighborhood definition around the target pixel, resulting in border pixels, which require special processing. Border pixels correspond to those pixels where the neighborhood of a target pixel is incomplete. Incomplete neighborhoods occur:

- *In-plane*—In the corners of the image plane
- *Through-plane*—On the first and last (few) image planes

In-plane border pixels are not filtered and the value assigned to the output pixel is simply the corresponding input value. Through-plane border pixels are filtered by duplicating the values of corresponding pixels on the target pixel plane.

---

## IMAGE TYPES

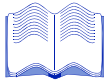
You can apply this algorithm to 2D, 2.5D, and 3D images. Since the algorithm counts the number of pixels with the same value, it only allows the following list of data types.

- BYTE and UBBYTE
- SHORT and USHORT
- INTEGER and UINTEGER

---

## NOTES

The resulting image is, by default, the same type as the input image.



## REFERENCES

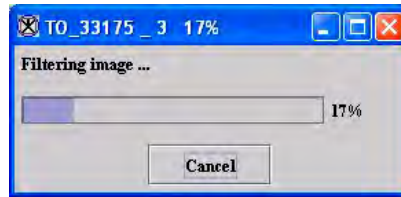
This algorithm was developed by members of the MIPAV team. Mode filtering is similar to median filtering, and to our knowledge there is not a significant amount of literature about the topic.

## Applying the Mode algorithm

To run this algorithm, complete the following steps:

- 1** Open an image.
- 2** Select Algorithms > Filters (Spatial) > Mode. The Mode Filter dialog box (Figure 2) opens.
- 3** Complete the fields in the dialog box.
- 4** Click OK when complete. The algorithm begins to run.

A pop-up window appears with the status.



When the algorithm finishes running, the pop-up window closes.

Depending on whether New image or Replace image was selected, results appear in a new window or replace the image to which the algorithm was applied.

<b>Kernel size</b>	Specifies the size of the local neighborhood when performing mode filtering. A $k \times k$ region selects a neighborhood that extends in all directions by $(k - 1)/2$ placing the target pixel in the center of the kernel. Kernels for 2D images are odd valued and include $3 \times 3$ , $5 \times 5$ , $7 \times 7$ , and so on.	
<b>Kernel shape</b>	Selects which pixels in the kernel size box are included as a part of the mode's neighborhood.	
<b>New image</b>	Indicates where the results of the algorithm appear. If you select this option, the results appear in a new image window.	
<b>Replace image</b>	Indicates where the results of the algorithm appear. If you select this option, the results replace the current active image.	
<b>Whole image</b>	Applies the algorithm to the whole image.	
<b>VOI region(s)</b>	Applies the algorithm to the volumes (regions) delineated by the VOIs.	
<b>Apply slice kernel</b>	Filters each slice of the image independently of other slices, finding neighboring pixels from the slice.	
<b>Apply volume kernel</b>	Gathers pixel values from the slice as well as from neighboring slices.	

Figure 2. Mode Filter dialog box

---

<b>OK</b>	Applies the Mode filtering algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

---

**Figure 2. Mode Filter dialog box (continued)**

## Filters (Spatial): Nonlinear Noise Reduction

*This algorithm provides a part of the SUSAN (Smallest Univalve Segment Assimilating Nucleus) low-level image processing program.*

For information about SUSAN, see <http://www.fmrib.ox.ac.uk/~steve>.

SUSAN noise reduction uses nonlinear filtering to reduce noise in an image while preserving both the underlying structure and the edges and corners. It does this by averaging a voxel only with local voxels that have similar intensity.

---

For more information about the algorithm, refer to the MIPAV web site: {<http://mipav.cit.nih.gov/documentation/HTML%20Algorithms/FiltersSpatialNonlinearNoiseReduction.html>}.

---

### Background

The SUSAN program compares the brightness of each pixel within a mask with the brightness of the mask's nucleus. It then defines an area of the mask, called the *univalve segment assimilating nucleus* (USAN), that has the same or similar brightness as the nucleus. The USAN filter takes an average of the pixels in the USAN and then uses a Gaussian in both the brightness and spatial domains.

Table 1 explains the possible responses that the USAN filter provides if it cannot determine the neighborhood and based on whether you selected the option of using the median.

**Table 1. Possible responses of the SUSAN filter**

Use median	USAN area	The SUSAN filter . . .
Selected	Zero	Uses the median of the pixel's 8 nearest neighbors in 2D images or 26 nearest neighbors in 3D images to estimate the pixel's correct value.
Selected	Nonzero	Uses sums taken over the local neighborhood, not including the center pixel itself, to estimate the pixel's correct value. This allows good reduction of impulse noise.

**Table 1. Possible responses of the SUSAN filter**

Not selected	Nonzero	Uses sums taken over the local neighborhood including the center pixel to estimate the pixel's correct value.
Not selected	Zero	Leaves the pixel unchanged.

In 2D images, the algorithm is based on the following equation:

$$R(x,y) = \frac{\sum_{i,j} I(x+i,y+j) \cdot S \left( \frac{-i \cdot i}{(2 \cdot sdx \cdot sdx)} - \frac{j \cdot j}{(2 \cdot sdy \cdot sdy)} \right) \cdot B \left( \frac{(I(x+i,y+j) - I(x,y))}{T^2} \right)}{\sum_{i,j} S \left( \frac{-i \cdot i}{(2 \cdot sdx \cdot sdx)} - \frac{j \cdot j}{(2 \cdot sdy \cdot sdy)} \right) \cdot B \left( \frac{(I(x+i,y+j) - I(x,y))}{T^2} \right)}$$

where

$R$  = replacement intensity

$I$  = intensity

$S$  = spatial Gaussian

$B$  = brightness Gaussian

$sdx$  = mask standard deviation/image  $x$  resolution

$sdy$  = mask standard deviation/image  $y$  resolution

$T$  = brightness threshold

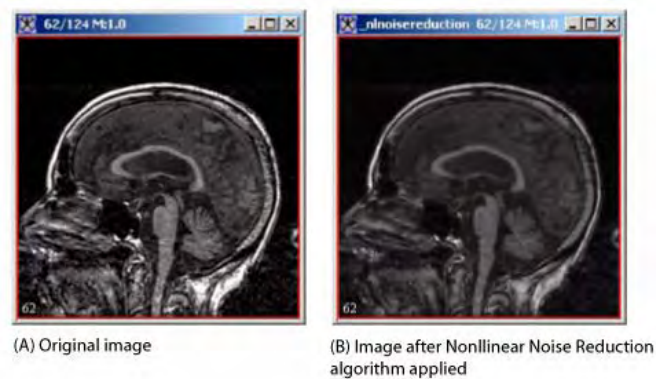
The *brightness threshold* allows discrimination between noise and the underlying image. Ideally, this value should be set greater than the noise level and less than the contrast of the underlying image. The filter blurs edges of contrast smaller than this threshold but not those of greater contrast. Reducing the brightness threshold gives less smoothing. The brightness threshold has a default value = 0.1 \* (image maximum – image minimum).

The mask standard deviation (SD) determines the spatial extent of the



smoothing. The mask is basically Gaussian with a standard deviation that you specify. However, for a small, fast, flat response with a 3 x 3 or 3 x 3 x 3 voxel mask, set the mask SD to 0. By default, the mask SD equals the  $x$  resolution.

This filter can process a 3D image as an entire indivisible unit or as separate independent slices. Refer to Figure 1 for an example of image processing using the Nonlinear Noise Reduction algorithm.



**Figure 1. Image processing using the Nonlinear Noise Reduction algorithm**

---

## IMAGE TYPES

You can apply this algorithm to black-and-white 2D, 2.5D, and 3D images.

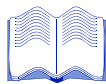
---

## SPECIAL NOTES

None.

---

## REFERENCES



Refer to the following references for more information:

For information about SUSAN, see <http://www.fmrib.ox.ac.uk/~steve>.

S. M. Smith and J. M. Brady, "SUSAN—A New Approach to Low Level Image Processing," *International Journal of Computer Vision*, Vol. 23, No. 1, May 1997, pp. 45-78.

## Applying the Nonlinear Noise Reduction algorithm

To run this algorithm, complete the following steps:

- 1 Open an image.
- 2 Select Algorithms > Filter (spatial) > Nonlinear noise reduction. The Nonlinear Noise Reduction dialog box (Figure 2) opens.
- 3 Complete the information in the dialog box.
- 4 Click OK. The algorithm begins to run.

A pop-up window appears with the status. The following message appears: “Performing the nonlinear noise reduction.” When the algorithm finishes running, the pop-up window closes.

Depending on whether you selected New image or Replace image, the results appear in a new window or replace the image to which the algorithm was applied.

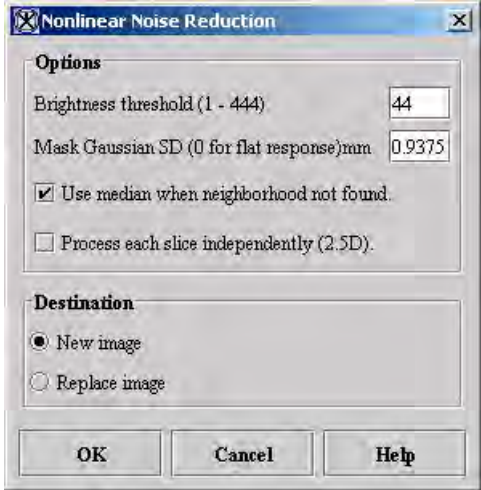
<b>Brightness threshold</b>	Blurs edges of contrast smaller than this threshold. Reducing the brightness threshold gives less smoothing. The default value is $0.1 * (\text{image maximum} - \text{image minimum})$ .	
<b>Mask Gaussian SD (0 for flat response) mm</b>	Determines the spatial extent of the smoothing (the default value is equal to the X resolution). For a small, fast, flat response with a 3 x 3 or 3 x 3 x 3 voxel mask, set the mask SD to 0.	
<b>Use median when neighborhood not found</b>	Refer to Table 1 on page 279 for an explanation. By default, this check box is selected.	
<b>Process each slice independently</b>	Smooths each slice of the dataset independently (applies only to 3D images).	
<b>New image</b>	Shows the results of the algorithm in a new image window (default choice).	
<b>Replace image</b>	Replaces the current active image with the newly calculated image.	
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.	

Figure 2. Nonlinear Noise Reduction dialog box

---

<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

---

**Figure 2. Nonlinear Noise Reduction dialog box (continued)**

## Filters (Spatial): Nonmaximum Suppression

*This algorithm defines the edges of an image by calculating the nonmaximum suppression of an image at a user-defined scale. An edge is defined as the union of points for which the gradient magnitude assumes a maximum in the gradient direction.*

For more information about the algorithm, refer to the MIPAV web site: {<http://mipav.cit.nih.gov/documentation/HTML%20Algorithms/FiltersSpatialNonmaximumSuppression.html>}.

### Background

The Nonmaximum Suppression algorithm uses a local orthonormal coordinate system  $(u, v)$  at any point  $(P_i)$ , where the  $v$  axis is parallel to the gradient direction at  $P_i$ , and the  $u$  axis is perpendicular, and where  $I$  is the intensity value of the pixel.

$$I_u = \sin \alpha I_x - \cos \alpha I_y$$

$$I_v = \cos \alpha I_x + \sin \alpha I_y$$

$$\cos \alpha = \frac{I_x}{\sqrt{I_x^2 + I_y^2}} \text{ evaluated at } P_i$$

$$\sin \alpha = \frac{I_y}{\sqrt{I_x^2 + I_y^2}} \text{ evaluated at } P_i$$

$$I_u = \frac{(I_x I_y - I_x I_y)}{\sqrt{I_x^2 + I_y^2}} = 0$$

$$I_v = \frac{(I_x^2 + I_y^2)}{\sqrt{I_x^2 + I_y^2}} = \sqrt{I_x^2 + I_y^2}$$

$$I_v^2 = I_x^2 + I_y^2$$

$$\begin{aligned}
 I_v^2 I_{vv} &= I_x^2 I_{xx} + 2I_x I_y I_{xy} + I_y^2 I_{yy} \\
 I_v^3 I_{vvv} &= I_x^3 I_{xxx} + 3I_x^2 I_y I_{xxy} + 3I_x I_y^2 I_{xyy} + I_y^3 I_{yyy} \\
 I_{vv} &= (\cos \alpha I_x + \sin \alpha I_y)(\cos \alpha I_x + \sin \alpha I_y) \\
 &= (\cos \alpha)^2 I_{xx} + 2\cos \alpha \sin \alpha I_{xy} + (\sin \alpha)^2 I_{yy} \\
 &= \frac{(I_x^2 I_{xx} + 2I_x I_y I_{xy} + I_y^2 I_{yy})}{(I_x^2 + I_y^2)}
 \end{aligned}$$

$$\begin{aligned}
 I_{vvv} &= ((\cos \alpha)^2 I_{xx} + 2\cos \alpha \sin \alpha I_{xy} + (\sin \alpha)^2 I_{yy}) \cos \alpha I_x + \sin \alpha I_y = (\cos \alpha)^3 I_{xxx} \\
 &+ 3(\cos \alpha)^2 \sin \alpha I_{xxy} + 3\cos \alpha (\sin \alpha)^2 I_{xyy} + (\sin \alpha)^3 I_{yyy} \\
 &= \frac{I_x^3 I_{xxx} + 3I_x^2 I_y I_{xxy} + 3I_x I_y^2 I_{xyy} + I_y^3 I_{yyy}}{(I_x^2 + I_y^2)^{1.5}}
 \end{aligned}$$

Assuming that the second- and third-order directional derivatives of  $I$  in the  $v$  direction are not simultaneously zero, a necessary and sufficient condition for  $P_i$  to be a gradient maximum in the gradient direction may be stated as:

$$\begin{aligned}
 I_{vv} &= 0 \\
 I_{vvv} &< 0
 \end{aligned}$$

Since only the sign information is important, this condition can be restated as:

$$I_v^2 I_{vv} = 0$$

$$I_v^3 I_{vvv} < 0$$

The nonmaximum suppression is the following:

$$I_v^2 I_{vv} = I_x^2 I_{xx} + 2I_x I_y I_{xy} + I_y^2 I_{yy}$$

Edge detection is an option for 2D images, but not for 2.5D or 3D images. If edge detection is selected, MIPAV generates an unsigned byte image in which the value of the edges is 255 and of all other points is 0. The nonmaximum suppression data must always be scaled to the same range, so the same noise threshold values always have the same meaning. For edge detection, the nonmaximum suppression data is scaled by  $20,000.0 / (\text{image maximum} - \text{image minimum})$ . We must not use a linear transformation, which shifts positive to negative or negative to positive.

If noise threshold is used, MIPAV sets scaled nonmaximum suppression values  $\geq$  low threshold and  $\leq$  high threshold equal to zero. It considers groups of four scaled, noise thresholded, nonmaximum suppression values, which intersect at one common corner. MIPAV records an edge value of 255 only if at least one of these four modified nonmaximum suppression values is negative, at least one of these four modified nonmaximum suppression values is positive, and the upper left-hand pixel of the group has a value of

$$I_v^3 I_{vvv} < 0 .$$

For 3D images a similar derivation follows:

$$I_v = \cos \alpha \sin \beta I_x + \sin \alpha \sin \beta I_y + \cos \beta I_z$$

where spherical coordinates are being used.

The nonmaximum suppression is:

$$I_v^2 I_{vv} = I_x^2 I_{xx} + 2I_x I_y I_{xy} + 2I_x I_z I_{xz} + I_y^2 I_{yy} + 2I_y I_z I_{yz} + I_z^2 I_{zz}$$

$$I_v = \frac{(I_x^2 + I_y^2 + I_z^2)}{\sqrt{I_x^2 + I_y^2 + I_z^2}} = \sqrt{I_x^2 + I_y^2 + I_z^2}$$

$$I_v^2 = I_x^2 + I_y^2 + I_z^2$$

$$\cos \alpha = \frac{I_x}{\sqrt{I_x^2 + I_y^2}} \text{ evaluated at } P_i$$

$$\sin \alpha = \frac{I_y}{\sqrt{I_x^2 + I_y^2}} \text{ evaluated at } P_i$$

$$\cos \beta = \frac{I_z}{\sqrt{I_x^2 + I_y^2 + I_z^2}} \text{ evaluated at } P_i$$

$$\sin \beta = \frac{\sqrt{I_x^2 + I_y^2}}{\sqrt{I_x^2 + I_y^2 + I_z^2}} \text{ evaluated at } P_i$$

$$I_v^3 I_{vvv} = I_x^3 I_{xxx} + 3I_x^2 I_y I_{xxy} + 3I_x^2 I_z I_{xxz} + 3I_x I_y^2 I_{xyy} + 6I_x I_y I_z I_{xyz} + 3I_x I_z^2 I_{xzz} + I_y^3 I_{yyy} + 3I_y^2 I_z I_{yyz} + 3I_y I_z^2 I_{yzz} + I_z^3 I_{zzz}$$

If you select VOI regions in the image, then the nonmaximum suppression image always contains the original source image values in regions outside the VOIs.

If Use image resolutions to normalized z scale is selected, then the z scale is multiplied by the image x resolution/image z resolution, so that the z scale is normalized to be the same as the x scale.



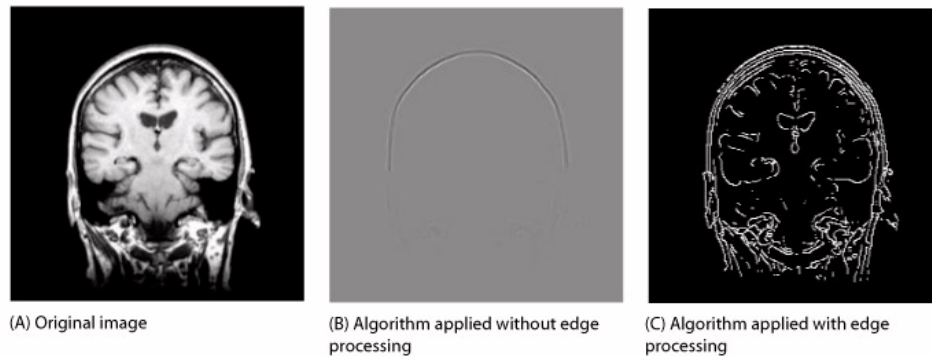


Figure 1. Examples of Nonmaximum Suppression processing

---

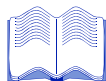
## IMAGE TYPES

The nonmaximum suppression image can be generated from 2D, 2.5D, and 3D black-and-white images. However, the option to output an edge image is only available for black-and-white 2D images.

---

## SPECIAL NOTES

None.



---

## REFERENCES

Refer to the following reference for more information about this algorithm:

Lindeberg, Tony, and Bart M. ter Haar Romeny, Chapter 2, “Linear Scale-Space II: Early Visual Operations,” in *Geometry-Driven Diffusion in Computer Vision*, ed. Bart M. ter Haar Romeny (Dordrecht, The Netherlands: Kluwer Academic Publishers, 1994) p. 45.

---

## Applying the Nonmaximum Suppression algorithm

To run this algorithm, complete the following steps:

- 1** Open an image.
- 2** Perform, as an option, any other image processing, such as improving the contrast, on the image.
- 3** Select Algorithms > Filter (spatial) > Nonmaximum suppression. The Nonmaximum suppression dialog box (Figure 2) opens.
- 4** Complete the information in the dialog box.
- 5** Click OK. The algorithm begins to run.

A pop-up window appears with the status. The following message appears: “Calculating the Nonmaximum suppression.”

When the algorithm finishes running, the pop-up window closes. Depending on whether you selected New image or Replace image, the results appear in a new window or replace the image to which the algorithm was applied.



---

**Note:** For 2D images, if you selected Output edge image, a new window with an edge image appears.

---

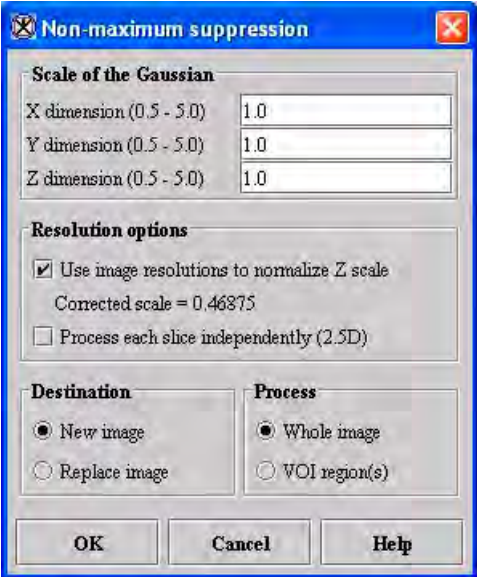
<b>X dimension</b>	Indicates the scale of the Gaussian in the <i>X</i> direction (the default value is 1.0).	
<b>Y dimension</b>	Indicates the scale of the Gaussian in the <i>Y</i> direction (the default value is 1.0).	
<b>Z dimension</b>	Indicates the scale of the Gaussian in the <i>Z</i> direction (for 3D images only). The default value is 1.0.	
<b>Use image resolutions to normalize Z scale</b>	<p>Normalizes the <i>Z</i> scale to compensate for the difference if the voxel resolution in distance per pixel is greater between slices than the voxel resolution in-plane (for 3D images only, the default value is enabled).</p> <p>If enabled, then <math>Z = Z \left( \frac{XR_s}{ZR_s} \right)</math> where <math>Z</math> = scale <i>Z</i>, <math>XR_s</math> = image <i>X</i> resolution, and <math>ZR_s</math> = image <i>Z</i> resolution).</p>	
<b>Process each slice independently</b>	Calculates nonmaximum suppression for each slice of the dataset independently (for 3D images only the default value is enabled).	
<b>New image</b>	Shows the results of the algorithm in a new image window (default choice). If selected, an output edge image appears in a second new window.	
<b>Replace image</b>	Replaces the current active image with the results of the algorithm.	
<b>Whole image</b>	Applies the algorithm to the whole image (default choice).	
<b>VOI region(s)</b>	Applies the algorithm inside VOIs. Outside VOIs, the pixel values are unchanged.	
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.	
<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes this dialog box.	
<b>Help</b>	Displays online help for this dialog box.	

Figure 2. Nonmaximum Suppression dialog box

---

## Filters (Spatial): Regularized Isotropic (Nonlinear) Diffusion

*Regularized isotropic nonlinear diffusion is a specific technique within the general classification of diffusion filtering. Diffusion filtering, which models the diffusion process, is an iterative approach of spatial filtering in which image intensities in a neighborhood are utilized to compute new intensity values.*

Two major advantages of diffusion filtering over many other spatial domain filtering algorithms are:

- *A priori* image information can be incorporated into the filtering process
- The iterative nature of diffusion filtering allows for fine-grain control over the amount of filtering performed.

There is not a consistent naming convention in the literature to identify different types of diffusion filters. This documentation follows the approach used by Weickert. Specifically, since the diffusion process relates a concentration gradient with a flux, *isotropic diffusion* means that these quantities are parallel. *Regularized* means that the image is filtered prior to computing the derivatives required during the diffusion process. In linear diffusion the filter coefficients remain constant throughout the image, while *nonlinear* diffusion means the filter coefficients change in response to differential structures within the image.

---

For more information about the algorithm, refer to the MIPAV web site: {<http://mipav.cit.nih.gov/documentation/HTML%20Algorithms/FiltersSpatialRegularizedIsotropicDiffusion.html>}.  

---

## Background

All diffusion filters attempt to determine the image  $I(x, y)$  that solves the diffusion equation, which is a second-order partial differential defined as

$$\partial_t I = \text{div}(D\nabla I)$$

where

$\partial_t$  =The time derivative of the image

$\text{div}$ =The divergence operator

$D$ =The diffusion tensor (which may be a scalar value)

$\nabla$  =The gradient of the image

The quantity that distinguishes different diffusion filters is primarily the diffusion tensor, which is also called the *diffusivity*.

In *homogeneous linear diffusion* filtering, the diffusivity,  $D$ , is set to 1, and the diffusion equation becomes:

$$\partial_t I = \partial_{xx} I + \partial_{yy} I$$

In *isotropic nonlinear diffusion* filtering, the diffusivity term is a monotonically decreasing scalar function of the gradient magnitude squared, which encapsulates edge contrast information. In this case, the diffusion equation becomes:

$$\partial_t I = \text{div}(D(|\nabla I|^2)\nabla I)$$

One useful diffusivity function is:

$$D(|\nabla I|^2) = \frac{1}{1 + (|\nabla I|^2/\lambda^2)}$$

In this equation  $\lambda$  is called a diffusion constant, which defines the point of inflection of the diffusivity function, which is shown for  $\lambda = 0.15$  in Figure 1.

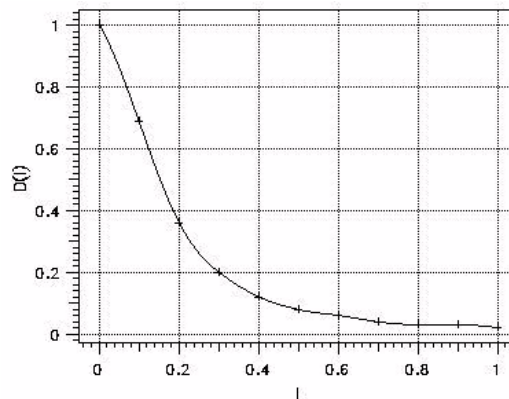


Figure 1. Diffusion constant for  $\lambda = 0.15$

Figure 1 shows that, for a given value of the diffusion constant  $\lambda$ , the larger the values of the image gradient, the smaller the value of the diffusivity function. Therefore, large values of the gradient, indicating the presence of an edge, the smaller the amount of diffusion. As the image gradient varies over the image, so does the amount of diffusion, which is why this is called an *isotropic nonlinear diffusion filter*.

It is well known that derivative operations performed on a discrete grid are an ill-posed problem, meaning derivatives are overly sensitive to noise. To convert derivative operations into a well-posed problem, the image is low-pass filtered or smoothed prior to computing the derivative. *Regularized isotropic (nonlinear) diffusion* filtering is formulated the same as the *isotropic nonlinear diffusion* detailed above; however, the image is smoothed prior to computing the gradient. The diffusion equation is modified slightly to indicate regularization by including a standard deviation term in the image gradient as shown in the following equation:

$$\partial_t I = \text{div} \left( D \left( |\nabla I_\sigma|^2 \right) \nabla I_\sigma \right)$$

The smoothing to regularize the image is implemented as a convolution over the image and therefore this filter is a linear operation. Since differentiation is also a linear operation, the order of smoothing and differentiation can be switched, which means the derivative of the convolution kernel is computed and convolved with the image, resulting in a well-posed measure of the image derivative.

The images in Figure 2 show the results of applying the *regularized isotropic (nonlinear) diffusion* filter to an example MR brain image.

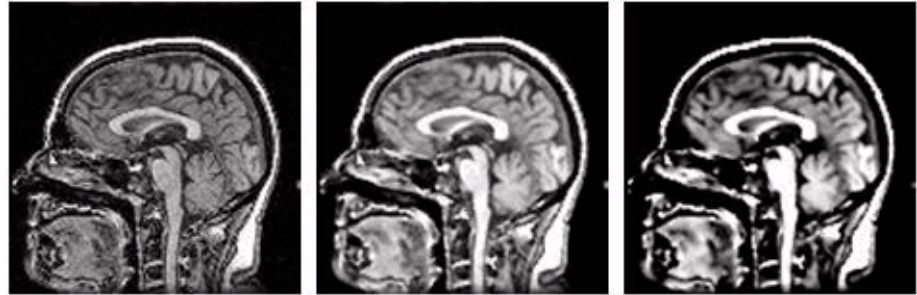


Figure 2. Examples of regularized isotropic (nonlinear) diffusion

---

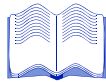
## IMAGE TYPES

You can apply this algorithm to all data types except complex and to 2D, 2.5D, and 3D images.

---

## SPECIAL NOTES

The resulting image is, by default, a float image.



---

## REFERENCES

Refer to the following references for more information about diffusion filtering in general and this algorithm in particular.

Weickert, Joachim. "Nonlinear Diffusion Filtering," in *Handbook of Computer Vision and Applications*, Volume 2, eds. Bernd Jahne, Horst Haussecker, and Peter Geissler. (Academic Press, April 1999), 423–450.

Weickert, Joachim. *Anisotropic Diffusion in Image Processing* (Stuttgart, Germany: Teubner, 1998).

## Applying the Regularized Isotropic (Nonlinear) Diffusion algorithm

To run this algorithm, complete the following steps:

- 1** Select Algorithms > Filter > Regularized Isotropic Diffusion. The Regularized Isotropic Diffusion dialog box opens (Figure 3).
- 2** Complete the fields in the dialog box.
- 3** When complete, click OK.

The algorithm begins to run, and a status window appears. When the algorithm finishes, the resulting image appears in a new image window.

<b>Number of iterations</b>	Specifies the number of iterations, or number of times, to apply the algorithm to the image.
<b>Gaussian standard deviation</b>	Specifies the standard deviation of the Gaussian filter used to regularize the image.
<b>Diffusion contrast parameter</b>	Specifies the inflection point in the diffusivity function, which dictates the shape of the function.
<b>Process each slice separately</b>	Applies the algorithm to each slice individually. By default, this option is selected.
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

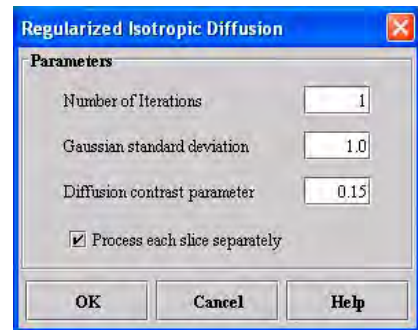


Figure 3. Regularized Isotropic Diffusion dialog box



## Filters (Spatial): Slice Averaging

*This algorithm provides a way of reducing image noise by summing together a set of noisy images and dividing the sum by the number of images. The algorithm assumes that the noise in the images is uncorrelated and has zero average value.*

---

For more information about the algorithm, refer to the MIPAV web site: {<http://mipav.cit.nih.gov/documentation/HTML%20Algorithms/FiltersSpatialSliceAveraging.html>}.

---

### Background

This algorithm averages together the slices of a 3D image. The average used here is:

Mean = sum of values / total number of values

If you select All as the number of slices averaged:

$$A(x, y) = \frac{\sum_{s=1}^{LSN} value(x, y, s)}{LSN}$$

If you select 3, 5, or 7 as the number of slices averaged:

$$A(x, y, z) = \frac{\sum_{s=lowest}^{highest} value(x, y, s)}{(highest - lowest + 1)}$$

where

*A* = slice average

*LSN* = last slice number

*lowest* = maximum (1, *z* - offset)

*highest* = minimum (last slice number,  $z + \text{offset}$ )

*offset* = (number of slices averaged - 1)/2

The preceding equations prevent the inclusion of slice numbers less than the first slice number or greater than the last slice number in the averaging equation.



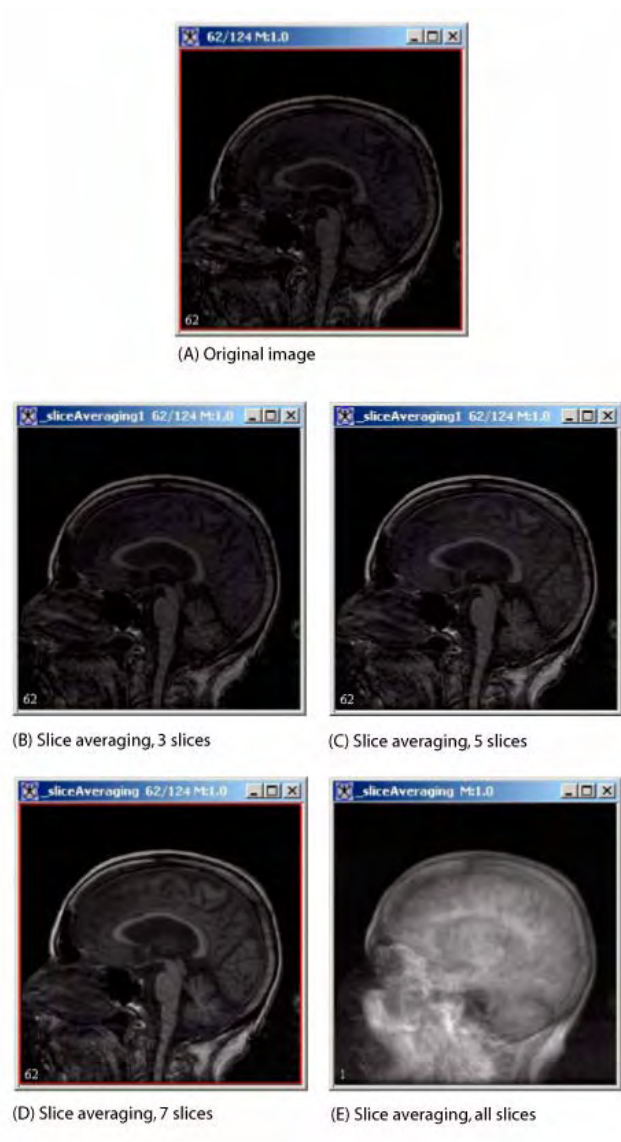
---

**Note:** When averaging on the first slice with an averaging Number = 7, only include the first slice and the three slices above it in the averaging. When averaging on the last slice with averaging Number = 5, only include the last slice and the two slices below it in the averaging.

---

Under **Slice Number**, the default choice is **All**. If **All** is chosen, MIPAV creates a 2D image. If 3, 5, or 7 are chosen, then a 3D image is created.

Because a 2D image cannot meaningfully replace a 3D image, if you select **All**, you can only choose **New image**. However, if you select 3, 5, or 7 as the number of slices, **New image** is the default choice, although you can select either **New image** or **Replace image**.



**Figure 1. Slice Averaging algorithm processing**

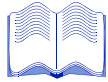
---

## IMAGE TYPES

You can apply this algorithm to all image data types and to 2D, 2.5D, and 3D images.

## SPECIAL NOTES

None.



## REFERENCES

Refer to the following for more information about this algorithm:

Raphael C. Gonzalez and Richard E. Woods. *Digital Image Processing*, Second Edition (Upper Saddle River, New Jersey: Prentice-Hall, Inc., 2002), pp. 119–123.

## Applying the Slice Averaging algorithm

To run this algorithm, complete the following steps:

- 1 Open an image.
- 2 Select Algorithms > Filter (spatial) > Slice Averaging. The Slice Averaging dialog box (Figure 2) opens.

3	Averages 3 slices in the dataset and creates a 3D image.
5	Averages 5 slices in the dataset and creates a 3D image.
7	Averages 7 slices in the dataset and creates a 3D image.
All	Averages all of the slices in the dataset and creates a 2D image (default choice).
New image	Shows the results of the algorithm in a new image window (default choice).
Replace image	Replaces the current active image with the newly calculated image.
OK	Applies the algorithm according to the specifications in this dialog box.
Cancel	Disregards any changes that you made in this dialog box and closes the dialog box.



Figure 2. Slice Averaging dialog box

---

<b>Help</b>	Displays online help for this dialog box.
-------------	---

---

**Figure 2. Slice Averaging dialog box**

**3** Complete the information in the dialog box.

**4** Click OK.

The algorithm begins to run, and a pop-up window appears with the status. The following messages appear: “Averaging data” and “Importing average data.”

When the algorithm finishes running, the pop-up window closes, and the results appear either in a new window or replace the image in which the algorithm was applied.

## Filters (Spatial): Unsharp Mask

*When applied to an image, this algorithm produces a sharpened version of the image or the volume defined by a VOI of the image. A high-pass filtered (unsharp masked) image can be calculated by taking the difference of the original image and the low-pass filtered (blurred) version of that image. The effect of a high-pass filter, like the unsharp mask technique, is to enhance sharp intensity transitions (i.e., edges) but at the cost of enhancing noise.*

For more information about the algorithm, refer to the MIPAV web site: {<http://mipav.cit.nih.gov/documentation/HTML%20Algorithms/FilterSpatialUnsharpMask.html>} .

## Background

The process of unsharp masking is defined by:

$$\textit{High pass} = \textit{original} - \textit{low pass}$$

To control the amount of filtering, a weighting factor ( $k$ ) is added to the function:

$$\textit{High pass} = \textit{original} - (k * \textit{blurred image}) \textit{ where } 1 > k > 0$$

Choosing a small value for  $k$  (i.e., 0.2) produces an image where a small amount of filtering is applied to the original image. Choosing a large value for  $k$  produces a highly filtered image. Recall that convolving a Gaussian function with an image produces a low-pass filtered image. The selection of the standard deviation of the Gaussian affects the results of the unsharp masked image. For example, a low-pass filtered image produced using a small standard deviation (i.e., a little blurring) produces an unsharp masked image that is filtered only a small amount (Figure 1).

## SPECIAL NOTES

The following notes apply:

- If you choose to show the results of the algorithm in a new window, the resultant image is, by default, a float type image. If you choose the replace the current image, the image type remains the same.
- To achieve 2.5D processing (each slice of the volume is filtered independently) of a 3D dataset, select Process each slice independently (2.5D) in the Unsharp Mask dialog box (Figure 2).



**Figure 1. Unsharp Mask processing**

(A) is the original image of retina of the eye, and (B) is the unsharp mask where  $k = 0.75$  and the Gaussian SD = 1.0.

## REFERENCES

See the following references for more information about this algorithm:

Tony Lindeberg and Bart Ter Har Romeny, "Linear Scale-Space I: Basic Theory," *Geometry-Driven Diffusion in Computer Vision*, ed. Bart Ter Har Romeny ((Dordrecht, The Netherlands: Kluwer Academic Publishers, 1994), pp. 1–38.

Raphael C. Gonzalez and Richard E. Woods, *Digital Image Processing* (Boston: Addison Wesley, 1992), pp. 196–197.



---

## IMAGE TYPES

You can apply this algorithm to all image data types, except complex and RGB images, and to 2D, 2.5D, 3D, and 4D images.

## Applying the Unsharp Mask algorithm

To run this algorithm, complete the following steps:

- 1** Select Algorithms > Filter (spatial) > Unsharp mask. The Unsharp Mask dialog box (Figure 2) opens.
- 2** Complete the information in the dialog box.
- 3** Click OK.

The algorithm begins to run, and a pop-up window appears with the status. The following message appears: “Unsharp Masking Image.”

When the algorithm finishes running, the pop-up window closes, and the results either appear in a new window or replace the image to which the algorithm was applied.

<b>X Dimension</b>	Specifies the standard deviation (SD) of Gaussian in the X direction.
<b>Y Dimension</b>	Specifies the SD of Gaussian in the Y direction.
<b>Z Dimension</b>	Specifies the SD of Gaussian in the Z direction.
<b>Use image resolutions to normalize Z scale</b>	Normalizes the Gaussian to compensate for the difference if the voxel resolution is less between slides than the voxel resolution in-plane. This option is selected by default.
<b>Process each slice independently (2.5D)</b>	Filters each slice of the dataset independently of adjacent slices.
<b>Image - (k &lt; 1) * blurred image</b>	Determines the amount of filtering, or sharpening, applied to the edges of the image or a VOI of the image. Low values sharpen image edges but increase noise. The default value is 0.75.
<b>New image</b>	Shows the results of the algorithm in a new image window.
<b>Replace image</b>	Replaces the current active image with the results of the algorithm.
<b>Whole image</b>	Applies the algorithm to the whole image.
<b>VOI region(s)</b>	Applies the algorithm to the volumes (regions) delineated by the VOIs.
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

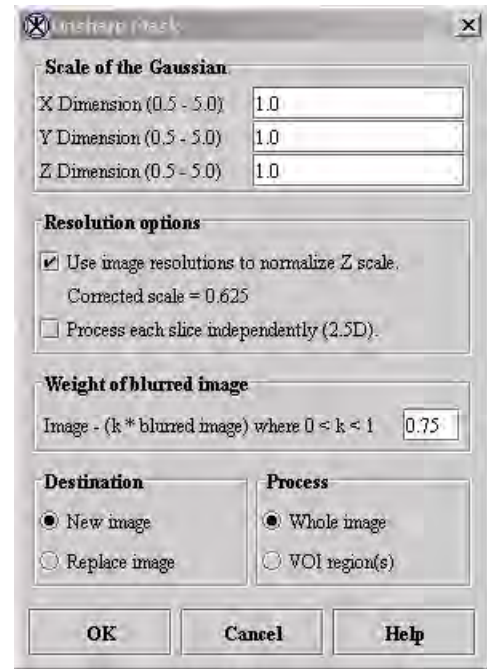


Figure 2. Unsharp Mask dialog box

---

## Filters (Wavelet): De-noising BLS GSM

*The wavelet transform or wavelet analysis is a solution to overcome the shortcomings of the Fourier transform. In wavelet analysis, the modulated window is shifted along the signal, and for every position, the spectrum is calculated. Then, this process is repeated many times with a slightly shorter (or longer) window for every new cycle. In the end, the result appears as a collection of time-frequency representations of the signal, all with different resolutions. Since the modulated window is fully scalable, this solves the signal-cutting problem which arises in the Fourier transform.*

Because the result appears as a collection of time-frequency representations of the signal, we can speak of a multiresolution analysis. However, in the case of wavelets, we normally do not speak about time-frequency representations, but about time-scale representations, scale being in a way the opposite of frequency, because the term frequency is reserved for the Fourier transform.

---

For more information about the algorithm, refer to MIPAV Volume 2 User Guide on the MIPAV web site: {<http://mipav.cit.nih.gov/documentation/HTML%20Algorithms/FiltersWaveletDenoising.html>}

---

## Background

This method is part of MATLAB BLS-GSM Image Denoising software written by Javier Portilla of Universidad de Granada, Spain. For more information, refer to <<http://decsai.ugr.es/~javier/denoise/index.htm>>.



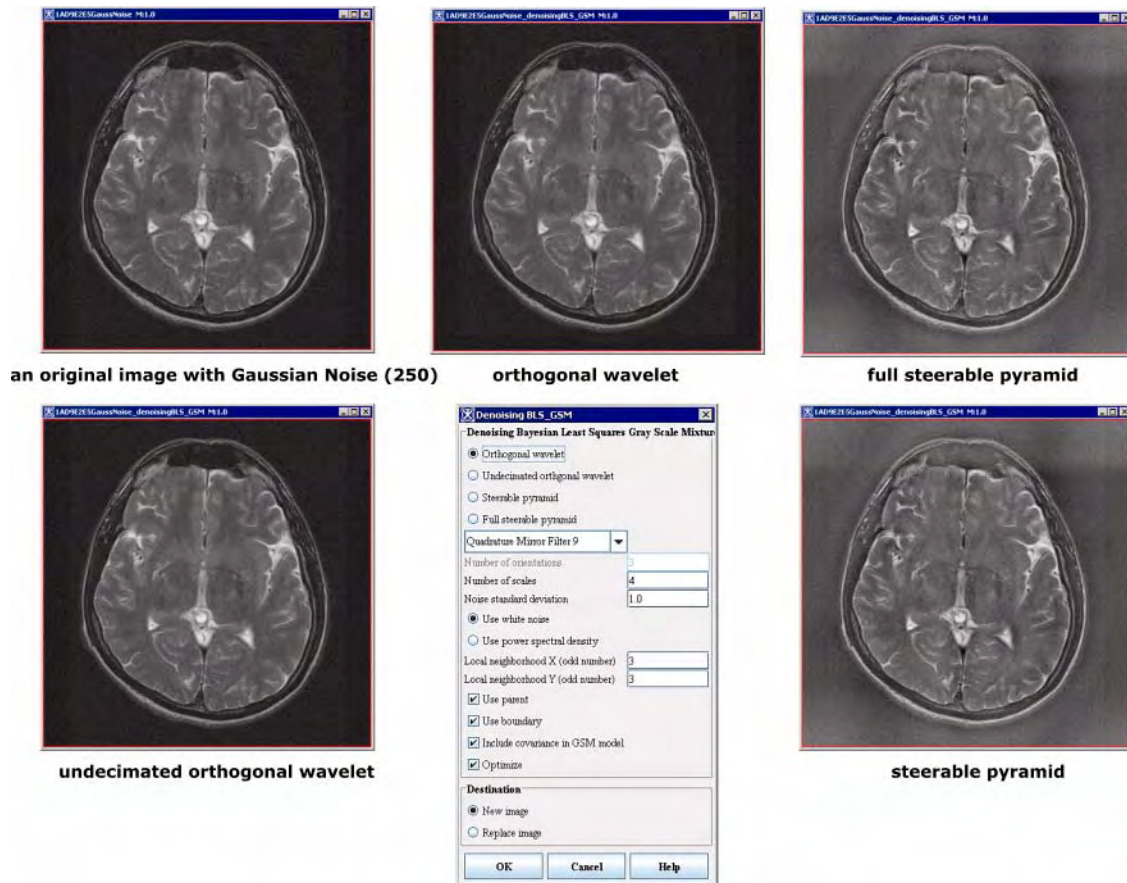


Figure 1. Application of the Filters wavelet BLS GSM algorithm to the noisy image

## STEERABLE PYRAMID

The Steerable Pyramid is a linear multi-scale, multi-orientation image decomposition that provides a useful front-end for image-processing and computer vision applications. The basis functions of the steerable pyramid are directional derivative operators that come in different sizes and orientations. An example decomposition of an image of a white disk on a black background is shown in Figure 2 (Left). This particular steerable pyramid contains 4 orientation subbands, at 2 scales. The number of orientations may be adjusted by changing the derivative order (for example,

first derivatives yield two orientations). The smallest subband is the residual lowpass information. The residual highpass subband is not shown.

The block diagram for the decomposition (both analysis and synthesis) is shown in Figure 2 (Right). Initially, the image is separated into low and highpass subbands, using filters  $L_0$  and  $H_0$ . The lowpass subband is, then divided into a set of oriented bandpass subbands and a low(er)-pass subband. This low(er)-pass subband is subsampled by a factor of 2 in the X and Y directions. The recursive construction of a pyramid is achieved by inserting a copy of the shaded portion of the diagram at the location of the solid circle (i.e., the lowpass branch). For more detailed descriptions see "References" on page 310.

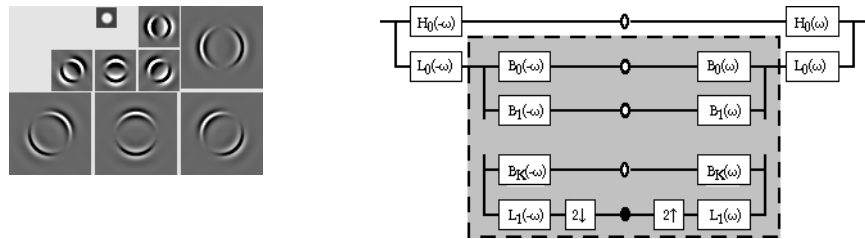


Figure 2. Left: an example of decomposition of an image of a white disk on a black background; Right: the block diagram for that decomposition

## REFERENCES

Refer to the following references for more information about the Thresholding algorithm.

"Image Denoising Using Scale Mixtures of Gaussians in the Wavelet Domain", Javier Portilla, Vasily Strela, Martin J. Wainwright, and Eero P. Simoncelli, IEEE Transactions on Image Processing, Vol. 12, No. 11, November, 2003, pp. 1338-1351.

"The Steerable Pyramid: A Flexible Architecture for Multi-Scale Derivative Computation", Eero P. Simoncelli and William T. Freeman, 2nd IEEE International Conference on Image Processing, Washington, D.C., vol. III, pp. 444-447, October, 1995.

## IMAGE TYPES

This algorithm only applies to 2D black and white images.

## Applying the Wavelet BLS GSM algorithm

To use this algorithm, do the following:

- 7** Open an image of interest.
- 8** Navigate to Algorithms > Filters (wavelet) > Denoising BLS GSM.
- 9** The Denoising Bayesian Least Squares Gray Scale Mixture dialog box appears.
- 10** Complete the dialog box. For more information about the dialog box options, refer to Denoising Bayesian Least Squares Gray Scale Mixture dialog box.
- 11** Specify where you wish the denoised image to appear - in a new image frame or in the same frame replacing the existing image.
- 12** Press OK.
- 13** The algorithm begins to run and the progress window appears with the status. When the algorithm finishes running, the denoised image appears in the designated image frame.

Denoising Bayesian Least Squares Gray Scale Mixture	
<b>Orthogonal wavelet</b>	<p>This is very fast technique, but of relatively poor denoising performance, because it is not translation invariant. For the Orthogonal wavelet technique the following filter options available:</p> <ul style="list-style-type: none"> <li>• Daubechies 2,3,4</li> <li>• Haar</li> <li>• Quadrative Mirror Filter 5, 8, 9, 12, 13, 16</li> </ul> <p>Numbers after the filter name refer to a highest number A of vanishing moments for given support width <math>N=2A</math> used in the filter.</p>

Figure 3. Wavelet Thresholding dialog box

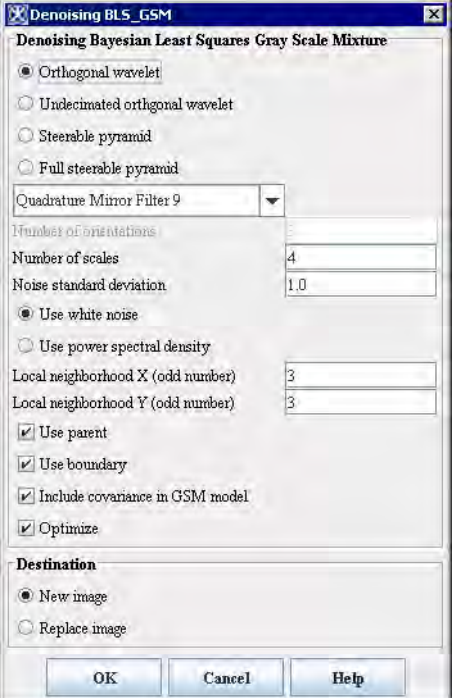
<p><b>Undecimated version of orthogonal wavelet</b></p>	<p>This is a highly redundant version of the orthogonal wavelet that avoids intra-subband aliasing, but still keeps a multiscale structure. The lowest level of the pyramid is extended 2 times in each dimension, whereas the rest of the scales are extended 4 times in each dimension. It provides a good trade-off between computational cost and denoising performance (which is very high).</p> <p>The following filter options available: Daubechies 2,3,4.</p>	
<p><b>Steerable Pyramid</b></p>	<p>This option allows the user to specify an arbitrary number of orientations. The splitting in oriented subbands is not applied to the very high frequencies, which are represented in a single subband (high-pass residual).</p>	
	<p>With a moderate computational cost, for a modest number of orientations, e.g. 4 or less (the default value is 8), its results depend on the type of image, and is comparable (or slightly worse) on average than those obtained with the Undecimated version of orthogonal wavelet option.</p>	
<p><b>Full steerable pyramid</b></p>	<p>Same as steerable pyramid, but now also the very high frequencies are spitted into orientations. It provides very high denoising performance (for some images slightly better than undecimated orthogonal wavelet), especially with a high number of orientations (8, for example), but it is very demanding computationally.</p>	
<p><b>Number of orientations</b></p>	<p>– enter a number of orientations that will be used in calculating steerable pyramid, the default value is 8.</p>	
<p><b>Number of scales</b></p>	<p>– enter a number of scales that will be used in calculating Orthogonal wavelet, Undecimated Orthogonal wavelet and Steerable Pyramid. The default value is 4.</p>	
<p><b>Noise standard deviation</b></p>	<p>enter the noise standard deviation.</p>	
<p><b>Use white noise</b></p>	<p>– if selected, the algorithm uses white noise.</p>	
<p><b>Use power spectral density</b></p>	<p>– if selected, the algorithm uses power spectral density that equals FFT (autocorrelation). Otherwise, it uses white noise. The default setting is to use white noise.</p>	

Figure 3. Wavelet Thresholding dialog box (continued)



<b>Local neighborhood X (odd number) Local neighborhood Y (odd number)</b>	– in these boxes, specify the number of pixels in neighborhood clusters which will be used for processing subbands.
<b>Use parent</b>	–if selected, this option process subbands in couples of the same orientation adjacent in scale. That is, include the coarse scale parent in the neighborhood. Otherwise, it process subbands individually. At high noise levels eliminating the coarse-scale parent from the neighborhood significantly decreases performance.
<b>Use boundary</b>	–if selected, the algorithm uses the Mirror reflected boundary padding.
<b>Include covariance in GSM model</b>	– if selected, the algorithm includes covariance in calculating the GSM model.
<b>Optimize</b>	–if selected, the algorithm uses the full Bayesian Least Squares estimator. Otherwise, it uses a two-step estimator (the MAP estimator of the local multiplier, followed by linear estimation of the coefficient). Using the two-step estimator results in a substantial reduction in performance.
<b>New image</b>	Shows the results of the algorithm in a new image window (default choice).
<b>Replace image</b>	Replaces the current active image with the newly calculated image.
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes this dialog box.
<b>Help</b>	Displays online help for this dialog box.

**Figure 3. Wavelet Thresholding dialog box (continued)**

---

## Filters (Wavelet): Thresholding

*The wavelet transform or wavelet analysis is a solution to overcome the shortcomings of the Fourier transform. In wavelet analysis, the modulated window is shifted along the signal, and for every position, the spectrum is calculated. Then, this process is repeated many times with a slightly shorter (or longer) window for every new cycle. In the end, the result appears as a collection of time-frequency representations of the signal, all with different resolutions. Since the modulated window is fully scalable, this solves the signal-cutting problem which arises in the Fourier transform.*

Because the result appears as a collection of time-frequency representations of the signal, we can speak of a multi resolution analysis. However, in the case of wavelets, we normally do not speak about time-frequency representations, but about time-scale representations, scale being in a way the opposite of frequency, because the term frequency is reserved for the Fourier transform.

---

For more information about the algorithm, refer to MIPAV Volume 2 User Guide on the MIPAV web site: {<http://mipav.cit.nih.gov/documentation/HTML%20Algorithms/FiltersWaveletThresholding.html>}

---

## Background

The most basic definition of a wavelet is simply a function with a well defined temporal support that twist about the X axis (the wavelet should have exactly the same area above and below the axis).

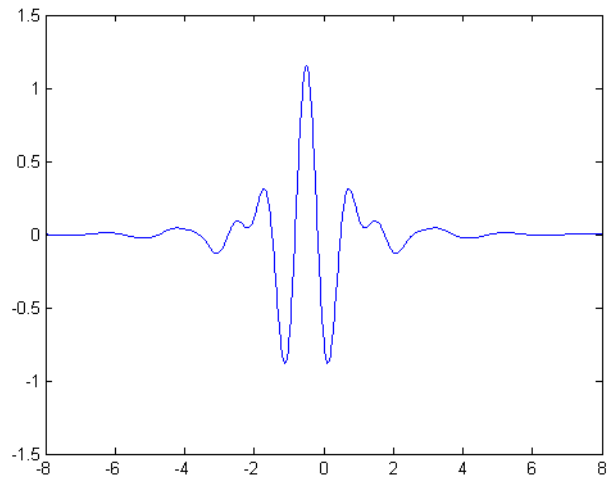


Figure 1. Meyer wavelet

For more information about wavelets, refer to EE482C: Advanced Computer Organization Stream Processor Architecture, Stanford University, Homework #1, April, 18, 2002, 1.1 What are Wavelets.

## USING WAVELETS TO REDUCE THE NOISE IN IMAGES

Consider that the image intensities for a given image are specified in the standard univariate non parametric regression setting

$$Y_i = g(t_i) + s * e_i, \quad i = 1, \dots, n, \quad (\text{EQ 1})$$

where  $e_i$  are independent  $N(0, 1)$  random variables and the noise level *sigma* may be known or unknown. The goal is to recover the underlying function  $\mathbf{g}$  from the noisy data,  $Y = (y_1, \dots, y_n)$ , without assuming any particular parametric structure for  $\mathbf{g}$ , e.g. using the non parametric regression.

Assume that

- $t_i$  are within the unit interval  $[0, 1]$ ,
- the sample points are equally spaced, i.e.  $t_i = i/n$ ,
- and the sample size  $n$  is a power of two:  $n = 2^j$  for some positive integer  $J$ .

All these assumptions can be done without loss of generality and allow to perform both the discrete wavelet transform (DWT) and the inverse discrete wavelet transform (iDWT) using the Mallat's fast algorithm.

One of the basic approaches to non parametric regression is to consider the unknown function  $g$  expanded as a generalized Fourier series, and then to estimate the generalized Fourier coefficients from the data. This method applies the same generalized Fourier series approach using a wavelet series.

Since the DWT of white noise is also an array of independent  $N(0, 1)$  random variables, from Equation 1 follows that

$$\begin{aligned} \hat{c}_{j_0k} &= c_{j_0k} + \sigma \epsilon_{jk}, \quad k = 0, 1, \dots, 2^{j_0} - 1, \\ \hat{d}_{jk} &= d_{jk} + \sigma \epsilon_{jk}, \quad j = j_0, \dots, J - 1, \quad k = 0, \dots, 2^j - 1, \end{aligned} \tag{EQ 2}$$

where

$$\hat{c}_{j_0k} \quad \text{and} \quad \hat{d}_{jk}$$

are respectively the empirical scaling and the empirical wavelet coefficients of the noisy data  $Y$ , and  $\epsilon_{jk}$  are independent  $N(0, 1)$  random variables. The scaling coefficients

$$\hat{c}_{j_0k}$$

represent *low-frequency* terms that usually contain important components about the underlying function  $g$ .

The method assumes that essentially only a few *large*  $d_{jk}$  or *vanishing moments* contain information about the underlying function  $g$ , while *small*  $d_{jk}$  can be attributed to the noise which uniformly contaminates all wavelet coefficients. This allows to keep only large coefficients and set all others to zero, which gives us approximate wavelet representation of the underlying function  $g$ .

To decide which the significant large wavelet coefficients are, the method uses the *hard* and *soft threshold* algorithms.

---

## WAVELET THRESHOLDING

First, we assume that the smoothness index *sigma* of the function **g** (which is the function to be recovered) is known, so the resulting estimator can be obtained by estimating the scaling coefficients  $c_{j0k}$  by their empirical counterparts

and by estimating the wavelet coefficients  $d_{jk}$  via a linear shrinkage

$$\tilde{d}_{jk} = \frac{\hat{d}_{jk}}{1 + \lambda 2^{2js}} \quad (\text{EQ 3})$$

Where  $\lambda > 0$  is a smoothing parameter.

In some thresholding methods the smoothing parameter  $\lambda$  is used to determine those wavelet coefficients which contribute the most to the function **g** by thresholding them in a way where wavelet coefficients are set to zero if their absolute value is below a certain threshold level,  $\lambda$  more or equal to 0.

The following thresholding techniques are implemented in the method:

- Hard thresholding
- Soft thresholding
- Nonnegative garrote thresholding
- SCAD thresholding

**In hard thresholding**, a wavelet coefficient whose magnitude is below the product of threshold and the maximum wavelet magnitude is zeroed, and a wavelet coefficient whose magnitude is greater than or equal to the product is left unchanged. See Equation 4.

(EQ 4)

$$\delta_{\lambda}^H(\hat{d}_{jk}) = \begin{cases} 0 & \text{if } |\hat{d}_{jk}| \leq \lambda \\ \hat{d}_{jk} & \text{if } |\hat{d}_{jk}| > \lambda \end{cases}$$

**In soft thresholding**, a wavelet coefficient whose magnitude is below the product of threshold and the maximum wavelet magnitude is zeroed, and a wavelet coefficient whose magnitude is greater than or equal to the product has its magnitude decreased by the product. See Equation 15.

(EQ 5)

$$\delta_{\lambda}^S(\hat{d}_{jk}) = \begin{cases} 0 & \text{if } |\hat{d}_{jk}| \leq \lambda \\ \hat{d}_{jk} - \lambda & \text{if } \hat{d}_{jk} > \lambda \\ \hat{d}_{jk} + \lambda & \text{if } \hat{d}_{jk} < -\lambda \end{cases}$$

Thresholding allows the data itself to decide which wavelet coefficients are significant; hard thresholding is a discontinuous function, which employs a *keep or kill* rule, while soft thresholding is a continuous function that uses a *shrink or kill* rule. However, the signal reconstructed with the coefficients left after hard thresholding may have undesirable artifacts including side lobes. Therefore, two more thresholding techniques are implemented:

The **nonnegative garrote thresholding** is as follows

(EQ 6)

$$\delta_{\lambda}^G(\hat{d}_{jk}) = \begin{cases} 0 & \text{if } |\hat{d}_{jk}| \leq \lambda \\ \hat{d}_{jk} - \frac{\lambda^2}{\hat{d}_{jk}} & \text{if } |\hat{d}_{jk}| > \lambda \end{cases}$$

And the **SCAD thresholding** is as follows

(EQ 7)

$$\delta_{\lambda}^{\text{SCAD}}(\hat{d}_{jk}) = \begin{cases} \text{sign}(\hat{d}_{jk}) \max(0, |\hat{d}_{jk}| - \lambda) & \text{if } |\hat{d}_{jk}| \leq 2\lambda \\ \frac{(\alpha-1)\hat{d}_{jk} - \alpha\lambda \text{sign}(\hat{d}_{jk})}{\alpha-2} & \text{if } 2\lambda < |\hat{d}_{jk}| \leq \alpha\lambda \\ \hat{d}_{jk} & \text{if } |\hat{d}_{jk}| > \alpha\lambda \end{cases}$$

---

## ZERO PADDING

Because the wavelet transform is mathematically defined only within a signal, the boundary problem arises. To solve that problem the method uses the zero image padding technique. The image is expanded so that all dimensions are powers of 2 and there is zero padding going past each original boundary by the coefficient number - 2.

---

## OUTLINE OF THE ALGORITHM

The image is expanded so that all dimensions are powers of 2 and there is zero padding going past each original boundary by the coefficient number equals 2.

- 1** The image is transformed into wavelet coefficients using the threshold type selected by the user.
- 2** The signal reconstructed from wavelet coefficients.
- 3** Then, an inverse wavelet transform is performed.
- 4** The image is stripped down to its original size. The data is clamped so as not to exceed the bounds of the data type of the image into which it is imported.

---

Note: the forward and inverse Daubechies wavelet transform routines are taken from Numerical Recipes in C, The Art of Scientific Computing 2nd edition by William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, Cambridge University Press, 1997, Chapter 13.10, pp. 591 - 606.

---

---

## REFERENCES

Refer to the following references for more information about the Thresholding algorithm.

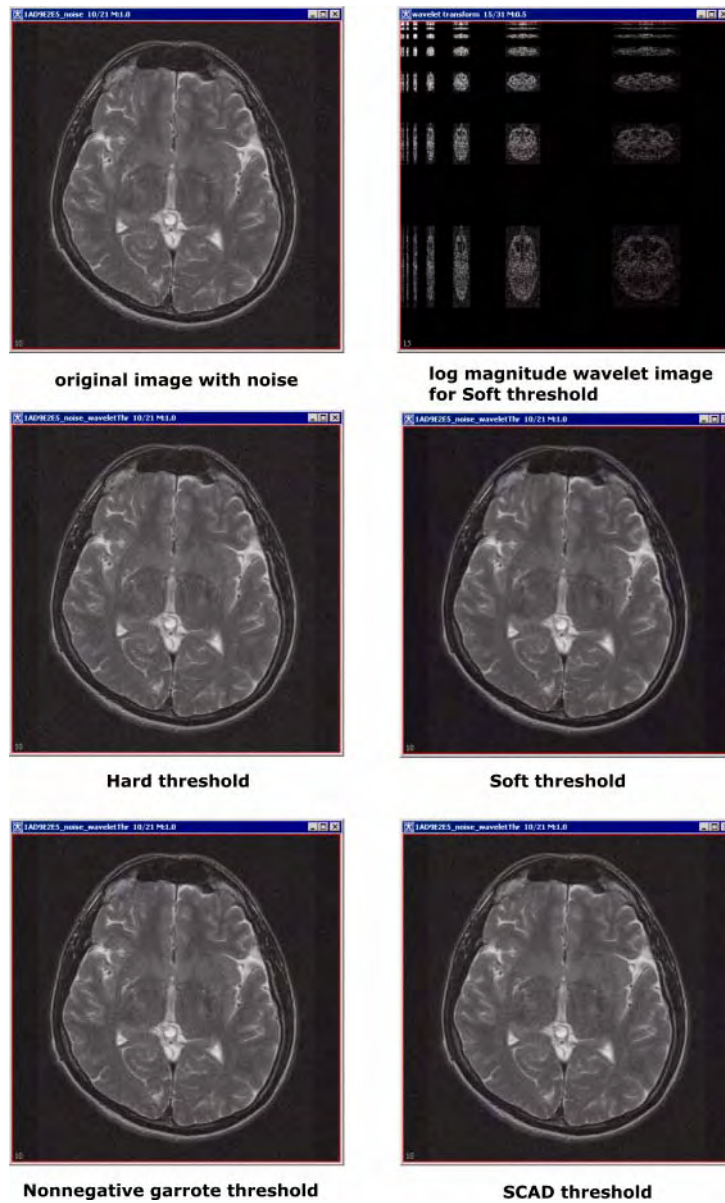
Numerical Recipes in C The Art of Scientific Computing, 2nd edition, by William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, Cambridge University Press, 1997, Chapter 13.10, pp. 591 - 606.

“Empirical Evaluation of Boundary Policies for Wavelet-based Image Coding” by Claudia Schremmer, Book Series Lecture Notes in Computer Science, Springer Berlin / Heidelberg, ISSN0302-9743 (Print) 1611-3349 (Online) January 01, 2001.

Wavelet Estimators in Nonparametric Regression: A Comparative Simulation Study by Anestis Antoniadis, Jeremie Bigot, Theofanis Sapatinas, Journal of Statistical Software, American Statistical Association, Vol. i06, 2006 <http://www.jstatsoft.org/v06/i06>.

Advanced Computer Organization Stream Processor Architecture, Homework #1, Stanford University, <http://cva.stanford.edu/classes/ee482s/homework/hw1.pdf>.





**Figure 2.** The sample images after applying the Filters Wavelet Threshold algorithm. All options were applied with the default dialog box settings which are Number of terms =4, Wavelet threshold=1.0E-4

---

## IMAGE TYPES

2D and 3D black and white images.

---

## Applying the Wavelet Thresholding algorithm

To use this algorithm, do the following:

**1** To apply the algorithm, do the following:

**2** Open an image of interest.

**3** Navigate to Algorithms > Filters Wavelet > Thresholding.

**4** In the Wavelet Thresholding dialog box that appears, complete the following fields:

- In the Number of Terms list box, select 4, 12 or 20. Numbers of Terms here means the number of wavelet vanishing moments or an approximation order;
- In the Wavelet Threshold box, specify the threshold value;
- To specify the threshold type, use the radio buttons: Hard, Soft, Nonnegative garrote, and SCAD;
- To display the log magnitude wavelet image, check the corresponding box. Displaying this image will help you to adjust the algorithm settings;
- Select the destination, use the New Image option to view the result image in a new frame, or use the Replace Image option to view the result image in the same frame.

**5** Press OK. The algorithm begins to run and the new image appears in the specified frame.


<b>Number of terms</b>	– this is the number of wavelet vanishing moments or an approximation order. Select 4, 12 or 20.	
<b>Wavelet threshold</b>	– use to specify the threshold value	
<b>Hard</b>	– use these radio buttons specify the threshold type.	
<b>Soft</b>		
<b>Nonnegative garrote</b>		
<b>SCAD</b>		
<b>Display log magnitude wavelet image</b>	– check this box to display the log magnitude wavelet image. Displaying this image will help you to adjust the algorithm settings.	
<b>New image</b>	use this to view the result image in a new frame.	
<b>Replace image</b>	use this to view the result image in the same frame.	
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.	
<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes this dialog box.	
<b>Help</b>	Displays online help for this dialog box.	

Figure 3. Wavelet Thresholding dialog box

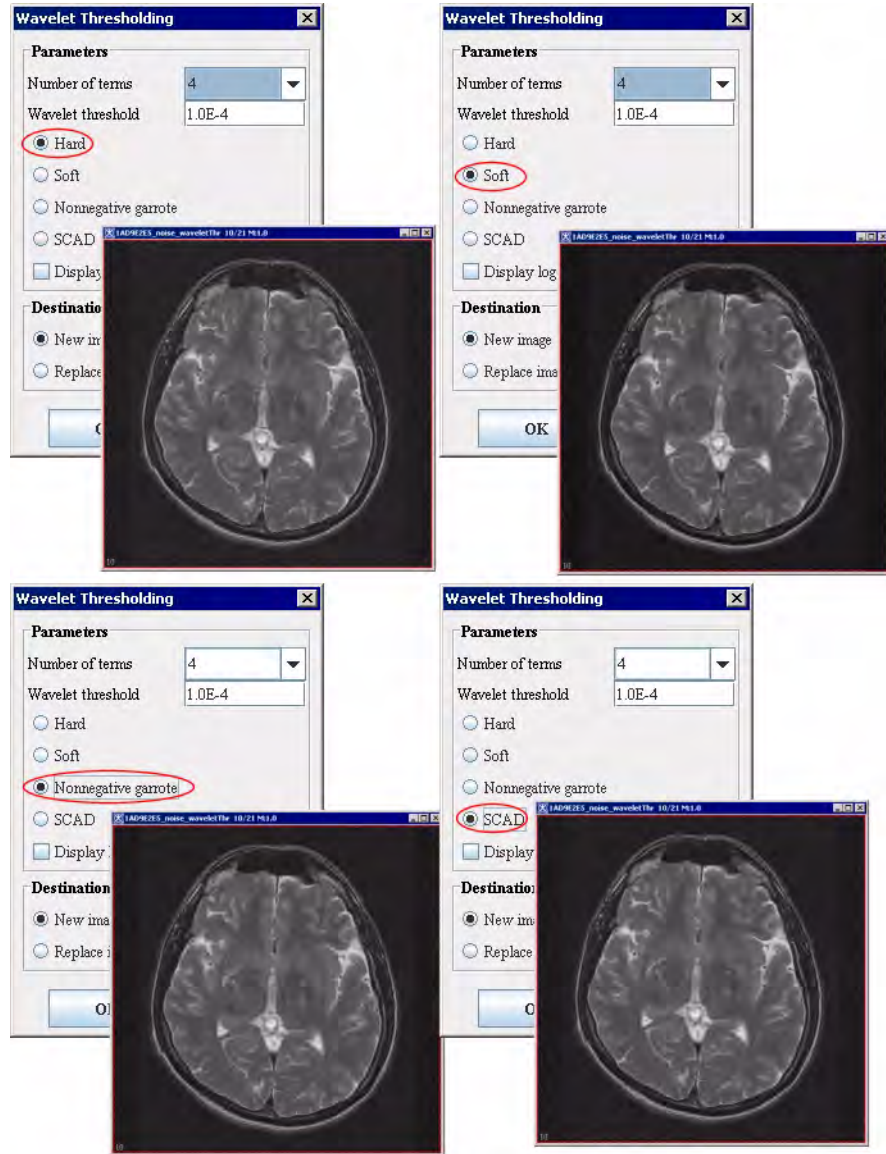


Figure 4. Applying the Wavelet Thresholding algorithm

---

# Fuzzy C-Means: Multispectral and Single Channel Algorithms

Segmentation involves dividing an image into distinct classes or types. For example, researchers may segment the brain into three classes: gray matter, white matter, and cerebrospinal fluid. There are two types of segmentation: *hard* and *soft*, or *fuzzy*.

---

## HARD SEGMENTATION

A pixel is assigned to only one class. In medical images, absolute classification of a pixel is often not possible because of partial volume effects where multiple tissues contribute to a pixel or because a voxel causes intensity blurring across boundaries.

---

## SOFT, OR FUZZY, SEGMENTATION

This type allows for the uncertainty in the location of object boundaries. In fuzzy segmentation, a membership function exists for each class at every pixel location. At each pixel location a class membership function has the value of zero if the pixel does not belong to the class. At each pixel location a class membership function has a value of 1 if the pixel belongs, with absolute certainty, to the class. Membership functions can vary from 0 to 1, with the constraint that at any pixel location the sum of the membership functions of all the classes must be 1. The fuzzy membership function reflects the similarity between the data value at that pixel and the value of the class centroid. As a pixel data value becomes closer to the class centroid, the class membership function approaches unity.

## Background

The Fuzzy C-Means algorithm is an *unsupervised* method. That is, it works without the use of a specialized set of data for determining parameters of an algorithm. This algorithm, which allows for fuzzy segmentation based on fuzzy set theory, generalizes the K-Means algorithm. The technique clusters

data by iteratively computing a fuzzy membership function and mean value estimate for each tissue class. The algorithm minimizes the sum over all pixels  $j$  and all classes  $k$  of:

$$J = \sum_{j \in \Omega} \sum_{k=1}^C u_{jk}^q \|y_j - v_k\|^2 \quad (\text{EQ 1})$$

where

$u_{jk}$  is the membership value at pixel location  $j$  for class  $k$  such that the sum over  $k$  from  $k = 1$  to  $k = C$  ( $nClass$ ) for  $u_{jk} = 1$ .

$q$  is a weighing exponent on each membership value and determines the amount of “fuzziness” of the resulting segmentation.  $q$  must be greater than 1, is typically set to 2.

$y_j$  is the observed single channel or multispectral image intensity at location  $j$ .

$v_k$  is the centroid of class  $k$

$C$  is a number of classes ( $nClass$ )

Users provide initial centroid values or simply use default evenly spread pixel values generated by:

for ( $i = 0$ ;  $i < nClass$ ;  $i++$ )

centroid[ $i$ ] = minimum + (maximum - minimum)\*( $i + 1$ )/( $nClass + 1$ );

Minimization is achieved by an iterative process that performs two computations. First, the minimization process computes the membership functions using a current estimate of the centroids.

In the single channel case (select Algorithms > Segmentation > Fuzzy C-means > Single channel in the MIPAV window):

$$u_{jk} = \frac{\|y_j - g_j v_k\|^{-2/(q-1)}}{\sum_{k=1}^C \|y_j - g_j v_k\|^{-2/(q-1)}} \quad (\text{EQ 2})$$

The minimization process also computes the centroids using current

estimates of the membership functions.

In the single channel case:

$$v_k = \frac{\sum_{j \in \Omega} u_{jk}^q g_j y_i}{\sum_{j \in \Omega} u_{jk}^q g_j^2} \quad (\text{EQ 3})$$

In the multispectral case:

$$v_k = \sum_s \sum_{k=1}^C \frac{\sum_{j \in \Omega} u_{jk}^q g_j y_i}{\sum_{j \in \Omega} u_{jk}^q g_j^2} \quad (\text{EQ 4})$$

where summation is over all classes  $k$  and all images  $s$

The iteration continues until the user-specified maximum number of iterations occurs or until convergence is detected. Convergence occurs when all membership functions over all pixel locations  $j$  change by less than the tolerance value between two iterations. The default maximum iteration number is 100 for the single channel case and 200 for the multispectral case. The default tolerance is 0.01.

After entering the parameters for the algorithm in the Fuzzy C-Means dialog box (Figure 3), researchers select one of three choices for the output images:

- HARD ONLY
- FUZZY ONLY
- HARD & FUZZY BOTH

Hard segmentation produces only one unsigned-byte output image. Pixels that do not meet threshold requirements are assigned the value of 0. The first class is assigned a value of  $255/(nClass)$ ; the second class is assigned a pixel value of  $2 \times 255/(nClass)$ , and so on. The last class has a value of 255.

Fuzzy segmentation produces one image, of the same type as the first source image, for every segmentation class. The membership function is scaled so that the minimum membership value scales to the first source image minimum value and the maximum membership value scales to the first source image maximum value. If boundary cropping is applied, all pixels outside the bounding box are assigned the minimum value of the first source image.

If researchers apply the multispectral version of the algorithm, they can add images to a list.

---

## ADDING IMAGES TO THE LIST

To add an image to the list:

- 1 Click Load. The Open dialog box appears.
- 2 Select an image.

If the following are true, MIPAV adds the image to the list:

- The image has the same dimensionality as the original image.
- The length of each dimension is the same as that of the original image.

Otherwise, an error message appears.

## Removing images from the list

To remove an image from the list, click Remove.

---

**Note:** You cannot remove the original image (the first image in the list) that was present at the time you selected Algorithms > Fuzzy C-means > Multispectral. The Remove Image button is only enabled (active) when at least two images are in the list.

---

The multispectral case allows color images to be loaded. If the original image is color, then the initial dialog box includes, by default, three selected check boxes for red, green, and blue components. If the original image is not color, then the later loaded images cannot be color.



With the single channel case, the signal threshold value is entered on the initial dialog box and the image centroids are entered on a later dialog box. The multispectral case does not have signal threshold on the initial dialog box. With the multispectral case, a later dialog box appears for every black-and-white image or for every selected component of every color image and the signal thresholds and initial centroids are entered on these later dialog boxes.

---

## IMAGE TYPES

The Fuzzy C-means algorithm runs on 2D and 3D datasets and can also be applied to RGB datasets. This algorithms cannot run to complex datasets.

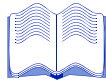
---

## SPECIAL NOTES

None.

---

## REFERENCES



Refer to the following references for more information about the Fuzzy C-means algorithm.

Dzung L. Pham, Chenyang Xu, and Jerry L. Prince, "A Survey of Current Methods in Medical Image Segmentation," Technical Report JHU/ECE 99-01 (Baltimore: Department of Electrical and Computer Engineering, The Johns Hopkins University, 2001).

Alberto F. Goldszal and Dzung L. Pham, "Volumetric Segmentation," Chapter 12 in *Handbook of Medical Image Processing*, (San Diego: Academic Press, 2000).

Dzung L. Pham and Jerry L. Prince, "Adaptive Fuzzy Segmentation of Magnetic Resonance Images," *IEEE Transactions on Medical Imaging*, Vol. 18, No. 9, September 1999, pp. 737-752.

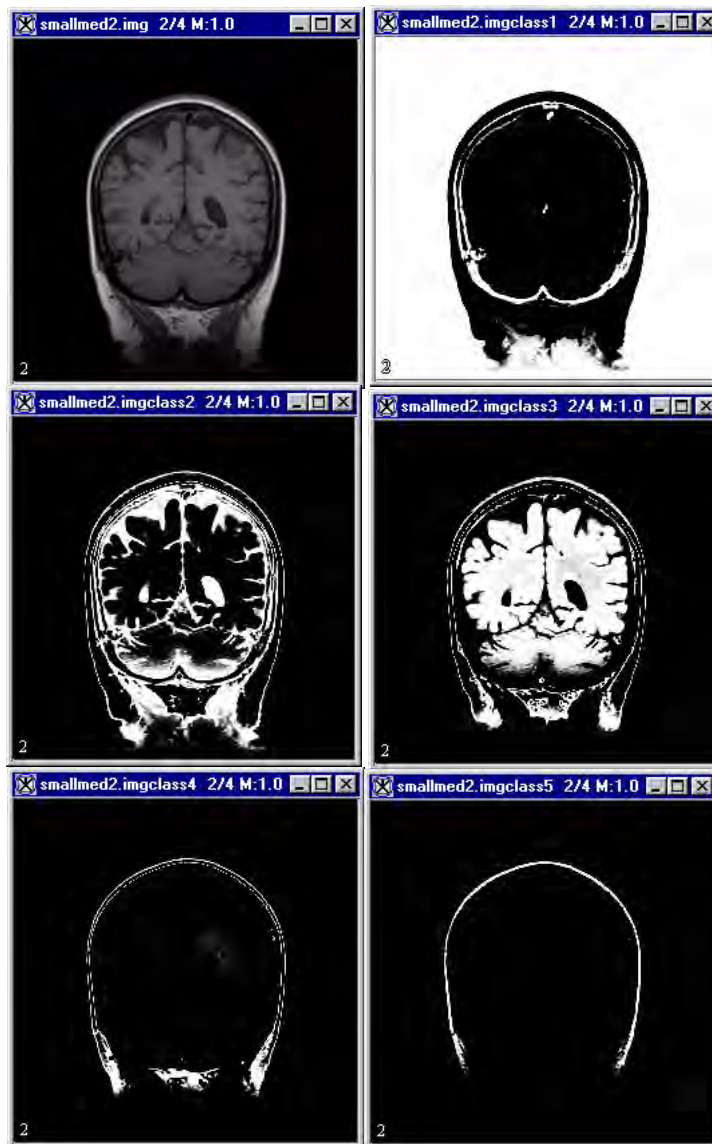


Figure 1. Fuzzy C-means algorithm processing

## Applying the Fuzzy C-means algorithm

You can apply this algorithm on single images separately, or you can apply it on multispectral images.

## ON SINGLE IMAGES

To run this algorithm, complete the following steps:

- 1** Start MIPAV if not already started. The MIPAV and the Output windows open.
- 2** Open an image. After a few moments, the image appears.
- 3** Select Algorithms > Fuzzy C-means > Multispectral. The Fuzzy C-means dialog box (Figure 2) opens.
- 4** Complete the fields.
- 5** Click OK.

The Threshold & Initial Centroids dialog box appears.

By default, MIPAV determines the value of the initial centroids by result of the total number of intensities divided by the number of classes. To change the values, click OK.

### Number of desired classes

Indicates the number of groups into which the algorithm divides the dataset structures.

By default, the number of desired classes is 3.

### Desired exponent value

Determines the amount of "fuzziness" in the resulting segmentation.

By default, the exponent value is 2.

### End tolerance

Indicates when convergence is reached. Generally, membership functions over all pixel locations change between iterations of the application of the fuzzy C-means algorithm. Convergence occurs if the change is equal to or less than the end tolerance level and when all membership functions over all pixel locations change by less than the tolerance value between two iterations. By default, the tolerance is 0.01.

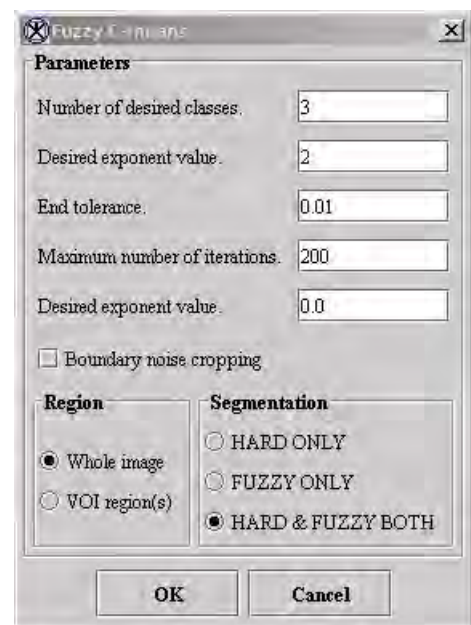


Figure 2. Fuzzy C-Means dialog box for a single image

<b>Maximum number of iterations</b>	Indicates the maximum number of times the algorithm is applied. If convergence occurs, the system may apply the algorithm less times than the maximum number of iterations. By default, the maximum iteration value for single channel cases is 200.
<b>Signal treshold</b>	Determines the treshold. By default, the exponent value is 0.0.
<b>Background cropping</b>	Finds the smallest bounding box in the image or delineated VOI outside of which all image pixel values are below the image threshold.
<b>Whole image</b>	Applies the algorithm to the whole image.
<b>VOI region(s)</b>	Applies the algorithm only to the delineated VOIs.
<b>HARD ONLY</b>	Performs only hard segmentation on the image
<b>FUZZY ONLY</b>	Performs only fuzzy segmentation on the image.
<b>HARD &amp; FUZZY BOTH</b>	Performs both types of segmentation—hard and soft—on the image.
<b>OK</b>	Performs the Fuzzy C-Means algorithm on the image based on your choices in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this algorithm.

Figure 2. Fuzzy C-Means dialog box for a single image (continued)

## ON MULTISPECTRAL IMAGES

To run this algorithm, complete the following steps:

- 1** Start MIPAV if not already started. The MIPAV and the Output windows open.
- 2** Open an image. After a few moments, the image appears.
- 3** Select Algorithms > Fuzzy C-means > Multispectral. The Fuzzy C-means dialog box (Figure 3) opens.
- 4** Complete the fields.

To add an image to the list, click Load and then select the image in the Open dialog box. The system then adds the image to the list and then, if it has the same dimensions as the original image, opens it in a new window.

To remove an image from the list, click Remove.

**5** Click OK.

The Threshold & Initial Centroids dialog box appears.

By default, MIPAV determines the value of the initial centroids by result of the total number of intensities divided by the number of classes. To change the values, click OK.



**Note:** If you are applying this algorithm to several datasets, when you click OK another Threshold & Initial Centroids window appears sequentially for each dataset. Click OK when complete. MIPAV applies the algorithm to the datasets. The resultant images appear in new image windows.

<b>Number of desired classes</b>	Indicates the number of groups into which the algorithm divides the dataset structures.
<b>Desired exponent value</b>	Determines the amount of "fuzziness" in the resulting segmentation.
<b>Boundary noise cropping</b>	Finds the smallest bounding box outside of which all first image pixel values are below the first image threshold. To save space, MIPAV copies values inside the bounding box to a smaller array and then uses the reduced array to performs calculations.
<b>Signal threshold</b>	<p>Appears only when you select Algorithm &gt; Fuzzy C-means &gt; Single Channel.</p> <p>By default, MIPAV assigns the image minimum value and uses only pixels whose values equal or exceed the threshold used in the centroid calculation.</p> <p>If you select the HARD ONLY option, MIPAV sets the pixels whose values are less than the threshold to a segmentation value of 0. This means that the pixels are outside of the specified classes.</p>

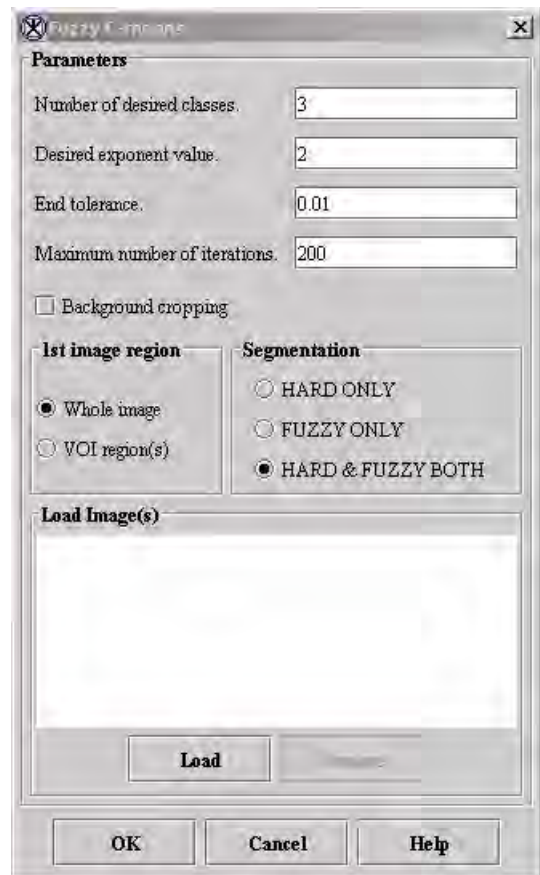


Figure 3. Fuzzy C-Means dialog box for multispectral images

<b>End tolerance</b>	Indicates when convergence is reached. Generally, membership functions over all pixel locations change between iterations of the application of the fuzzy c-means algorithm. Convergence occurs if the change is equal to or less than the end tolerance level and when all membership functions over all pixel locations change by less than the tolerance value between two iterations. By default, the tolerance is 0.01.
<b>Maximum number of iterations</b>	Indicates the maximum number of times the algorithm is applied. If convergence occurs, the system may apply the algorithm less times than the maximum number of iterations. By default, the maximum iteration value for single channel cases is 100. The maximum number in multispectral cases is 200.
<b>Background noise cropping</b>	Finds the smallest bounding box in the image or delineated VOI outside of which all image pixel values are below the image threshold.
<b>Whole image</b>	Applies the algorithm to the whole image.
<b>VOI region(s)</b>	Applies the algorithm only to the delineated VOIs.
<b>HARD ONLY</b>	Performs only hard segmentation on the image
<b>FUZZY ONLY</b>	Performs only fuzzy segmentation on the image.
<b>HARD &amp; FUZZY BOTH</b>	Performs both types of segmentation—hard and soft—on the image.
<b>Load</b>	Loads the images that you select.
<b>Remove</b>	Removes the images that you select.
<b>OK</b>	Performs the FuzzyC-Means algorithm on the selected images based on your choices in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this algorithm.

Figure 3. Fuzzy C-Means dialog box for multispectral images (continued)

## 2D Histogram

The 2D Histogram algorithm takes as an input, two grayscale images or one color image to create a 2D histogram image based on the data in two input images.

### Background

The active image window is the source image whose frequency values are placed along the x-axis of the 2D histogram image. The second image forms the base image whose frequency values are placed along the y-axis of the 2D histogram image.

If the color image is an input to the algorithm, the user must select which two color channels should be used for processing.

The user has the option to linearly rescale the second image for both grayscale and color images. If true, the range of data in second image is rescaled to be the same as the range of data in the first image. In other words, the second image minimum is rescaled to the first image minimum and the second image maximum is rescaled to the first image maximum. Also, if linear rescaling is used the number of bins in the second image is set equal to the number of bins in the first image.

- The default number of bins equals the minimum of 256 and (*max. value - min. value + 1*) if the data is not float.
- If the data is float or double the default number of bins is set to 256.

The processing of maximum values is specially handled to prevent loss of top bin counts from rounding errors.

The user has the option to calculate the log of result image for better visualization.

Figure 3 on page 339 shows the two input images, the output 2D histogram image and its lookup table.

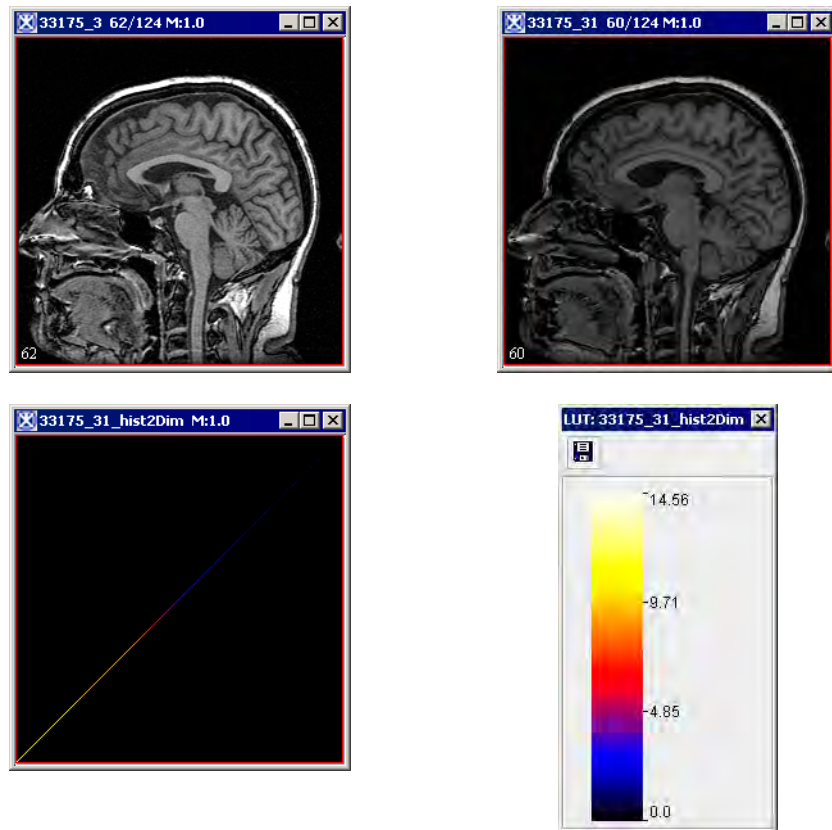


Figure 1. Two input images, output 2D histogram image and its lookup table

---

## IMAGE TYPES

You can apply this algorithm to both color and black-and-white images, as well as 2D and 3D images.

---

## NOTES

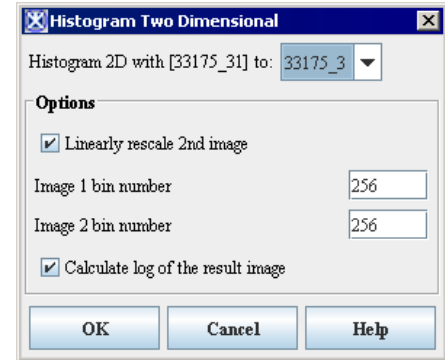
No notes.



## Applying the 2D Histogram algorithm to grayscale images

- 1 Select Algorithms > Histogram Tools > 2D Histogram in the main MIPAV window. The Histogram Two Dimensional dialog box appears.
- 2 Select the base image from the drop-down list.
- 3 Check the *Linearly rescale 2nd image* checkbox if you want to rescale the 2-nd image. By default, the box is checked.
- 4 Enter the appropriate bin number for Image1 and Image2. The default value for both images is 256.
- 5 Check the *Calculate log of the result image* checkbox if you want to calculate the log of the result image for better visualization. By default, the box is checked. Refer to Figure 2 on page 337 for details.

<b>Histogram 2D with &lt;File name&gt; to:</b>	Select the base image from the list box.
Options	
<b>Linearly rescale 2-nd image</b>	Use this checkbox if you want to rescale the 2-nd image. By default, the box is checked.
<b>Image 1 bin format</b>	Enter the bin number for the image 1. The default value is 256.
<b>Image 2 bin format</b>	Enter the bin number for the image 2. The default value is 256.
<b>Calculate the log of the result image</b>	Check if you want to calculate the log of the result image for better visualization. By default, the box is checked.
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes this dialog box.
<b>Help</b>	Displays online help for this dialog box.



**Figure 2.** The Histogram Two Dimensional dialog box for grayscale images

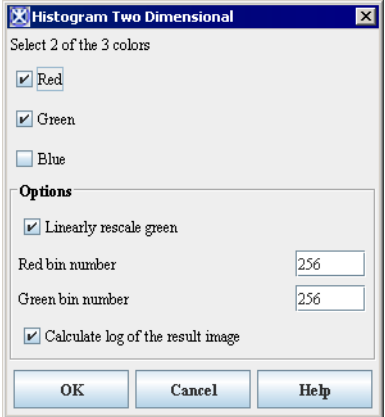
- 6 Click OK. The algorithm begins to run, and a progress bar appears momentarily with the status. When the algorithm finishes running, the progress bar disappears, and the results appear in a separate image window.

## Applying the 2D Histogram algorithm to color images

To run this algorithm for color images, complete the following steps:

- 1 Select Algorithms > Histogram tools > 2D Histogram.
- 2 Use the checkbox to select any two colors from red, green, and blue. Note that, only two colors can be selected.
- 3 Check the *Linearly rescale 2nd image* checkbox if you want to rescale the 2-nd image. By default, the box is checked.
- 4 Enter the appropriate bin number for the two colors selected. The default values is 256 for both colors.
- 5 Check the *Calculate log of the result image* checkbox if you want to calculate the log of the result image for better visualization. By default, the box is checked.
- 6 Click OK. See Figure 3 on page 339.

The algorithm begins to run, and the pop-up window appears to run with the status. The following message appears: “Processing Image”. When the algorithm is completed the pop-up window closes and the 2D histogram image appears in the new window.

<p><b>Select 2 of 3 colors</b></p>	<p>Use the checkbox to select any two colors from red, green, and blue. Note that, only two colors can be selected.</p>	
<p><b>Options</b></p>		
<p><b>Linearly rescale</b></p>	<p>Use this checkbox if you want to rescale the 2-nd color. By default, the box is checked.</p>	
<p><b>Color 1 bin number</b></p>	<p>Enter the bin number for color 1. The default value is 256.</p>	
<p><b>Color 2 bin number</b></p>	<p>Enter the bin number for color 2. The default value is 256.</p>	
<p><b>Calculate the log of the result image</b></p>	<p>Check if you want to calculate the log of the result image for better visualization. By default, the box is checked.</p>	
<p><b>OK</b></p>	<p>Applies the algorithm according to the specifications in this dialog box.</p>	
<p><b>Cancel</b></p>	<p>Disregards any changes that you made in the dialog box and closes this dialog box.</p>	
<p><b>Help</b></p>	<p>Displays online help for this dialog box.</p>	
<p><b>Figure 3. The Histogram Two Dimensional dialog box for color images</b></p>		

## Cumulative Histogram

*The cumulative histogram is a histogram in which the vertical axis gives not just the counts for a single bin, but rather gives the counts for that bin plus all bins for smaller values of the response variable.*

### Background

Let  $I:Q \rightarrow [0,1]$  be an image and  $h$  is the distribution of its values (a histogram). The *cumulative histogram* of  $I$  is the function  $H: [0,1] \rightarrow [0,1]$  defined by

EQUATION 1

$$H(x) = \int_0^x h(t) dt$$

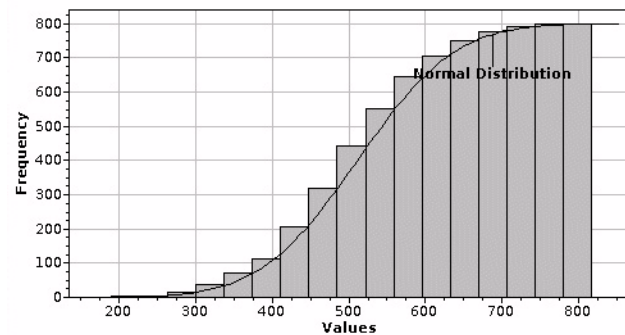


Figure 1. A sample cumulative histogram

### EXAMPLES

Following examples display cumulative histograms for grayscale and RGB images. Figure 2 shows a 3D grayscale image and its cumulative histogram. Figure 3 and Figure 4 show 2D RGB images and their cumulative histograms for each channel. Note that, the RGB image from Figure 3 has in fact

only two color channels - Red and Blue, therefore its cumulative histogram for the Green channel shows nothing.

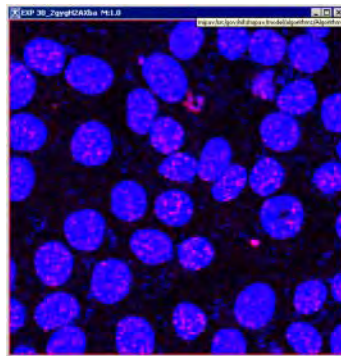


a

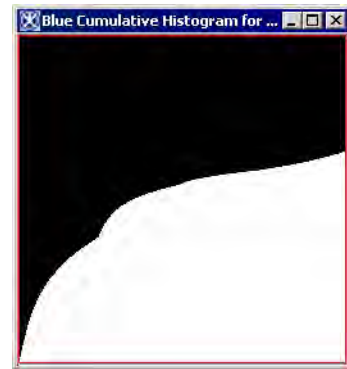


b

Figure 2. A 3D grayscale image (a) and its cumulative histogram (b).



a



b

Figure 3. A 2D RGB image and its cumulative histograms for each channel.



Figure 3. A 2D RGB image and its cumulative histograms for each channel.

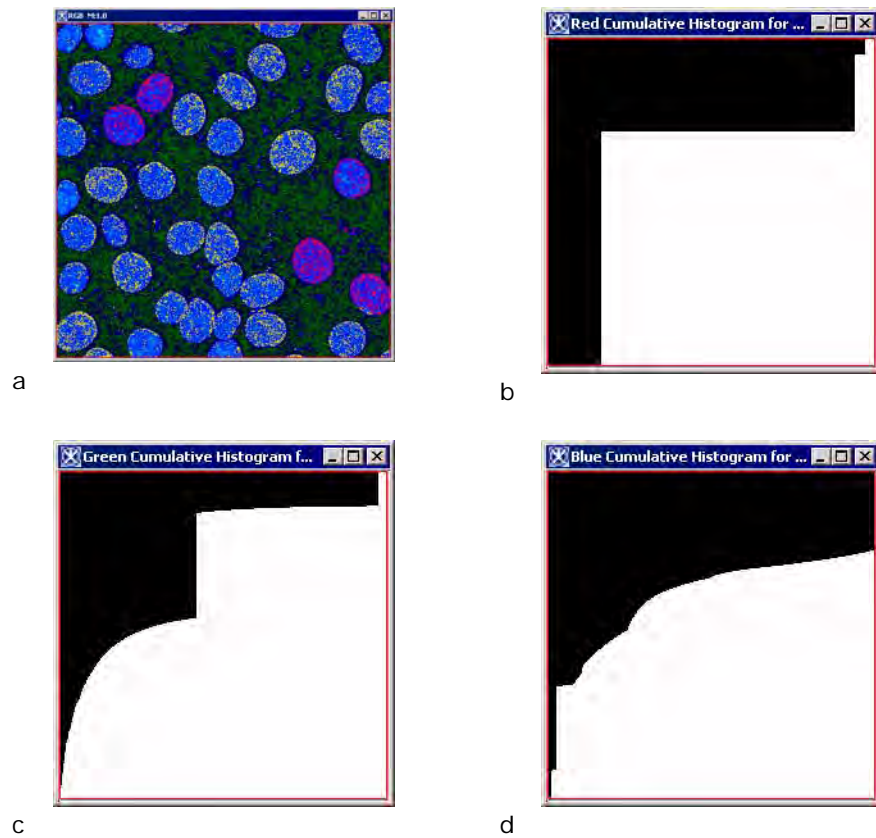


Figure 4. A 2D RGB image and its cumulative histograms .

---

## NOTES

No notes.

---

## REFERENCES

J. Delon, Midway Image Equalization, Journal of Mathematical Imaging and Vision, vol.21, no.2, pp.119-134, 2004.

Understanding Histograms in Digital Photography, <http://www.luminous-landscape.com/tutorials/understanding-series/understanding-histograms.shtml>

---

## IMAGE TYPES

You can apply the Cumulative Histogram algorithm to 2D, 2.5D and 3D grayscale and color (RGB) images. For RGB images the algorithm will display a separate cumulative histogram for each channel (R, G, and B).

## Applying Cumulative Histogram algorithm to images

---

### TO APPLY CUMULATIVE HISTOGRAM TO GRAYSCALE IMAGES:

- 1** Open an image of interest.
- 2** In the main MIPAV window, navigate to Algorithms > Histogram Tools > Cumulative Histogram.
- 3** The algorithm starts running and the cumulative histogram appears in a new image frame. See Figure 2.

## TO APPLY CUMULATIVE HISTOGRAM TO RGB IMAGES

- 1** Open an image of interest.
- 2** Call Algorithms > Histogram Tools > Cumulative Histogram from the main MIPAV menu.
- 3** The algorithm starts running and the cumulative histograms for each channel (R,G, and B) appear. See Figure 3 and Figure 4.



## Histogram Equalization: Regional Adaptive

The Histogram Equalization: Regional Adaptive algorithm, a high-pass filter, enhances the contrast in an image by reevaluating the gray-scale, or intensity, value of each pixel based on the values of nearby pixels in the same region.

### Background

Features in some images require a larger range of intensity values than are displayed. Histogram equalization is a method to improve the contrast of an area in an image by distributing an equal number of pixels across the range of intensities in the image. This algorithm tabulates the histogram for each region, then assigns the pixel to the new histogram level.

This algorithm acts in regions of an image, dividing the image into  $m$  rows and  $n$  columns. Each region is the source when remapping the histogram, adapting the value of the pixel to its location in the histogram of the region.



**bin width**—The permitted range of floating-point values of a pixel to equate to an intensity level in a 256-color image.

Although, depending on image type, there are some minor variations in how this algorithm proceeds, generally the algorithm first calculates the *bin width* of each region. It then processes each region by judging each pixel's location within this new histogram. Regardless of pixel value, the algorithm remaps the brightest pixel in each region to the brightest value in the histogram (i.e., 256) and remaps the darkest pixel to the lowest value in the histogram (i.e., 0) of the pixels' neighboring region, recording the new values as pixel-intensity values. The algorithm then processes each of the remaining regions in the original, or *source*, image in the same way and builds a new, or *reference*, image using the new pixel-intensity values.

Performing histogram equalization makes it possible to see minor variations within portions of the reference image that appeared nearly uniform in the source image. Arithmetically, the process in a region is rather simple:

$$k = \sum_{i=0}^j \frac{N_i}{T}$$

where  $N_i$  is the intensity of the  $i$ th pixel, and  $T$  is the total number of pixels in the region.

The reference image using the new histogram is similar to the source image; however, in areas where the source image had similar values, the reference image is enhanced. The reference histogram now stretches over a greater proportion of the possible image intensities (Figure 1).

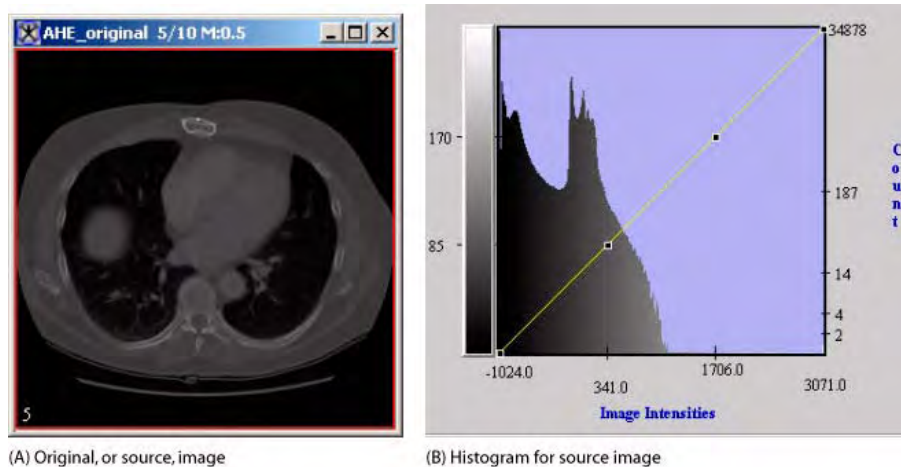
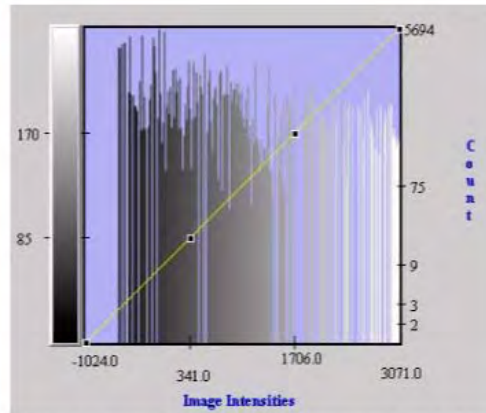


Figure 1. Source image and histogram and reference images and histograms for reference images that were separated into 1 x 1, 2 x 2, and 3 x 3 regions



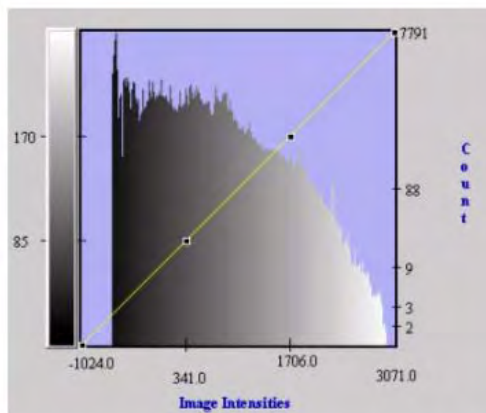
(C) Reference image that was separated into 1 x 1 regions



(D) Histogram for reference image that was separated into 1 x 1 regions



(E) Reference image that was separated into 2 x 2 regions



(F) Histogram for reference image that was separated into 2 x 2 regions

Figure 1. Source image and histogram and reference images and histograms for reference images that were separated into 1 x 1, 2 x 2, and 3 x 3 regions (continued)

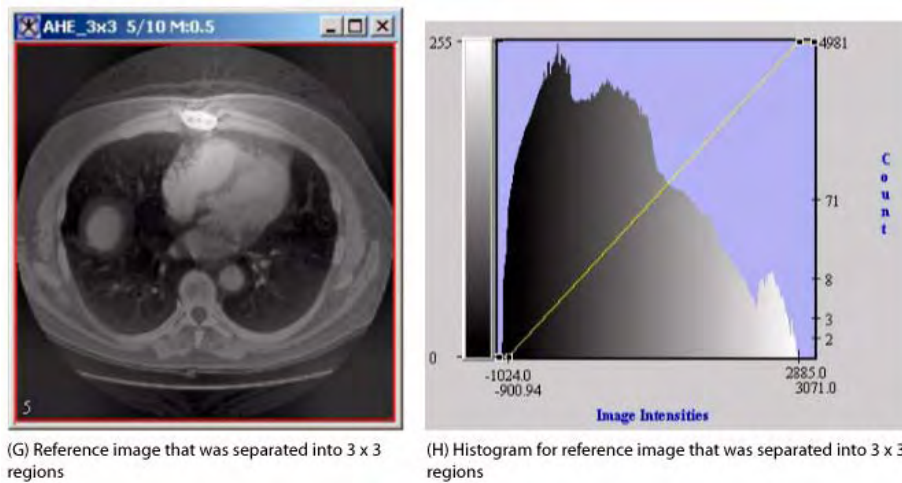


Figure 1. Source image and histogram and reference images and histograms for reference images that were separated into 1 x 1, 2 x 2, and 3 x 3 regions (continued)



**Note:** All histograms shown in this section are in log scale in the vertical axis.

The histogram for the source image is very compact in structure; that is, it has a great number of darker pixels, but very few brighter pixels. In fact, the bottom-most intensity values make up almost all of the pixels' values in the image. Note, however, in the histogram of the reference image, the look-up table is more evenly spread than that of the histogram for the source image. Also, areas of low intensity and low contrast in the source image were correspondingly remapped in the reference image to areas of higher contrast and generally overall brighter values.

Notice that different low areas of the reference image are more distinguishable depending on the number of rows and columns that divide the image. The contrast changes are due to which pixel is considered the brightest and which the darkest in a particular region. The brightest pixel in a 1 x 1 separated image is the brightest of the image. However, when the image is divided into more regions, the brightest pixel in the region gets remapped as one of the brightest pixels in the image regardless of its absolute intensity in the image. Likewise, the darkest pixel in the region is remapped as one of the darkest in the image regardless of its absolute intensity.

The histogram equalization process can enforce certain characteristics about the image called *clipping*. As the algorithm processes the image, it counts out the number of pixels of a certain intensity. Clipping cuts that counting short; it stores the excess pixels of a particular intensity and redistributes that number equally among all intensities. For instance, if a particular intensity must be clipped at 100 and the count is 145, the algorithm redistributes 45 pixels to other brightnesses. It adds to the count of each intensity level an equal number of pixels if there are enough excess, but it attempts to spread the excess over as many pixels as possible. Figure 2 shows examples of the source image from Figure 1 that was separated into 1 x 1 regions with no clipping, 45-percent clipping, and 75-percent clipping.

## IMAGE TYPES

You can apply this algorithm to both color and black-and-white images.

## SPECIAL NOTES

This algorithm only works on the whole image, not on volumes of interest (VOIs).

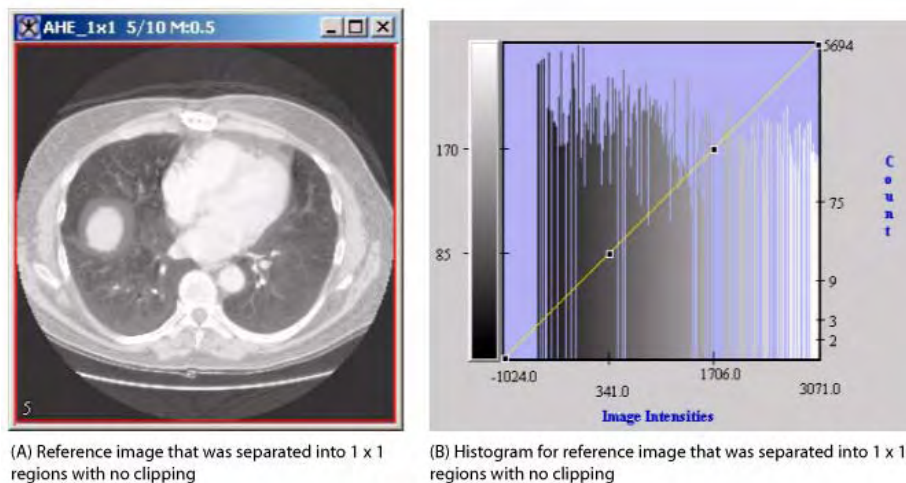
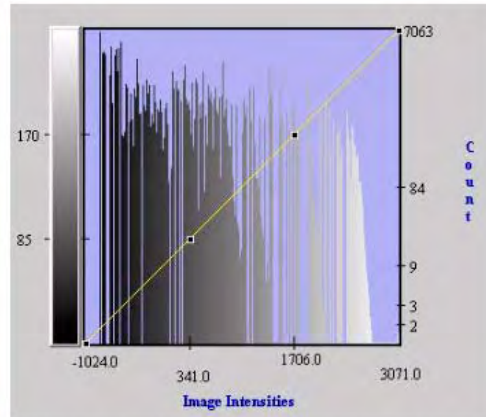


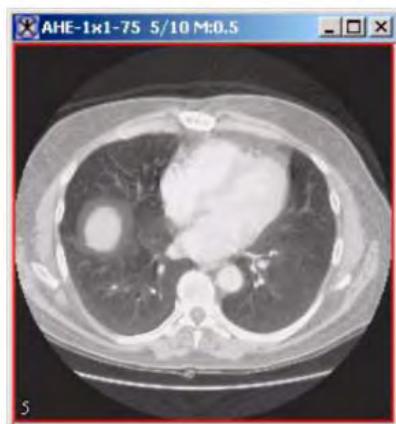
Figure 2. Examples of no clipping, 45-percent clipping, and 75-percent clipping



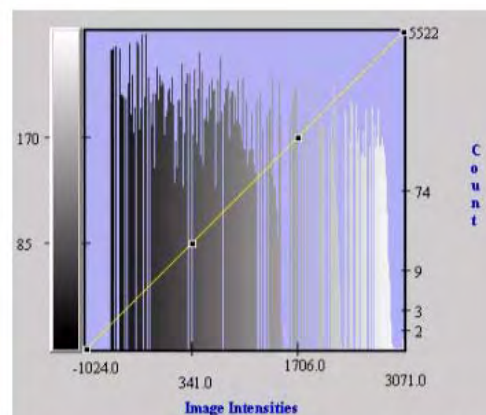
(C) Reference image that was separated into 1 x 1 regions with 45-percent clipping



(D) Histogram for reference image that was separated into 1 x 1 regions with 45-percent clipping

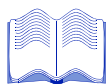


(E) Reference image that was separated into 1 x 1 regions with 75-percent clipping



(F) Histogram for reference image that was separated into 1 x 1 regions with 75-percent clipping

Figure 2. Examples of no clipping, 45-percent clipping, and 75-percent clipping



## REFERENCES

Refer to the following references for more information:

Bob Cromwell, "Localized Contrast Enhancement: Histogram Equalization," paper on his web site at <http://www.cromwell-intl.com/3d/histogram/Index.html>.

Douglas A. Lyon, *Image Processing in Java* (Upper Saddle River, New Jersey: Prentice-Hall, 1999).

Tamer Rabie, Rangaraj Rangayan, Raman Paranjape, "Adaptive-Neighborhood Image Deblurring," The University of Calgary, Department of Electrical and Computer Engineering (Calgary, Alberta, Canada: NEC Research Institute, 2002).

John Russ, *The Image Processing Handbook* (Boca Raton, Florida: CRC Press LLC, 2003).

J. Alex Stark, "Adaptive Image Contrast Enhancement Using Generalizations of Histogram Equalization," in *IEEE Transactions on Image Processing* (May 2000).

Claes Tidestav, "Short Introduction to Adaptive Equalization." (Uppsala, Swedon: Uppsala University, April 1996).

## Applying the Histogram Equalization: Regional Adaptive algorithm

To run this algorithm, complete the following steps:

- 1** Open an image.
- 2** Perform, as an option, any other image processing on the image.
- 3** Select Algorithms > Histogram Equalization > Regional Adaptive. The Regional Adaptive Histogram Equalization dialog box (Figure 3) opens.
- 4** Complete the information in the dialog box.
- 5** Click OK. The algorithm begins to run.

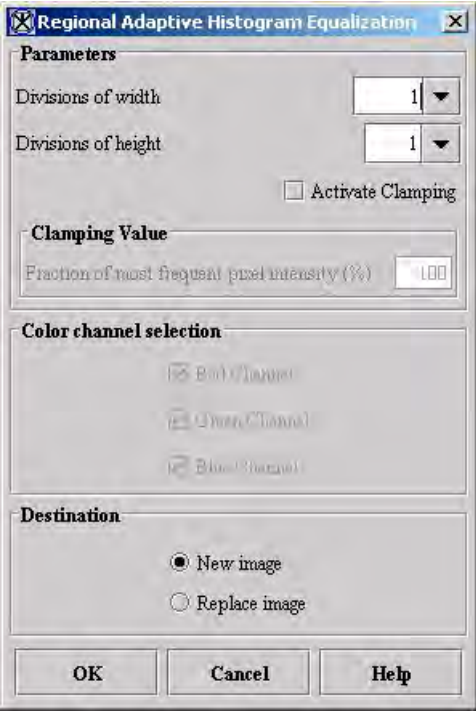
<b>Divisions of width</b>	<p>Indicates the number of times the algorithm should divide the image in width to make a region.</p> <p>If you select 2, the algorithm divides the image in width by 2 yielding two areas: a left side and a right side. Each is used separately for histogram equalization.</p>	
<b>Divisions of height</b>	<p>Indicates the number of times the algorithm should divide the image in height to make a region.</p> <p>If you select 3, the algorithm divides the image in height by 3 yielding three areas: a top, middle, and bottom part.</p>	
<b>Activate clamping</b>	<p>Allows you to specify the maximum number of pixels with a certain pixel bin.</p> <p>Selecting this check box makes Fraction of most frequent pixel intensity active.</p>	

Figure 3. The Regional Adaptive Histogram Equalization dialog box



<b>Fraction of most frequent pixel intensity (%)</b>	Indicates the percentage (1 through 99) of the maximum number of pixels per intensity allowed. The algorithm records and redistributes the pixels for any intensity that has more pixels than the calculated value. The default value is 75 percent. A lower value reduces the contrast of a region and is sometimes required to prevent parts of the image from becoming oversaturated. To use this field, first select Activate clamping.
<b>Red channel</b>	Treats all of the pixels in the red channel as its own intensity as if the image were three separate monochrome images (only applicable to color images).
<b>Green channel</b>	Treats all of the pixels in the green channel as its own intensity as if the image were three separate monochrome images (only applicable to color images).
<b>Blue channel</b>	Treats all of the pixels in the blue channel as its own intensity as if the image were three separate monochrome images (only applicable to color images).
<b>New image</b>	Shows the results of the algorithm in a new image window.
<b>Replace image</b>	Replaces the source image with the results of the algorithm.
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

**Figure 3. The Regional Adaptive Histogram Equalization dialog box (continued)**

A pop-up window appears with the status. A series of status messages appear in the window.

When the algorithm finishes running, the pop-up window closes.

Depending on whether you selected New image or Replace image, the results appear in a new window or replace the image to which the algorithm was applied.

## Histogram Equalization: Neighborhood Adaptive

The Histogram Equalization: Neighborhood Adaptive algorithm, a high-pass filter, enhances the contrast in an image by reevaluating the grayscale, or intensity, value of each pixel based on a region of nearby pixels.

### Background

Features in some images require a larger range of intensity values than are displayed. *Histogram equalization* is a method to improve the contrast of an area in an image by shifting intensity values so that there are an equal number of pixels in an image in each intensity. This algorithm tabulates the histogram for a collection of neighboring pixels, sometimes called a *kernel* and then assigns the pixel to the new histogram level.

The name of this algorithm signifies that it processes the image by looking at the neighboring pixels when performing the equalization; hence, the name *Neighborhood*, or *Local, Adaptive Histogram Equalization*. Each neighborhood of pixels is the source when remapping the histogram, adapting the value of the pixel to its location in the histogram of the neighborhood.

The algorithm first collects a list of the pixel's neighbors, calculates the *bin width* of the pixel neighborhood, then builds the histogram of the neighborhood.



**bin width**—The permitted range of floating-point values of a pixel to equate to an intensity level in a 256-color image.

The algorithm locates the value of the central pixel in the local histogram and replaces its intensity with the corresponding value in the histogram of its neighbors. The algorithm remaps the brightest pixel in a pixel neighborhood to the brightest value in the histogram (i.e., 256) and the darkest pixel to the lowest value of the histogram (i.e., 0), recording the new values as pixel-intensity values. It then processes each pixel in this way, remapping pixel value to intensity as measured against a histogram of the pixel's neighbors until the algorithm finishes checking all pixels of the image against the histogram of its neighbors and builds a new, or reference, image.

Performing histogram equalization makes it possible to see minor variations within portions of the reference image that appeared nearly uniform in the original, or source, image.

Arithmetically, the process in a neighborhood is rather simple:

$$k = \sum_{i=0}^j \frac{N_i}{T}$$

where  $N_i$  is the intensity of the  $i_{th}$  pixel, and  $T$  is the total number of pixels in the neighborhood.

Neighborhood size greatly affects the reference image, since the intensity of each resulting pixel depends on the neighborhood. The more pixels in the neighborhood that are darker than the current pixel, the greater in intensity the current pixel becomes in the reference image. Likewise, the shape of the kernel determines the number of pixels in the neighborhood.

The algorithm creates a kernel around a pixel in a region of the image as shown in Figure 1. It then ranks the intensity of that pixel as compared to the surrounding pixels. In the figure, the box is 65 pixels wide. Using a square kernel, the algorithm collects all the pixel intensities inside the square and at the edges of the square. (Using a cross kernel, however, it collects only the intensities under the orange cross-hairs.) It next ranks the pixel intensity at the center of the cross hairs against the histogram of the pixels in the kernel.

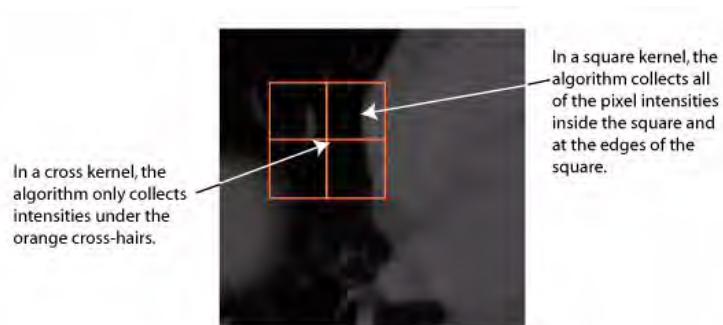


Figure 1. Sample of kernel created by algorithm

The reference image using the new histogram is similar to the source image; however, in areas where the original image had similar values, the reference image is enhanced. The output histogram now stretches over a greater proportion of the possible image intensities.

Figure 2 shows example images that were created by starting with a source image then using a 21 x 21, a 45 x 45, or 65 x 65 square kernel through the image.

The histogram of the source image is very compact in structure; that is, it has a great number of darker pixels, but very few brighter pixels. In fact, the bottom-most intensity values make up almost all of the pixels' values in the image. Note, however, in the histogram of the reference image, the look-up table is more evenly spread than in the histogram of the source image. Also, areas of low intensity and low contrast in the source image were correspondingly remapped in the reference image to areas of higher contrast and generally overall brighter values.



**Note:** The red line at the border of the image is an *active image* line, a part of the program to help the user determine which image is on top and not a part of the image itself.

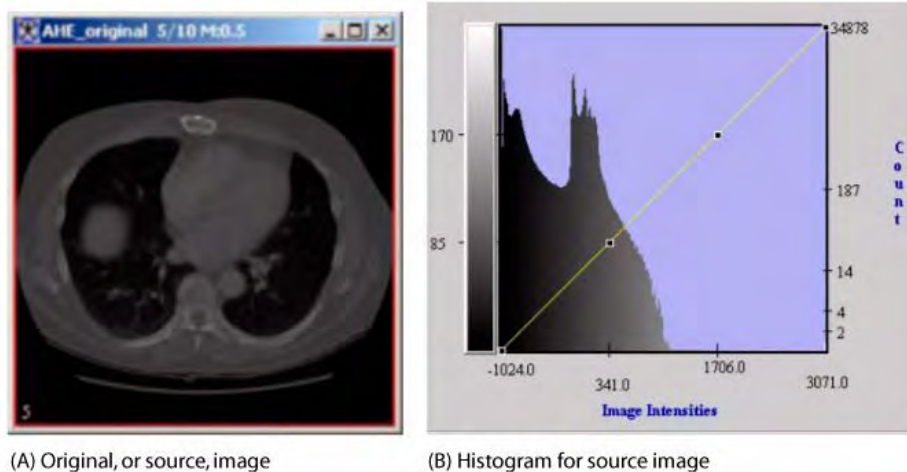
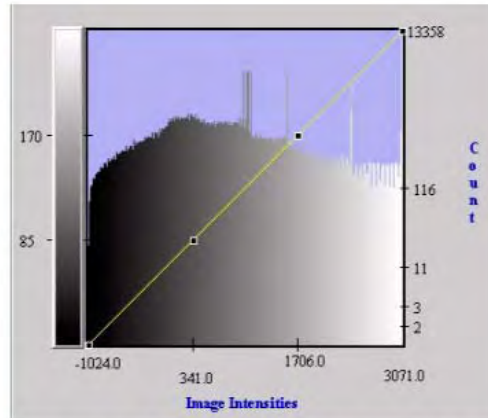


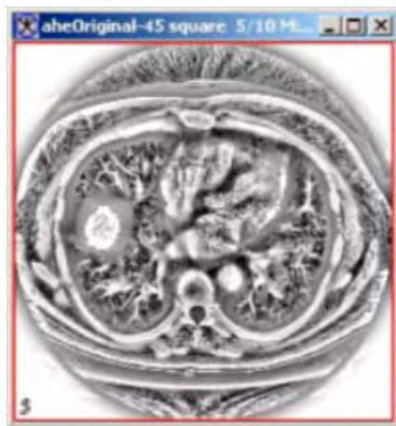
Figure 2. A source image and its histogram (A, B) compared to reference images using a 21 x 21 (C, D), 45 x 45 (E,F), or 65 x 65 (G,H) square kernels and 65 x 65 (I,J) cross kernels



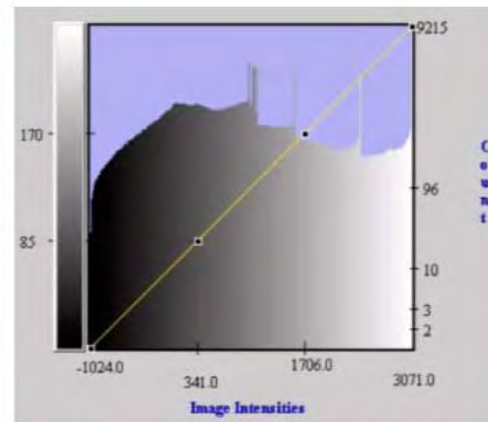
(C) Reference image using a 21 x 21 square kernel



(D) Histogram for reference image using a 21 x 21 square kernel



(E) Reference image using a 45 x 45 square kernel

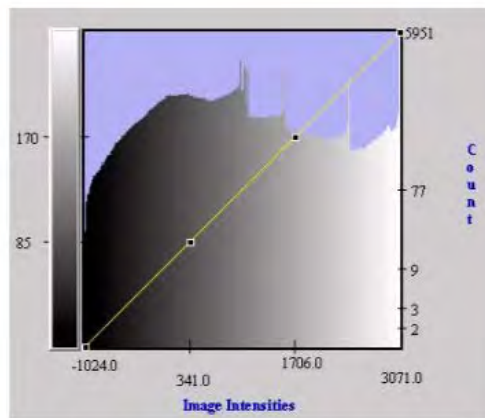


(F) Histogram for reference image using a 45 x 45 square kernel

Figure 2. A source image and its histogram (A, B) compared to reference images using a 21 x 21 (C, D), 45 x 45 (E,F), or 65 x 65 (G,H) square kernels and (continued) 65 x 65 (I,J) cross kernels



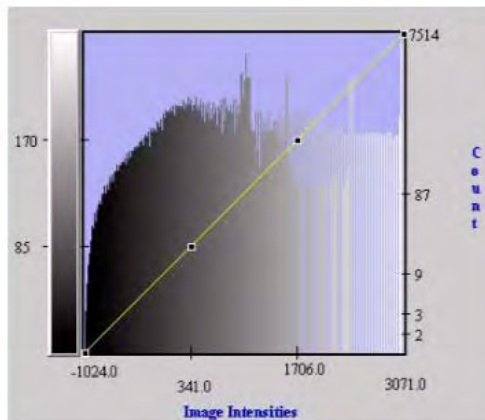
(G) Reference image using a 65 x 65 square kernel



(H) Reference image using a 65 x 65 square kernel



(I) Reference image using a 65 x 65 cross kernel



(J) Histogram for reference image using a 65 x 65 cross kernel

Figure 2. A source image and its histogram (A, B) compared to reference images using a 21 x 21 (C, D), 45 x 45 (E,F), or 65 x 65 (G,H) square kernels and (continued) 65 x 65 (I,J) cross kernels

There are a couple of features to notice about the resultant images. First, the reference images contain a lot of noise. This noise results because the algorithm remapped the areas of the image with smooth features using a histogram used for features with great variation (such as the lung cavity in the example images). Second, notice that the algorithm remaps large areas of the same level of intensity values very high-intensity areas. This feature could be expected, is distracting when viewing the image, and is therefore a good reason to include a thresholding feature to prevent processing on a

pixel when its intensity is too low.

The histogram equalization process can enforce certain characteristics about the image called *clipping*. As the algorithm processes the image, it counts out the number of pixels of a certain intensity. *Clipping* cuts that counting short; it stores the excess pixels of a particular intensity and redistributes that number equally among all intensities. For instance, if a particular intensity must be clipped at 100 and the count is 145, the algorithm redistributes 45 pixels to other brightnesses. It adds to the count of each intensity level an equal number of pixels if there are enough excess, but attempts to spread the excess over as many pixels as possible.

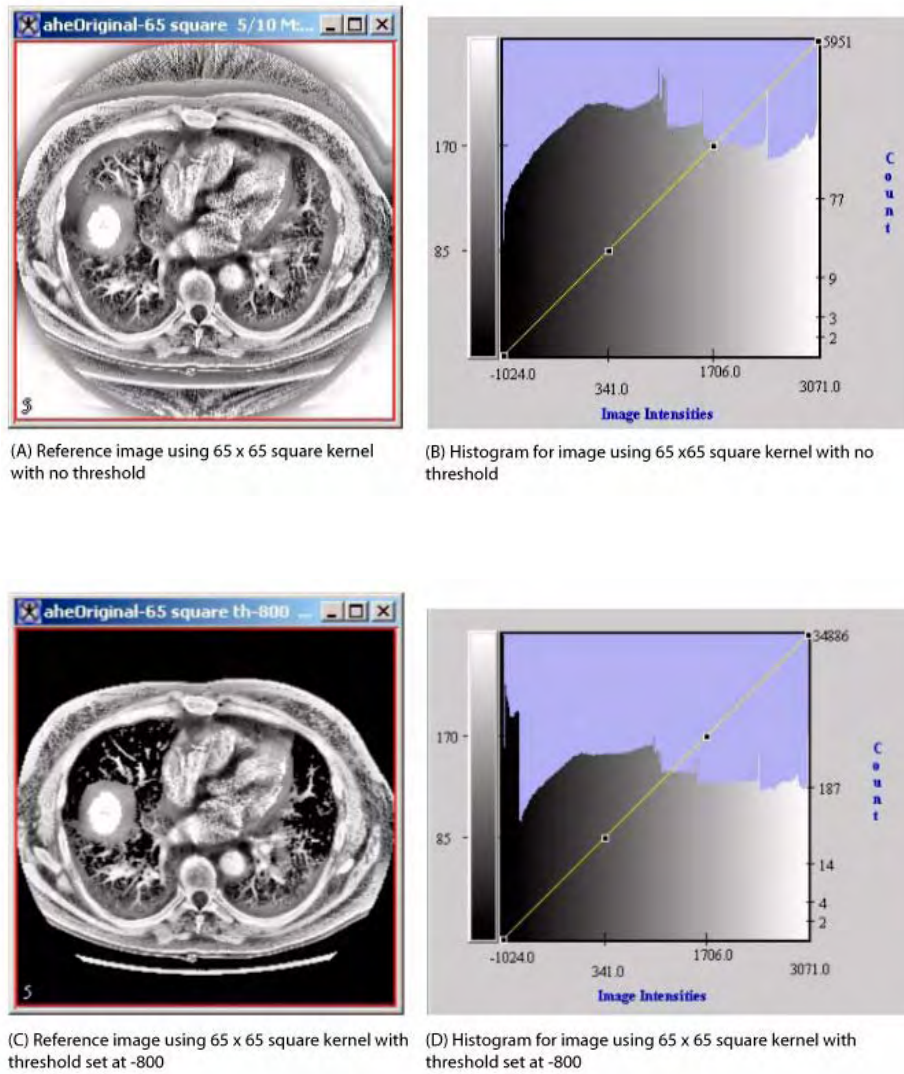


Figure 3. The effect of thresholding on images



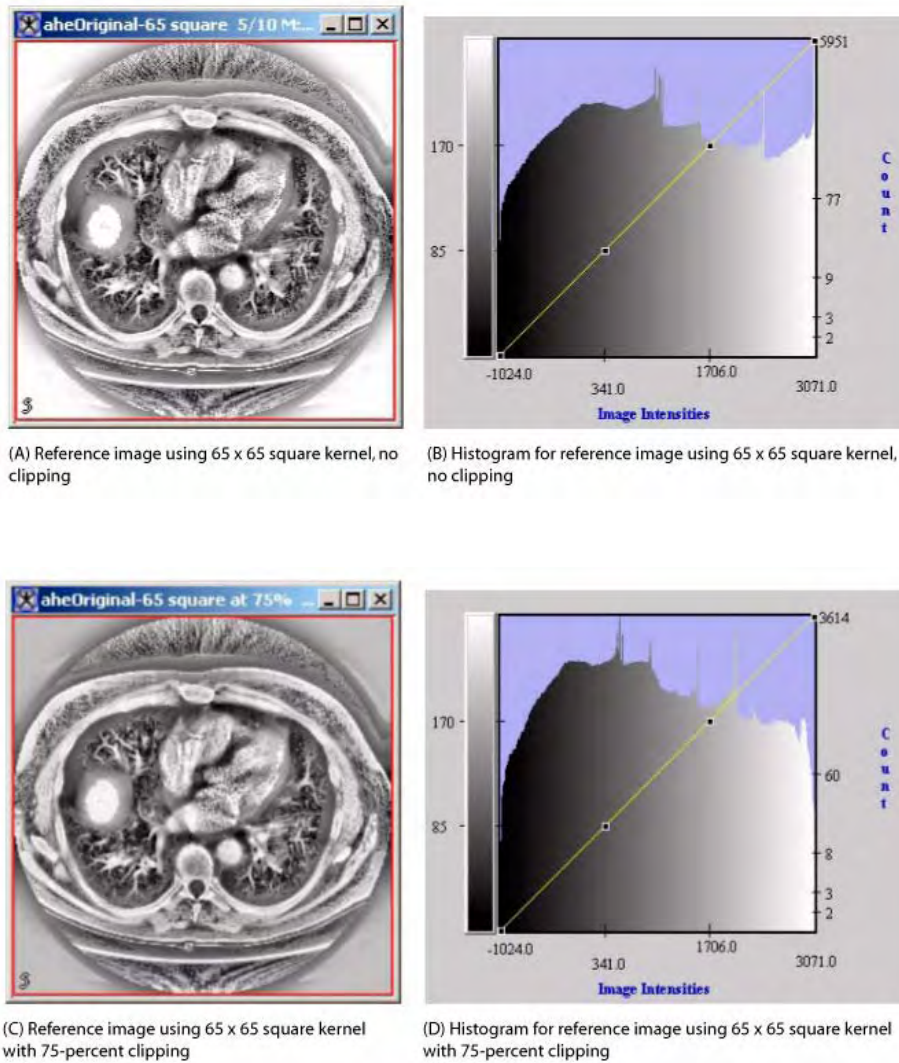


Figure 4. A reference image and histogram with no clipping and with 75-percent clipping

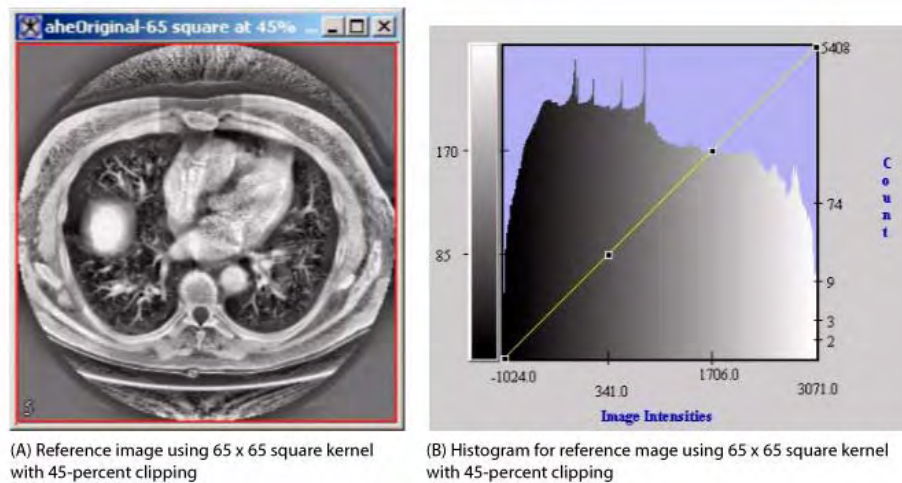


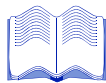
Figure 5. A reference image and histogram with 45-percent clipping

## IMAGE TYPES

You can apply this algorithm to both color and black-and-white images.

## SPECIAL NOTES

This algorithm can evaluate the entire image or only specified volumes of interest (VOIs).



## REFERENCES

Refer to the following references for more information:

Bob Cromwell, "Localized Contrast Enhancement: Histogram Equalization," paper on his web site at <http://www.cromwell-intl.com/3d/histogram/Index.html>.

Douglas A. Lyon, *Image Processing in Java* (Upper Saddle River, New Jersey: Prentice-Hall, 1999).

Tamer Rabie, Rangaraj Rangayan, Raman Paranjape, "Adaptive-Neighborhood Image Deblurring," The University of Calgary, Department of Electrical and Computer Engineering (Calgary, Alberta, Canada: NEC Research Institute, 2002).

John Russ, *The Image Processing Handbook* (Boca Raton, Florida: CRC Press LLC, 2003).

J. Alex Stark, "Adaptive Image Contrast Enhancement Using Generalizations of Histogram Equalization," in *IEEE Transactions on Image Processing* (May 2000).

Claes Tidestav, "Short Introduction to Adaptive Equalization." (Uppsala, Sweden: Uppsala University, April 1996).

## Applying the Histogram Equalization: Neighborhood Adaptive algorithm

To run this algorithm, complete the following steps:

- 1** Open an image.
- 2** Perform, as an option, any other image processing on the image.
- 3** Select Algorithms > Histogram Equalization > Neighborhood Adaptive. The Local Adaptive Histogram Equalization dialog box (Figure 6) opens.
- 4** Complete the information in the dialog box.
- 5** Click OK. The algorithm begins to run.

A pop-up window appears with the status. A series of status messages appear in the window.

When the algorithm finishes running, the pop-up window closes.

Depending on whether you selected New image or Replace image, the results appear in a new window or replace the image to which the algorithms was applied.

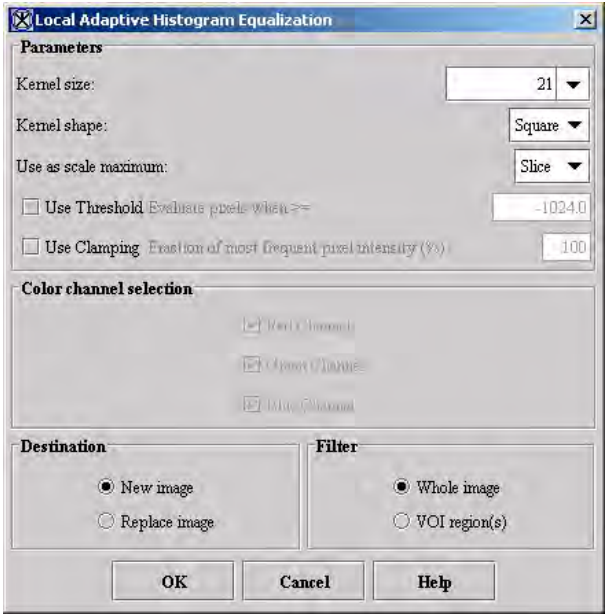
<p><b>Kernel size</b></p>	<p>Indicates the number of pixels horizontally or vertically to be used in finding the histogram with the pixel in question at the center. Select 31, 45, 65, or 129, or type an odd number.</p>	
<p><b>Kernel shape</b></p>	<p>Specifies the shape of the kernel the algorithm should use. Select either:</p> <ul style="list-style-type: none"> <li>• Square—To evaluate all pixels on any column or row away from the central pixel to the perimeters of the kernel</li> <li>• Cross—To choose from only pixels in the same column or row to the perimeter of the kernel as neighbors of the pixel.</li> </ul>	
<p><b>Use as scale maximum</b></p>	<p>Specifies where to find the maximum for the resulting histogram. Select Local, Slice (default), or Image. Note: This value is predefined for some image formats.</p>	
<p><b>Use threshold</b></p>	<p>Limits the algorithm to evaluate pixels only when above the specified value. By default, this field is clear and the algorithm therefore evaluates all pixels.</p> <p>Any given input, however, prevents some pixels from being evaluated. Although using this option potentially causes some valuable pixels to be lost, it reduces the noise associated with evaluating the image edges, which is usually only an image of the imaging device. The default is image minimum value.</p>	
<p><b>Use clamping</b></p>	<p>Indicates the percentage (1 through 99) of the maximum number of pixels per intensity allowed. The algorithm first calculates the number of pixels per intensity and then applies the clamping percentage to yield a maximum number of pixels per intensity allowed. The algorithm records the excess pixels for any intensity that has more pixels than the calculated value and redistributes them equally among all intensities. By default, this field is clear; when selected, 75 is the default percentage.</p>	
<p><b>Red channel</b></p>	<p>Treats all of the pixels in the red channel as its own intensity as if the image were three separate monochrome images (only applicable to color images). You can opt to act on any combination of red, blue, or green color channels.</p>	

Figure 6. Local Adaptive Histogram Equalization dialog box

<b>Green channel</b>	Treats all of the pixels in the green channel as its own intensity as if the image were three separate monochrome images (only applicable to color images). You can opt to act on any combination of red, blue, or green color channels.
<b>Blue channel</b>	Treats all of the pixels in the blue channel as its own intensity as if the image were three separate monochrome images (only applicable to color images). You can opt to act on any combination of red, blue, or green color channels.
<b>New image</b>	Shows the results of the algorithm in a new image window.
<b>Replace image</b>	Replaces the source image with the results of the algorithm.
<b>Whole image</b>	Applies the algorithm on the entire image.
<b>VOI regions</b>	Applies the algorithm to only the selected VOI.
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

**Figure 6. Local Adaptive Histogram Equalization dialog box (continued)**

## Histogram Matching

The Histogram Matching (also called Histogram Specification) algorithm generates an output image based upon a specified histogram.

### Background

The process of Histogram Matching takes in an input image and produces an output image that is based upon a specified histogram. The required parameters for this algorithm are the input image and the specified image, from which the specified histogram can be obtained.

The algorithm works as follows:

The histograms for the input image  $p(f)$  and the specified image  $p_d(g)$  are generated. Here, the desired histogram of the specified image is denoted as  $p_d(g)$ . Cumulative histograms –  $P(f)$  and  $P_d(g)$  are then generated using the following equations:

EQUATION 1

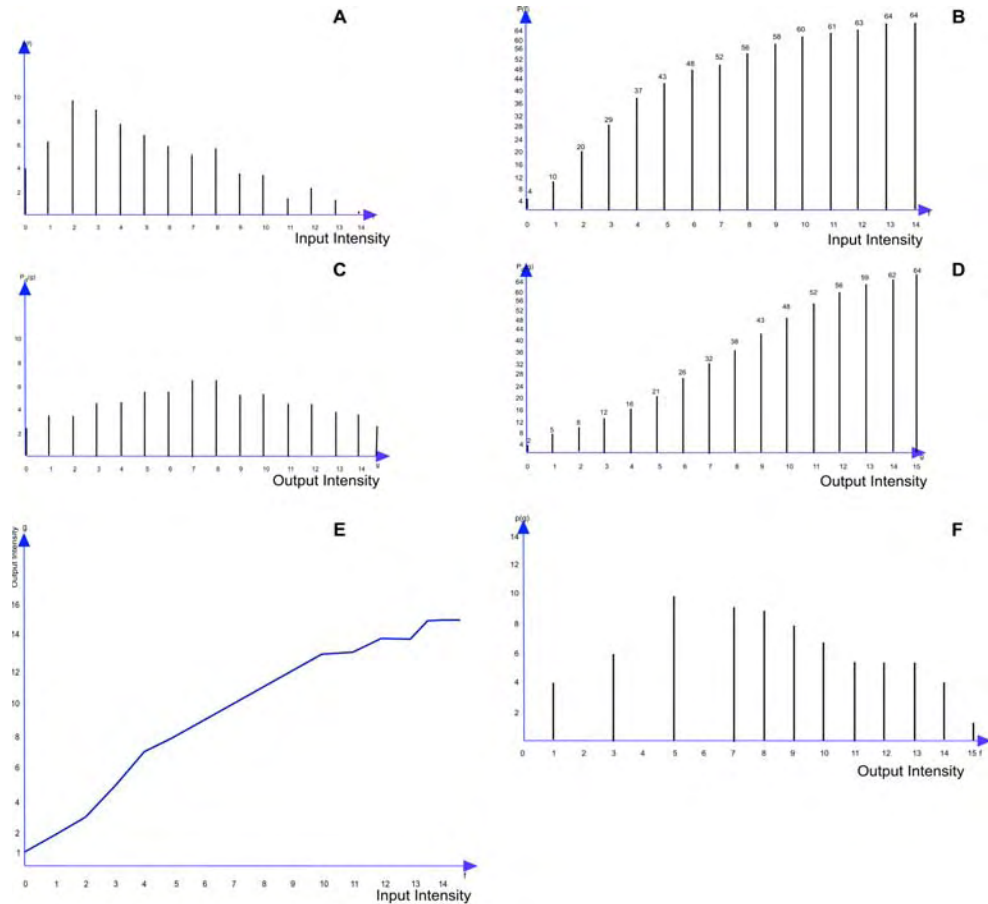
$$P(f) = \sum_{k=0}^f p(k)$$

EQUATION 2

$$P_d(g) = \sum_{k=0}^g p_d(k)$$

where  $f$  and  $g$  are the pixel intensity levels.

The transformation function is obtained by choosing  $g$  for each  $f$ , such that if  $g > f$ , then  $P_d(g) > P(f)$ . The transformation function is then applied to the input image to produce an output image by remapping the pixel intensities. The objective is to find a transformed digital picture of a given picture, such that the sum of absolute errors between the intensity level histogram of the transformed picture and that of a reference picture is minimized. Refer to Figure 1.



**Figure 1. Histograms and cumulative histograms:**  
**(A) histogram of 8x8 pixel image;**  
**(B) cumulative histogram derived from (A);**  
**(C) desired histogram of the same 8x8 pixel image;**  
**(D) cumulative histogram derived from (C);**  
**(E) gray scale transformation function that approximately transforms the histogram (A) to desired histogram (C);**  
**(F) histogram of gray-scale-transformed image obtained by applying the transformation function in (E) to the image with the histogram shown in (A).**

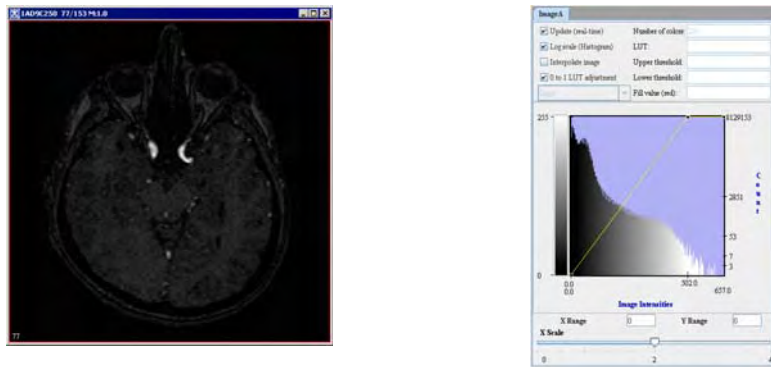


Figure 2. The input image and its histogram  $p(f)$

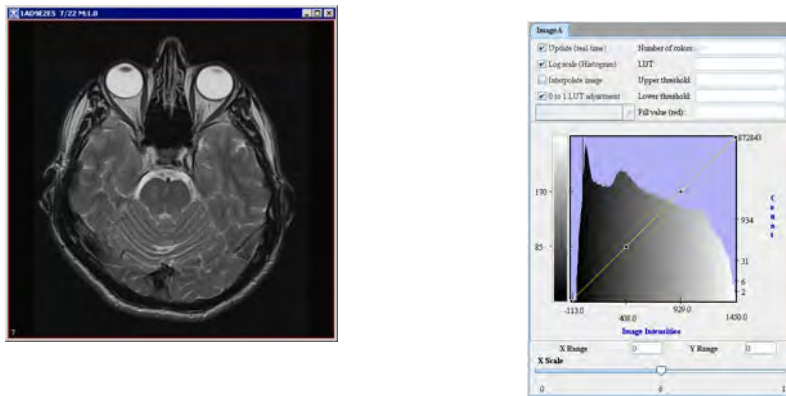


Figure 3. The specified image and its histogram  $pd(g)$

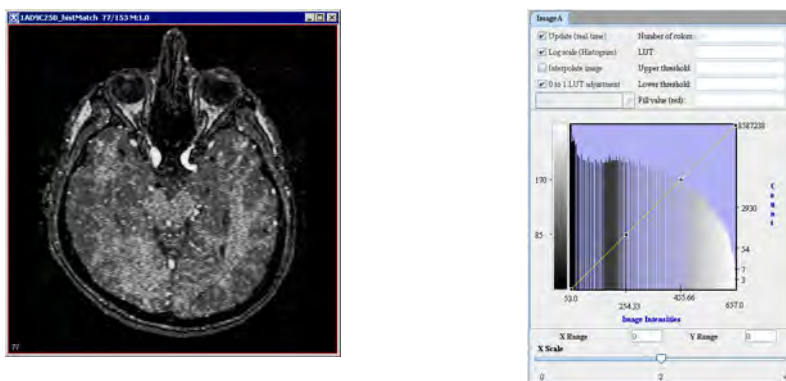


Figure 4. The result image and its histogram



## REFERENCES

Rafael C. Gonzalez and Richard E. Woods. Digital Image Processing. Addison-Wesley Publishing Company, 1992, pp. 173-182

Jae S. Lim. Two-Dimensional Signal and Image Processing. Prentice-Hall, Inc., 1990, pp. 453-459


## IMAGE TYPES

You can apply this algorithm to both color and black-and-white images, as well as 2D and 3D images.

## Applying the Histogram Matching algorithm

To use this algorithm, do the following:

- 1** Select Algorithms > Histogram Tools > Histogram Matching in the main MIPAV window. The Histogram Matching dialog box (Figure 5) appears.
- 2** Complete the information in the dialog box.
- 3** Click OK. The algorithm begins to run, and a progress bar appears momentarily with the status. When the algorithm finishes running, the progress bar disappears, and the results appear in a separate image window.

<b>Histogram match &lt;File name&gt; to</b>	Select the specified image $p_d(g)$ from the list box	
<b>New image</b>	Shows the results of the algorithm in a new image window	
<b>Replace image</b>	Shows the results of the algorithm in the same image window	
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.	

---

<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes this dialog box.
<b>Help</b>	Displays online help for this dialog box.

---

**Figure 5. Histogram Matching dialog box**

## Histogram summary



Sometimes it is more useful to display information in graphical form. At other times, a tabular form is easier to use. In MIPAV, you can view the frequency, or intensity level, distribution in an image or VOI region in either form. To view the information as a graphic representation, you would use either the LUT icon on the image toolbar or the LUT menu in the MIPAV window (refer to “Improving contrast by generating and modifying histograms” in Chapter 1, “Getting Started Quickly with MIPAV”). However, to view the information in a tabular form, you use the Histogram Summary command on the Utilities menu. You can also save the histogram summary information in a TXT (.txt) file.

## Applying Histogram Summary to the image

### To create a histogram summary on an entire image

- 1 Open an image.
- 2 Select Utilities > Histogram Summary in the MIPAV window. Frequency distribution information appears on the Data page in the Output window (Figure 1).

### The histogram summary table includes the following four columns, see Figure 1:

- **Image name**—Name of the image
- **Intensity value**—Value of the pixel
- **Count**—Number of times that pixels of this intensity value occur in the image
- **Square area or cubic volume**—For 2D images, area = count × x resolution × y resolution. For 3D images, volume = count × x resolution × y resolution × z resolution.

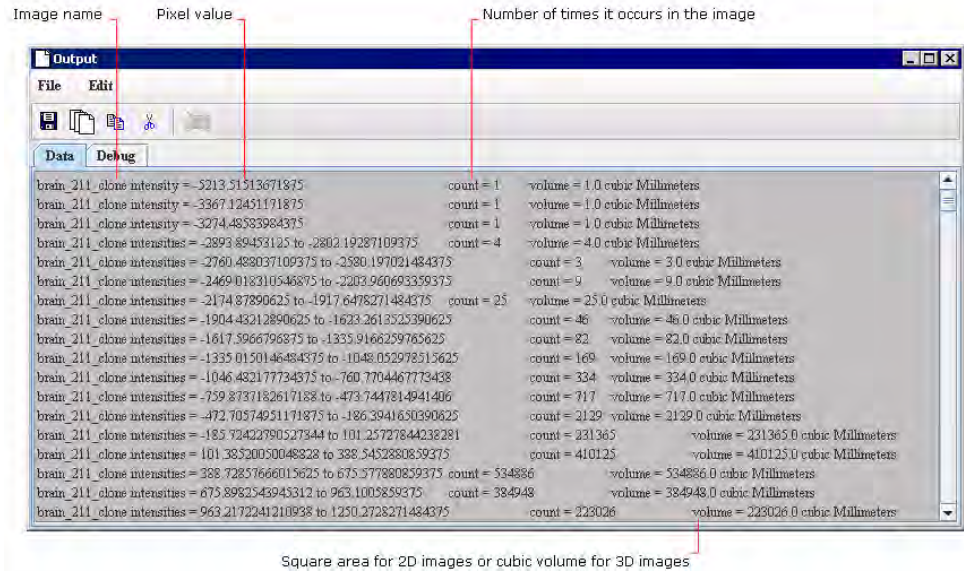


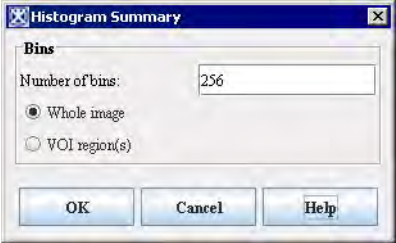
Figure 1. Histogram summary in the Output window

### To create a histogram summary on a VOI

- 1 Open an image.
- 2 Delineate a VOI on the image.
- 3 Select the VOI.
- 4 Select Utilities > Histogram Summary in the MIPAV window.  
Frequency distribution information appears on the Global Data page and on the Data page in the Output window.

### To save the histogram summary

- 1 Select File > Save Messages in the Output window. The Save dialog box opens.
- 2 Type a name and extension in File Name. Make sure that you use the \*.txt extension.
- 3 Click Save. MIPAV saves the histogram summary under the specified name.

<b>Number of bins</b>	Specify the number of bins here. The default value is 256.	
<b>Whole image</b>	Check this option if you want create histogram for the whole image.	
<b>VOI regions</b>	Check this option if you want create histogram only for selected VOI regions.	
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.	
<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes this dialog box.	
<b>Help</b>	Displays online help for this dialog box.	

**Figure 2. The Histogram summary dialog box options**

---

## Manual 2D Series

The Manual 2D Series algorithm allows you to manually or semi manually register two 2D grayscale or color images by placing corresponding points on both images, and then applying either the Landmark—Thin Plate Spline or Landmark—Least Squares algorithm. After applying one of these algorithms, you can then, if needed, use the other algorithm. In addition, at any time, you can manually adjust the alignment of each image.

The algorithm also lets you manually register different slices of a 2.5D and 3D image datasets in the same way as 2D images. See also “Applying the Manual 2D Series algorithm to 2.5D images” on page 391 and “Applying the Manual 2D Series algorithm to 3D images” on page 393.

### Background

Through the Registration: Manual 2D Series dialog (Figure 1), this algorithm provides the tools for manually registering images, as well as the ability to use two of the landmark registration algorithms—Thin Plate Spline and Least Squares.

The following tools are available for manually registering images:

- “Changing displaying proportion” on page 377
- “Moving the adjusted image” on page 380
- “Rotating the adjusted image” on page 381
- “Delineating the landmark points” on page 385
- “Applying the image registration algorithms” on page 386

---

### IMAGE TYPES

You can apply this algorithm to 2D, 2.5 D, and 3D grayscale and color images. The dimensions or image type of the adjusted image need not be the same as the dimensions or image type of the reference image. The adjusted image has the same image type and the same extents as the reference image. See also: “Applying the Manual 2D Series algorithm on 2D images” , “Apply-

ing the Manual 2D Series algorithm to 2.5D images”, and “Applying the Manual 2D Series algorithm to 3D images” .

---

## SPECIAL NOTES

For 2D, 2.5D and 3D images, three or more landmark points are required. The algorithm may fail if nearly all of the points fall on the same line.

---

## REFERENCES

“Least-Squares Fitting of Two 3-D Point Sets” by K.S. Arun, T.S. Huang, and S.D. Blostein, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-9, No. 5, September, 1987, pp. 698-700.

David Eberly, “Thin Plate Splines” of Magic Software at <http://www.magic-software.com>. Also at the same site are the files: MgcInterp2DThinPlateSpline.h, MgcInterp2DThinPlateSpline.cpp, MgcInterp3DThinPlateSpline.h, MgcInterp3DThinPlateSpline.cpp Paper that explains warping from one  $x,y$  source to another  $x,y$  target set.

Fred L. Bookstein, “Principal Warps: Thin-Plate Splines and the Decompositions of Deformations,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 6, June 1989, pp. 567–585.

## Applying the Manual 2D Series algorithm on 2D images

To run this algorithm, complete the following steps:

- 1 Open two images— the reference image and the image you want to register or *adjusted image*.
- 2 Select the reference image.
- 3 Select Algorithms > Registration > Manual 2D Series. The Load dialog box appears. See Figure 1.
- 4 Select the image that you want to register to the reference image. **Note that**, if only two images are open, the Set as Image B box automatically selects the other open image—the one you did not select—as Image B or *adjusted image*. See Figure 1.



**Figure 1. The Load dialog box**

- 5** To select a different image, select the File option, and then click Browse to find and open another image.
- 6** Click OK.
- 7** The Manual 2D Series window opens. By default, the window opens in Blended mode. Note that, opening this window automatically corrects image resolution and scaling.

As shown in Figure 2, the Registration: Manual 2D Series window has the main menu (see below) and two tabs: Blended and Dual. Refer to “Blended tab” on page 376 and “Dual tab” on page 382.

### Options available via the dialog menu

<b>File</b>	<b>Close Registration</b>	Closes this dialog without saving the images.
	<b>Show VOIs in blended window</b>	On the Blended tab, displays the VOIs that you’ve delineated on the Dual tab.
	<b>Thin plate spline for intensity</b>	Uses the Thin Plate Spline algorithm for calculating intensity values. See also “Registration: Landmark—TPSpline” .
<b>Help</b>		Runs help for this dialog box.

## Blended tab

The Blended tab shows both images: the adjusted image superimposed on top of the reference image. By default, the image frame shows both images blended in proportion 50/50.



## CHANGING DISPLAYING PROPORTION

You can use the AlphaBlending image slider to change the displaying proportion towards either increasing the amount of the reference image (the Ref.R side of the slider) or adjusted image (the Adj.A side of the slider) displayed. The image slider is located at the top of the Blended tab. See also Figure 2 and Section: “Moving and rotating the adjusted image” on page 379.

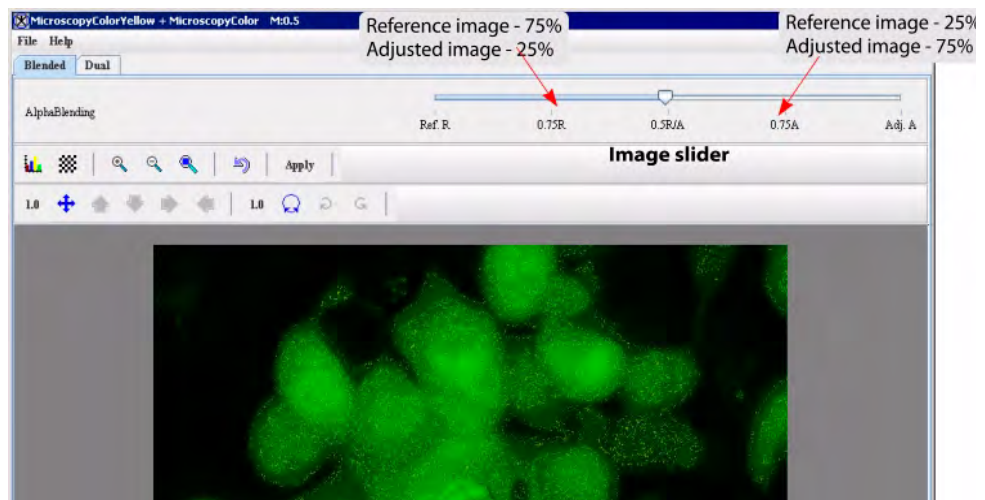


Figure 2. The Blended tab and the image slider

## THE BLENDED TAB OPTIONS:



**Display  
Lookup table**

Opens the Lookup table dialog box. For the Lookup table (LUT) dialog box options, refer to MIPAV User's Guide, Volume 1, Basics, Figure 120 "Look-up Table window". See also "Using LUT to arrange the images" on page 389.

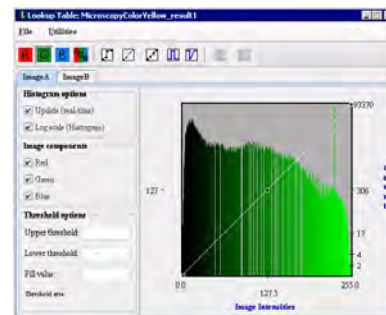


Figure 3. The Blended tab options


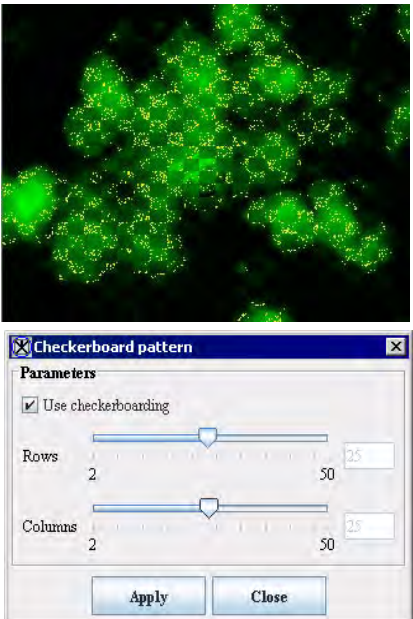



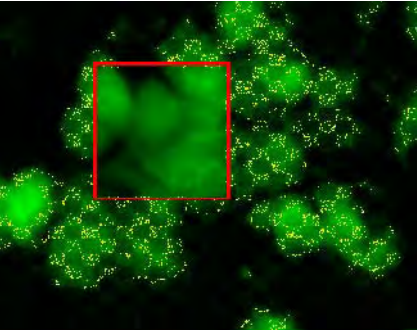

 <p><b>Checker Board</b></p>	<p>The checkerboard tool displays both portions of both images in an alternating fashion, much like the squares on a checkerboard. Portions of the reference image (Image A) appear where the light-colored squares would appear on the checkerboard; portions of the adjusted image (Image B) appear in place of the dark-colored squares.</p> <p>You can use the Checkerboard pattern dialog box options to adjust the number of rows and columns displayed.</p> <p>See also MIPAV User's Guide, Volume 1, Basics, Figure 116 for more information about the checkerboard tool.</p>	
 <p><b>Magnify image 2.0x</b></p>	<p>Magnifies the images <math>2 \times n</math> times.</p>	
 <p><b>Magnify image 0.5x</b></p>	<p>Magnifies the images <math>0.5 \times n</math> times; or minifies the images <math>2 \times n</math> times, which is the same.</p>	
 <p><b>Window region of Image B</b></p>	<p>Displays the window filled only with the adjusted image (Image B) over the reference image. You can use this window to look at the alignment of details in the images.</p>	
 <p><b>Reset image to its original state</b></p>	<p>Restores image to its original state. This icon also deletes all landmark points on the reference image and on the adjusted image. If the Apply button has already been selected, the Reset icon undoes all changes that occurred since the last time the Apply button was selected.</p>	

Figure 3. The Blended tab options (continued)



<p><b>Apply</b></p>	<p>Commits the changes that you made to the images, but it leaves the window open for you to make further changes, if desired.</p>
<p> <b>Translate image</b></p>	<p>Activates <b>moving mode</b> by making the moving icons available. The moving icons or the icons that display direction arrows let you move the adjusted image in one direction only by a specified number of pixels. The default number of pixels is 1; however, the number of pixels may be from 0.01 to 2048.0. Therefore, the direction arrows provide the greatest specificity in moving the image, which is perfect for fine adjustments.</p> <p>To change the number of pixels, click the Set pixel increment icon; enter a desired number in the Pixel Increment box; and then click Apply. Refer to “Moving and rotating the adjusted image” on page 379.</p>
<p> <b>Rotate Image</b></p>	<p>Activates <b>rotation mode</b>. If the rotation of the adjusted image is incorrect, you can change the rotation of the image with the two rotation icons: Rotate clockwise and Rotate counterclockwise. Rotation icons allow you to turn the adjusted image in the desired direction only by a specified degree (1–360). The default degree is 1, and it is shown next to the Rotate Image icon. To change the degree, click the Change Degree Increment icon; enter a desired number in the Degree Increment box; and then click Apply. Refer to “Moving and rotating the adjusted image” on page 379</p>

Figure 3. The Blended tab options (continued)

## MOVING AND ROTATING THE ADJUSTED IMAGE

Figure 4 shows the icons which allow you to move the adjusted image in all directions or specifically up, down, left, and right a certain number of pixels. You can also rotate the adjusted image clockwise or counterclockwise. Note that all these tools are available only on the Blended tab.



**To activate moving mode**, click on the Translate image icon first, and then use the arrow icons to move the adjusted image in desired direction. You can also set up the moving increment (in pixels). Current increment is shown next to the Translate image icon. Refer to Figure 4.



**To activate rotation mode**, click on the Rotate Image icon, and then use the Rotate Clockwise or Rotate Counterclockwise icons to rotate the image. You can also set up the rotation increment (in grades). Current increment is shown next to the Rotate Image icon.

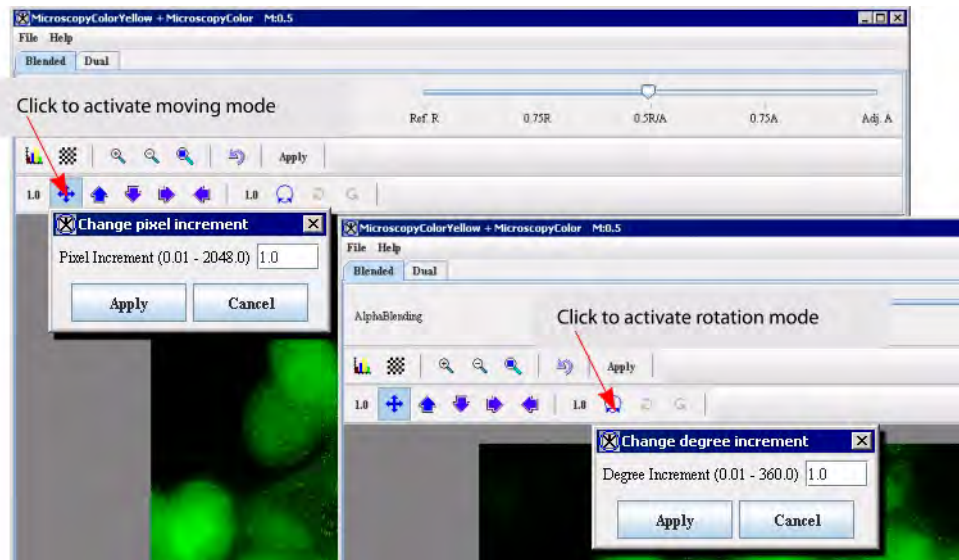
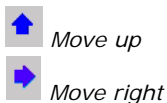


Figure 4. The Blended tab: activating moving and rotation modes

## MOVING THE ADJUSTED IMAGE

The Translate image icon activates moving mode for the adjusted image and allows you to move the image in any direction. This icon only appears on the Blended tab, which is where all manual adjustments to image registration occurs. To move the adjusted image, click Translate Image and, holding down the left mouse button, drag the icon in the direction in which you wish to move the image, and then release the mouse button. After a few moments, the image appears in the new position. Note that, depending on where you moved the adjusted image, its outer portions may disappear beyond the outer edges of the reference image. In this case, you can use the same icon to move the adjusted image again.



## MOVING THE ADJUSTED IMAGE WITH THE DIRECTION ARROWS

The icons that display direction arrows let you move the adjusted image in one direction only by a specified number of pixels. The default number of

pixels is 1; however, the number of pixels may be from 0.01 to 2048.0. Therefore, the direction arrows provide the greatest specificity in moving the image, which is perfect for fine adjustments.

To change the number of pixels, click the Set pixel increment icon; enter a number in the Pixel Increment box; and then click Apply. See Figure 4.

---

## ROTATING THE ADJUSTED IMAGE



*Rotate Image*

If the rotation of the adjusted image is incorrect, you can change the rotation of the image using the rotation icons as shown in as shown in Figure 5.

- 1** To activate rotation mode, click the Rotate Image icon.
- 2** Then, use the Set Degree Increment option to set up the desired degree increment. The increment's range is between 0.01 and 360.0 degrees.
- 3** Select the center of rotation for the image. In order to do that, click Rotate Image, and then set the center by clicking with the mouse on the image. The letter *C* appears with a yellow crosshair cursor (+). You can then drag a line from the center of rotation to move the image above or below the center of rotation.
- 4** To rotate the image clockwise, click Rotate Clockwise; and to rotate the image counterclockwise, click Rotate Counterclockwise.

The image can also be rotated manually by dragging away from the center of rotation, and then dragging either clockwise or counterclockwise.

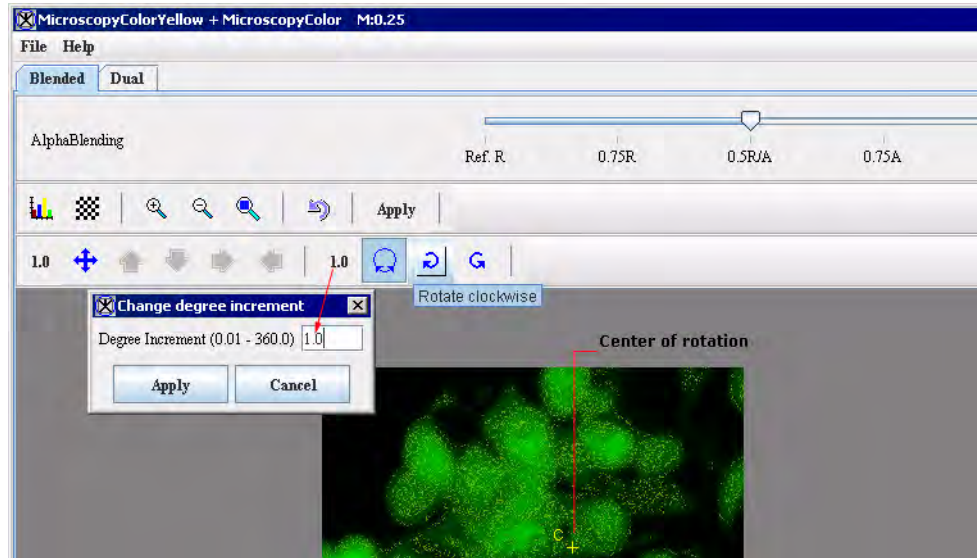


Figure 5. Rotating the adjusted image.

**Note:** You can change the degree increment any time to achieve exact registration.

## DUAL TAB

The Dual tab shows both the reference image and the adjusted image side by side in the image area at the bottom of the page.

Note that the border around the reference image, the image on the left side of the page, appears in red. The border around the adjusted image, which is on the right side of the page, appears in green. Several of the icons near the top of the page are color coded to the images. That is, the red icons apply to the reference image, and the green icons apply to the adjusted image.

## THE DUAL TAB OFFERS THE FOLLOWING OPTIONS

<p><b>Blended tab</b></p>	<p>Displays the Blended page, which shows the image to be registered superimposed on the reference image. This page allows you to move and align the image to be registered with the reference image. For the Blended tab options refer to Figure 2.</p>
<p><b>Dual tab</b></p>	<p>Displays the Dual page, which shows each image side by side, and allows you to delineate three or more landmarks on each image before you apply the Landmark—Least Squares algorithm or the Landmark—Thin Plate Spline algorithm to the images.</p>
<p> <b>Lookup Table</b></p>	<p>Displays the Lookup Table window, which includes two tabs: Image A, the reference image, and Image B, the image to be registered.</p>
<p> <b>Magnify reference image 2.0x</b></p>	<p>Doubles the size of reference images, magnifying them to twice their current size. If an image is too large for the current window size, scroll bars appear, and you may need to manually adjust the size of the window.</p> <p>To restore reference images to their original size, use the Magnify Reference Image 0.5x icon, as many times as you used this icon.</p>
<p> <b>Magnify reference image 0.5x</b></p>	<p>Reduces the magnification level of reference images in half. To restore reference images to their original size, use the Magnify Reference Image 2.0x icon, as many times as you used this icon.</p>
<p> <b>Magnify adjusted image 2.0x</b></p>	<p>Doubles the size of adjusted images, magnifying them to twice their current size. If an image is too large for the current window size, scroll bars appear, and you may need to manually adjust the size of the window.</p> <p>To restore adjusted images to their original size, use the Magnify Reference Image 0.5x icon, as many times as you used this icon.</p>
<p> <b>Magnify adjusted image 0.5x</b></p>	<p>Reduces the magnification level of adjusted images in half. To restore adjusted images to their original size, use the Magnify Reference Image 2.0x icon as many times as you used this icon.</p>
<p> <b>Apply least squares alignment</b></p>	<p>Applies the Landmark—Least Squares algorithm to the image to be registered. After the algorithm is applied, if needed, you can apply the Landmark—Thin Plate Splines algorithm on the image, or you can manually adjust the alignment of the image on the Blended page.</p>

Figure 6. The Dual tab options






 <p><b>Apply thin plate spline alignment</b></p>	<p>Applies the Landmark—Thin Plate Spline algorithm to the image to be registered. After this algorithm is applied, if needed, you can apply the Landmark—Least Squares algorithm on the image, or you can manually adjust the alignment of the image on the Blended page.</p>
 <p><b>Reset image to original state</b></p>	<p>Restores the image to its original state. This icon deletes all points on the reference image and on the adjusted image. If the Apply button has already been selected, the Reset icon undoes all changes that occurred since the last time the Apply button was selected.</p>
 <p><b>Reference slice markers</b></p>	<p>Adds a landmark point to the reference image each time you click the left mouse button. Note that the points are sequentially labeled from 1 on in the order in which you created them. To remove points, return the cursor to its default mode by clicking <b>Return to default mode</b>, select the point to be deleted, and then click the Delete Selected Reference Slice Markers icon.</p>
 <p><b>Adjusted slice markers</b></p>	<p>Adds a landmark point to the adjusted image each time you click the left mouse button. Note that the points are sequentially labeled from 1 to n in the order in which you created them. To remove points, return the cursor to its default mode by clicking <b>Return to default mode</b>, select the point to be deleted, and click the Delete Selected Adjusted Slice Markers icon. <b>Note:</b> landmarks in the reference image should correspond to landmarks in the adjusted image.</p>
 <p><b>Delete selected reference slice markers</b></p>	<p>Removes the landmark point that you selected in the reference image. To remove a point, first return the cursor to its default mode by clicking <b>Return to default mode</b>, then select the point to be deleted, and then click the Delete Selected Reference Slice Markers icon.</p>
 <p><b>Delete selected adjusted slice markers</b></p>	<p>Removes the landmark point that you selected in the adjusted image. To remove the point, first return the cursor to its default mode by clicking <b>Return to default mode</b>, then select the point to be deleted, and finally click the Delete Selected Adjusted Slice Markers icon.</p>
 <p><b>Copy reference markers to adjusted image</b></p>	<p>Copies landmarks from the reference image to adjusted image. Note that landmarks which were copied to the adjusted image have the same x, y coordinates as the reference image landmarks.</p>
 <p><b>Copy adjusted markers to reference markers</b></p>	<p>Copies landmarks from the adjusted image to reference image. Note that landmarks you copied to the reference image have the same x, y coordinates as the adjusted image landmarks.</p>

Figure 6. The Dual tab options (continued)





 <p><b>Return to default mode</b></p>	<p>Returns the cursor to its default mode, which allows you to change from adding points to deleting points from the image.</p>
 <p><b>Apply</b></p>	<p>Commits the changes that you made to the images, but it leaves the window open for you to make further changes if desired.</p>

Figure 6. The Dual tab options (continued)

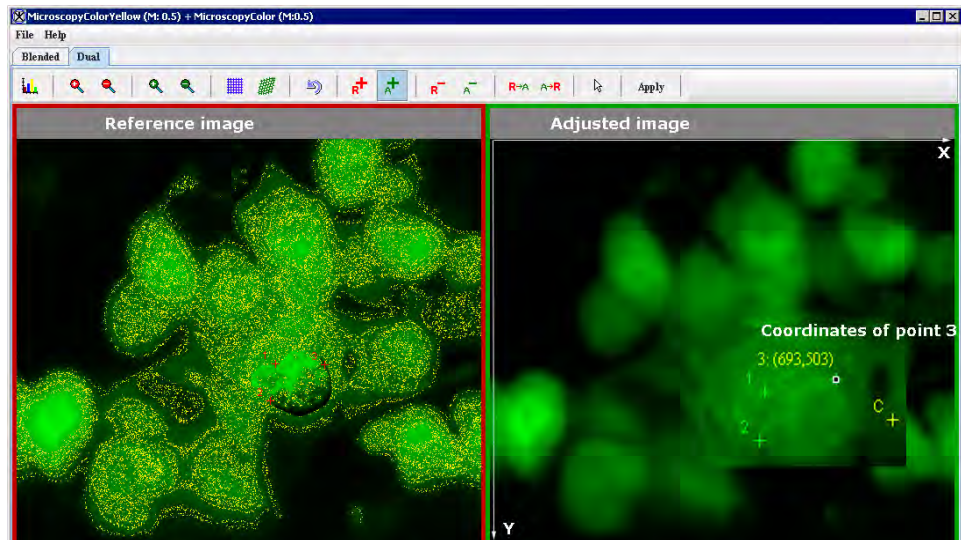


Figure 7. The Dual tab shows both: the reference image and adjusted image side by side. Three corresponding landmark points were delineated in the reference and adjusted image.

## USING THE IMAGE REGISTRATION ALGORITHMS

Before applying the Landmark—Least Squares algorithm or the Landmark—Thin Plate Spline algorithm to the images you must delineate three or more *landmark points* or *landmarks* on each image. See Figure 7.

## DELINEATING THE LANDMARK POINTS

**To place landmarks on the reference image**, click the Reference Slice Markers icon, and then click the mouse on the reference image where the

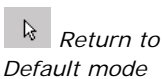
landmark should be located. The landmark appears on the image in red type and is sequentially numbered. Create at least two more landmarks on the image.



**To place landmarks on the adjusted image**, click the Adjusted Slice Markers icon, and create at least three landmarks on the image by clicking the mouse. The landmarks, which are sequentially numbered, appear in green type.

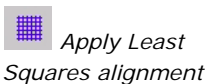


**To remove a landmark point** from the reference or adjusted image, you need to return the mouse cursor to its default mode by clicking on the Return to Default Mode icon. You can then select the point by clicking on it with the mouse, and then delete it by clicking the corresponding Delete Selected Markers icon.

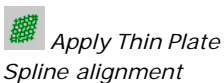



---

## APPLYING THE IMAGE REGISTRATION ALGORITHMS



After you have delineated the landmark points on both images, select the Least Squares algorithm. **The Least Squares algorithm** registers the adjusted image to the reference image by using corresponding points placed in both images.



**The Thin Plate Spline algorithm** allows you to register two images in a nonlinear manner using the corresponding landmark points that you delineate in both images.

The program registers the images and the landmarks appear in yellow type followed by the specific positions in parentheses.

**At this point, you should determine the following:**

- 1** Whether there is a need to make other manual adjustments to the registration. If so, return to the Blended tab and use the direction or the rotation icons. Use the **Window region of image B** icon to verify alignment. Refer to “making manual reconcilments to the adjusted image” on page 387.
- 2** Whether to run the other algorithm. If so, simply call either the Least Squares or Thin Plate Spline algorithm, as appropriate.

- 3 Whether to restore the adjusted image to its original form by clicking Reset.
- 4 Whether to select Apply to keep changes made by the algorithm.

---

## MAKING MANUAL RECONCILEMENTS TO THE ADJUSTED IMAGE

After you apply either the Least Squares or the Thin Plate Spline algorithm on the images, you may want to manually ascertain whether key portions of the reference image and the image to be registered, or adjusted image, are in alignment. One way of checking alignment, or registration, is using the **Window region of image B** icon. A recommended method for checking registration and for correcting it follows.

To check registration, do the following:



*Window region*

- 1 Click Blended in the Manual 2D Series window to display the Blended page. Both the reference image and the image to be registered, or adjusted image, appear at the bottom of the window. The adjusted image is superimposed over the reference image.
- 2 By default, the arrow on the AlphaBlending slider appears in the middle of the slider at 0.5R/A, which means that 50 percent of each image are shown on the page.
- 3 Slide the arrow on the AlphaBlending slider all the way to the right until it reaches the label “Adj. A.” This label indicates that 100 percent of the adjusted image is displayed on the page and, therefore, the reference image does not appear.
- 4 Now, open Lookup table for the adjusted image. Note that the window includes two tabs: one for Image A, the reference image, and one for Image B, the adjusted image.
- 5 Use the LUT options to change the intensity for one of the color channels for the adjusted image, for example, to increase the red channel. The purpose in doing this task is to easily distinguish the adjusted image from the reference image. Refer to “Using LUT to arrange the images” on page 389.

- 6 Return to the Blended tab of Manual 2D Registration dialog box and move the AlphaBlending Image slider arrow all the way to the left on the slider. The label “Ref. R” indicates that the page now displays 100 percent of the reference and does not display the adjusted image.
- 7 Click the Window Region of Image B icon and position the cursor over the reference image. Notice that a square box appears on the image. The box is filled with the adjusted image, which is appears in pale red. You can use this box to look at the alignment of details in the images. See Figure 8.

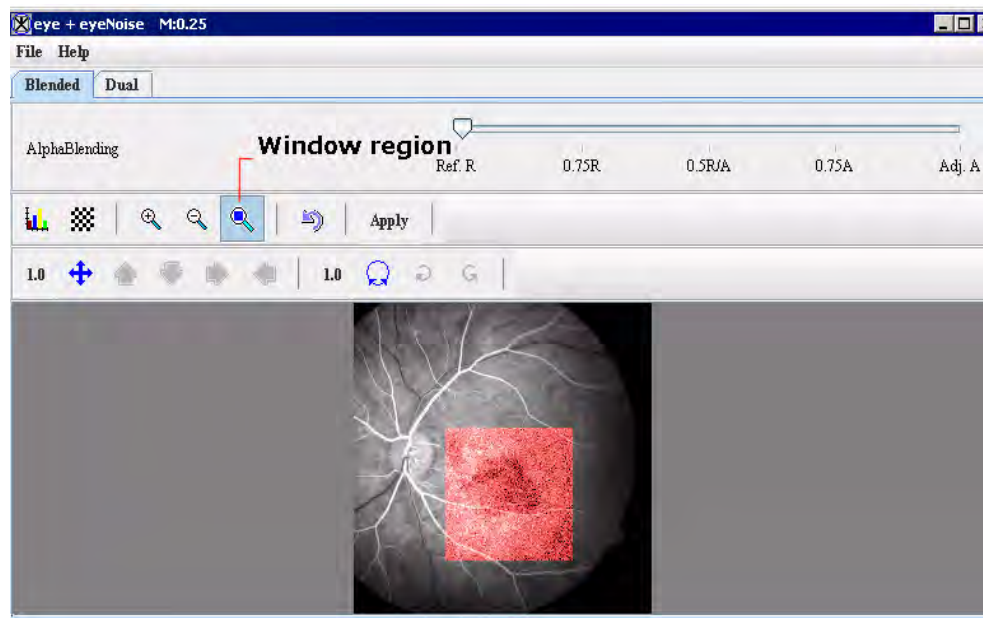


Figure 8. The Window Region box is filled with the adjusted image

- 8 Determine whether you want to correct the alignment of the images. If you do, there are several ways to align the adjusted image:
  - Moving the image using the Translate image icon and direction arrows. See also “Moving the adjusted image” on page 380.
  - Rotating the image using the Rotate Image icon. See also “Rotating the adjusted image” on page 381.

## USING LUT TO ARRANGE THE IMAGES

You can use the LUT options to distinguish the adjusted image from the reference image. In order to do that

- 1 Return to Blended mode.
- 2 Click the Lookup Table icon, to display the Lookup Table window. Note that the window includes two tabs: one for Image A – the reference image, and the other one for Image B – the adjusted image.
- 3 For the adjusted image, drag the image intensity line for the red channel up in the middle. In this case, it creates a lighter colored red. The adjusted image now appears in pale red. Actually, you can change the color of the image to any color. See Figure 9 below.

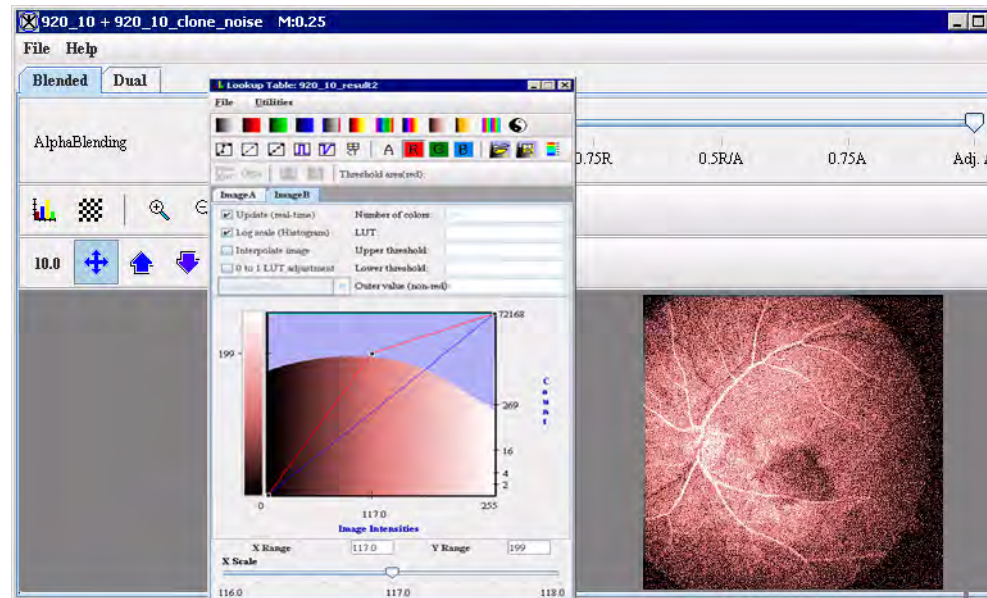
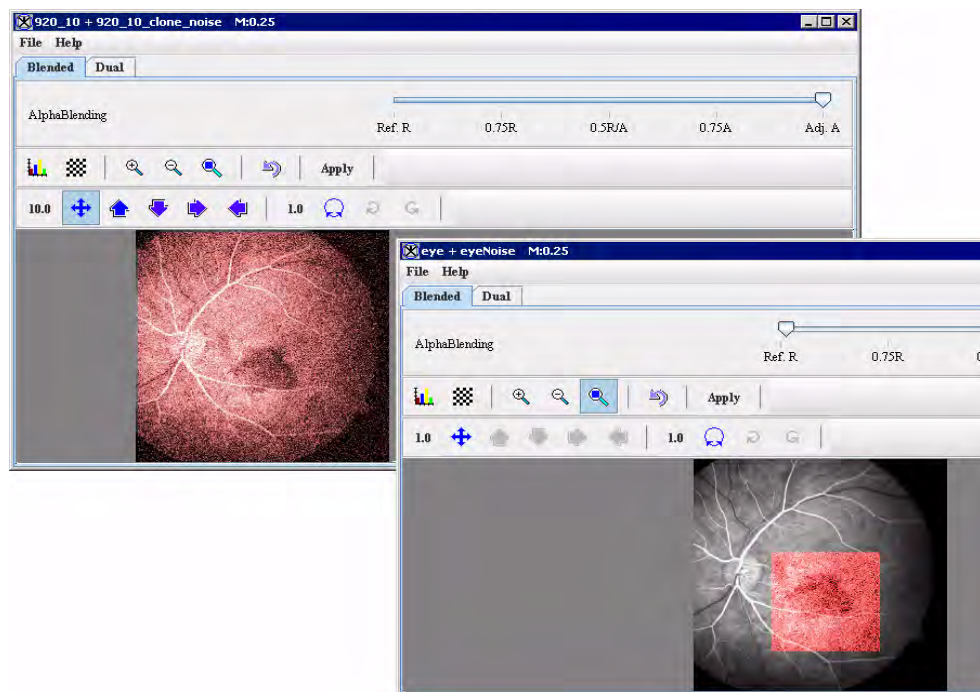


Figure 9. The adjusted image and its Lookup table



**Tip:** You may want to maximize the window and enlarge the images so that you can view details of the images while checking alignment. To maximize the window, simply click Restore. To maximize the image, click the Magnify image 2.0x icon, which magnifies the image to twice its current size, as many times as needed.

- 4** Return to the Blended tab and move the image slider arrow all the way to the left on the slider, so the page now displays only the reference image.
- 5** Click the Window Region of Image B icon and position the cursor over the reference image.
- 6** The Windows Region box appears on the image, filled with the adjusted image, which is appears in pale red.
- 7** Use this box to look at the alignment of details in the images.



**Figure 10.** The adjusted image which appears in the Blended tab and also in the Window Region.

## Applying the Manual 2D Series algorithm to 2.5D images

To run this algorithm on 2.5D images, complete the following steps:

- 1** Open a single 2.5 image dataset. When you apply this algorithm, you select the slices in the dataset that you want to register.
- 2** Perform, as an option, any other image processing, except scaling, on the image dataset.
- 3** Select Algorithms > Registration > Manual 2D Series. The Manual 2D Series window for 2.5D images (Figure 11) opens.
- 4** Because you are registering slices of the image, and not separate images, all of the necessary icons and controls appear in a simple window that does not include tabbed pages. Note that two additional sliders appear at the top of the page:
  - **Reference slice slider** allows you to choose which slice to use as a reference image.
  - **Adjusted slice slider** allows you to choose which slice to use as an image to be registered.



---

**Note:** the boxes to the right of each slider indicate the number of slice which you are currently viewing. All of the remaining icons and controls work in the same way that they do on the Manual 2D Series window for 2D images.

---

- 5** Choose the reference and adjusted slice. You can consider using the LUT to distinguish the adjusted image from the reference image as described in Section “Using LUT to arrange the images” on page 389.
- 6** Use the Alphablending image slider to display only the reference slice and then place at least four landmarks on the reference slice. Repeat the same procedure for the adjusted slice.
- 7** Now, move the image slider arrow all the way to the left on the slider, so the dialog box will display only the reference slice.

- 8 Click the Window Region of Image B icon and position the cursor over the reference image. The Windows Region box appears on the image, filled with the adjusted image, as shown in Figure 11. Use this box to look at alignment of fine detail in the slices.

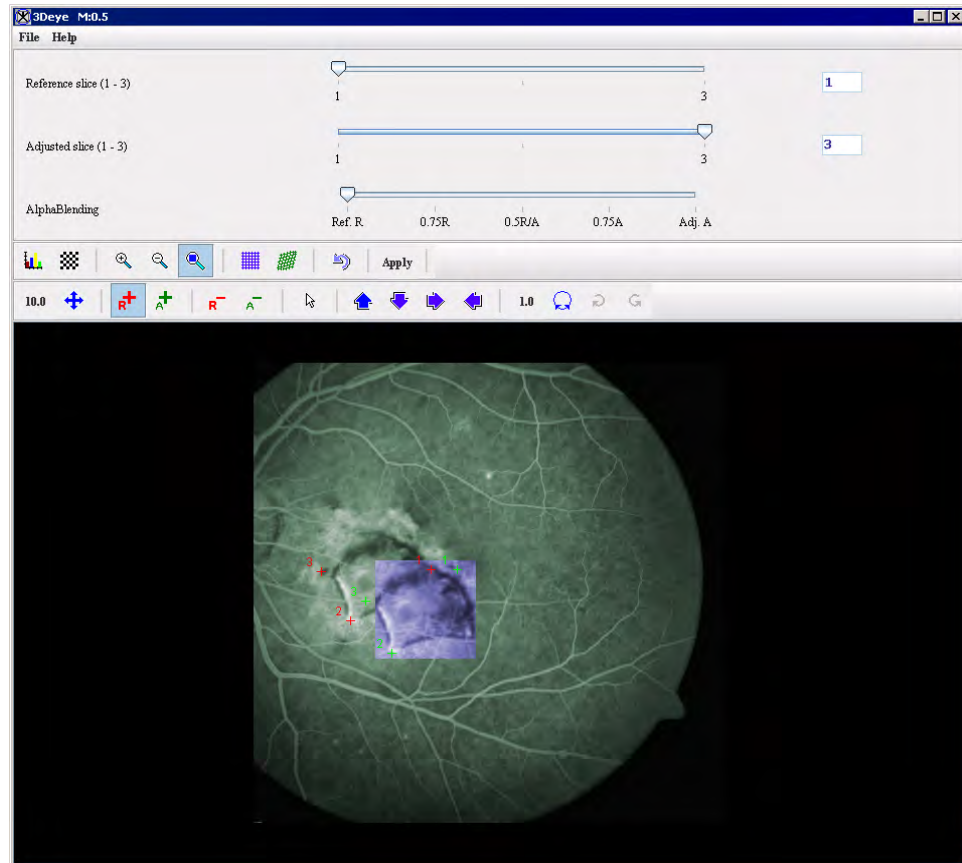


Figure 11. Applying the Manual 2D Series Registration algorithm on a 2.5D image. Slice 3 (adjusted slice) is registered to slice 1 (reference slice). To differentiate the slices, we use LUT options to color the reference slice to pale green and adjusted slice to pale blue.

- 9 At this point, you should determine the following:
  - Whether there is a need to make manual adjustments to the registration. If so, use the direction or the rotation icons to verify alignment.
  - Whether to run one of the registration algorithms. If so, simply call either the Least Squares or Thin Plate Spline algorithm, as appropriate. Select Apply to keep changes made by the algorithm.



## Applying the Manual 2D Series algorithm to 3D images

When you apply this algorithm, you select *the slices*, which you want to register, in a single 3D dataset. To run the algorithm on 3D images, complete the following steps:

- 1** Open a single3 image dataset. Perform, as an option, any other image processing, except scaling, on the image dataset.
- 2** Select Algorithms > Registration > Manual 2D Series. The Manual 2D Series window for 3D images opens.
- 3** Because you are registering *slices of a single 3D image*, all of the necessary icons and controls appear in a window that does not include tabs. Note that there are three sliders appear at the top of the page:
  - **Reference slice slider** allows you to choose which slice to use as a reference image.
  - **Adjusted slice slider** allows you to choose which slice to use as an image to be registered.
  - **AlphaBlending Slider** which, by default, displays both the reference and adjusted slices blended in proportion 50/50.

---

**Note:** the boxes to the right of the Reference Slice and Adjusted Slice slider indicate the number of slice which you are currently viewing. All of the remaining icons and controls work in the same way that they do on the Manual 2D Series window for 2D images.

---

- 4** Choose the reference and adjusted slice. Use the LUT to differentiate the adjusted image from the reference image. See also: Section “Using LUT to arrange the images” on page 389.
- 5** Use the AlphaBlending image slider to display only the reference slice and then place at least four landmarks on the reference slice. Repeat the same procedure for the adjusted slice.
- 6** Move the image slider arrow all the way to the left on the slider, so the window now displays only the reference slice.
- 7** Click the Window Region of Image B icon and position the cursor over the reference image. The Windows Region box appears on the image, filled with the adjusted image, as shown in Figure 12. You can use either this box or Checker Board to look at alignment of details in the slices.

**8** At this point, you should figure out the following:

- Whether there is a need to make further manual adjustments to the registration. If so, use the direction or the rotation icons to verify alignment.
- Whether to run one of the registration algorithms. If so, simply call either the Least Squares or Thin Plate Spline algorithm, as appropriate.

**9** When done, select Apply to keep changes made by the algorithm.

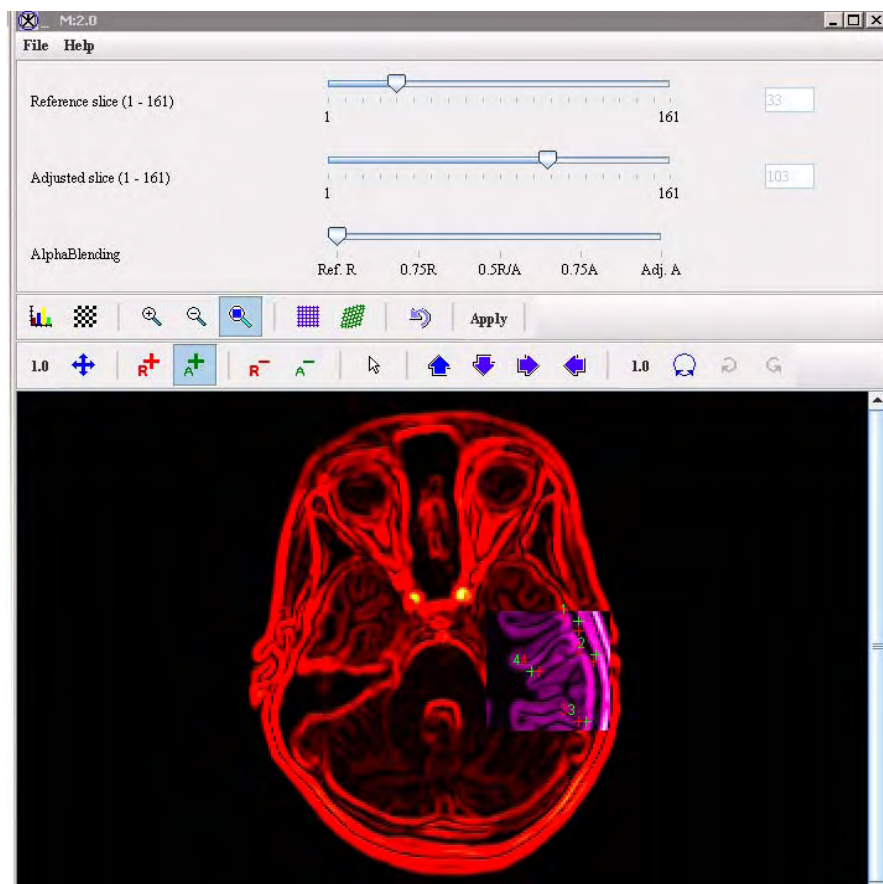


Figure 12. Registering the 3D image.

## Extract Surface (Marching Cubes)

*The Extract Surface algorithm described in this section is based on the Marching Cubes algorithm, which was designed by William E. Lorensen and Harvey E. Cline to extract surface information from a 3D array of values. The applications of this algorithm are mainly concerned with medical visualizations such as CT and MRI scan data images.*

### Background

The basic notion of the algorithm is that we can define a voxel by the pixel values at the eight corners of the cube. We also can define an isosurface which intersects the cube, refer to Figures 1 and 2. If one or more pixels of a voxel have values less than the user-specified isovalue, and one or more have values greater than this value, we know that the voxel must contribute some component of the isosurface.

By determining which edges of the voxel are intersected by the isosurface, we can create triangular patches that divide the voxel between regions *within the isosurface* and regions *outside*. By connecting the patches from all voxel on the isosurface boundary, we get a surface representation.

The indexing convention for vertices and edges used in the algorithm is shown in Figure 1 below.

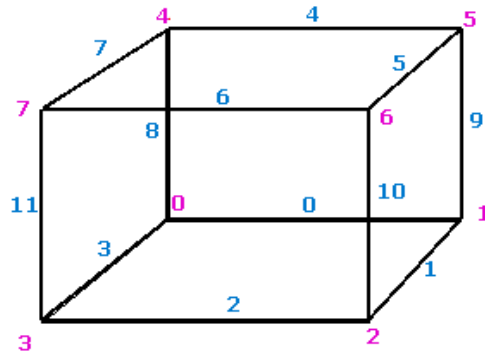
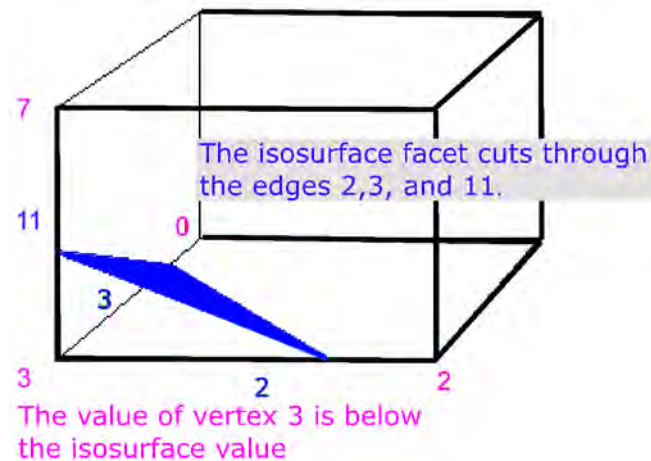


Figure 1. The indexing convention for vertices and edges. Here, edge indexes are shown in blue and vertex indexes are shown in magenta



**Example:** If for example, the value at vertex 3 is below the isosurface value and all the values of all the other vertices are above the isosurface value, then we would create a triangular facet which cuts through edges 2,3, and 11 as shown in Figure 2. The exact position of the vertices of the triangular facet depends on the relationship of the isosurface value to the values at the vertices 3-2, 3-0, 3-7 respectively.



**Figure 2. Vertex 3 is inside the volume**

Therefore, we must specify a *threshold value*, first. For this value, some voxels will be entirely *inside* or *outside* the corresponding isosurface and some voxels will be *intersected by the isosurface*. In the first pass of the algorithm, the *voxels that are intersected by the threshold value* are identified. In the second pass, these voxels are examined and a set of one or more polygons is produced. These polygons are then output for rendering.

Each of the voxel's 8 vertices can be either inside or outside of the isosurface value, thus, there are  $2^8 = 256$  possible ways how an isosurface can intersect the voxel. To simplify the algorithm, we can reduce this complexity by assuming that cell combinations are duplicated under the following conditions:

- Rotating by any degree over any of the 3 primary axis (X,Y, or Z);
- Mirroring the isosurface's shape across any of the 3 primary axis;
- Inverting all corners and flipping the normals of the relating polygons.

By taking this symmetry into account, we can reduce the original 256 combinations to a total of 15 combinations, refer to Figure 3.

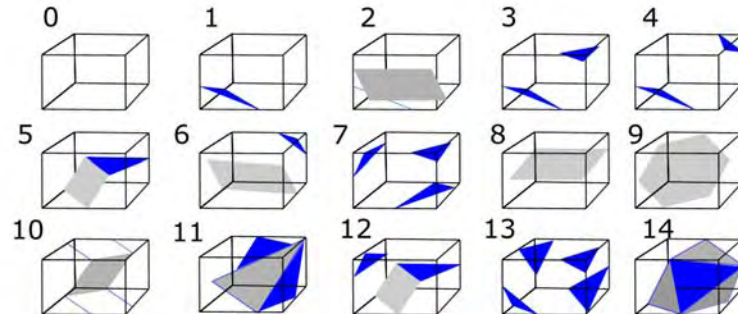


Figure 3. How 256 cases can be reduced to 15 cases

Now we can use a table (called *edgeTable*) which maps the vertices under the isosurface to the intersecting edges. An 8 bit index can be formed from the table where each bit corresponds to a vertex, refer to Table 1.

Table 1. *edgeTable*

```

cubeindex = 0;
if (grid.val[0] < isolevel) cubeindex |= 1;
if (grid.val[1] < isolevel) cubeindex |= 2;
if (grid.val[2] < isolevel) cubeindex |= 4;
if (grid.val[3] < isolevel) cubeindex |= 8;
if (grid.val[4] < isolevel) cubeindex |= 16;
if (grid.val[5] < isolevel) cubeindex |= 32;
if (grid.val[6] < isolevel) cubeindex |= 64;
if (grid.val[7] < isolevel) cubeindex |= 128;

```

Looking up the *edgeTable* returns a 12 bit number, each bit corresponding to an edge. 0 if the edge isn't cut by the isosurface and 1 if the edge is cut by the isosurface. If none of the edges are cut the table returns zero. This occurs when the *cubeindex* is 0, which means that all vertices are below the isosurface, or *0xff* (all vertices above the isosurface).

Using the example shown in Figure 1, where only vertex 3 was below the isosurface, *cubeindex* would equal 0000 1000 or 8. The *edgeTable*{8} equals 1000 0000 1100. This means that edge 2,3, and 11 are intersected by the isosurface, refer to Figure 2.

Now the intersection points can be calculated by linear interpolation. If  $P_1$  and  $P_2$  are the vertices of a cut edge and  $V_1$  and  $V_2$  are the scalar values of each vertex, the intersection point  $P$  is given by the following equation:

$$P = P_1 + (isovalue - V_1)(P_2 - P_1)/(V_2 - V_1)$$

The last part of the algorithm involves forming the correct triangular facets from the positions where the isosurface intersects the edges of the grid cell. In order to do that, another table (also called *triTable*) is used. This table uses the same cubeindex from edgeTable, but allows the vertex sequence to be searched for as many triangular facets as necessary to represent the isosurface within the grid cell. Each of the non-trivial configurations results in between 1 and 4 triangles is added to the isosurface. The actual vertices themselves can be computed either by interpolating along the edges, or, by default their location to the middle of the edge.

Since we can create surface patches for a single voxel, we can apply this algorithm to the entire volume. We can either treat each cube independently, or we can propagate edge intersections between the cubes which share the edges. The sharing of edge/vertex information also results in a more compact model.

---

## REFERENCES

The following sources were used:

Lorenson, W.E. and Cline, H.E., *Marching Cubes: a high resolution 3D surface reconstruction algorithm*, Computer Graphics, Vol. 21, No. 4, pp 163-169 (Proc. of SIGGRAPH), 1987.

Paul Bourke, *Polygonising a scalar field Also known as: "3D Contouring," "Marching Cubes," "Surface Reconstruction;"* May 1994, <http://local.wasp.uwa.edu.au/~pbourke/modelling/polygonise/>

## IMAGE TYPES

You can apply the Extract Surface algorithm to any type of images using in MIPAV.

## Output Files

The output of the algorithm is a set of vertices and normals as well as the map that shows how the vertices are connected. This information can be saved in following formats:

- Tab-delimited (\*.txt);
- Surface (\*.sur), which is MIPAV format. A surface file contains the above information plus the additional information that describes the surface color and more. This is defined by an XML scheme, which can be found on the MIPAV web site (<http://mipav.cit.nih.gov/documentation/xml-format/image/>);
- VRML (\*.wrl), which can then be processed by any application that can read and display VRML objects.

## Applying the Extract Surface Algorithm

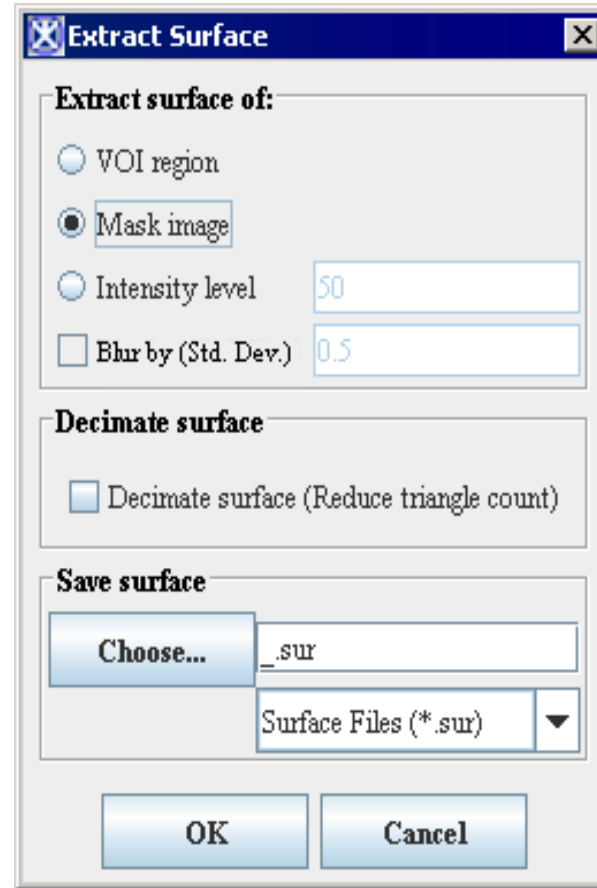
To run the algorithm, complete the following steps:

- 1** Open an image of interest;
- 2** Select Algorithms > Extract Surface (marching cubes);
- 3** The Extract Surface dialog box opens. Here, you can choose to apply following three options:
  - You may select a VOI region, and then apply the algorithm to the selected VOI;
  - You may apply a mask to the image first, and then run the algorithm;
  - Or you may extract a surface from the region with a chosen intensity level.
- 4** Select a type and location of the output file. You can save the result as a \*.txt file (in Tab-delimited format), \*.wrl file, or surface file (\*.sur). Refer to Section “Output Files” for more information.

**5 Press OK.**

The algorithm begins to run, and a progress bar appears with the status. When the algorithm finishes running, the progress bar disappears, and the results appear as a file in selected format in a chosen location. For more information about the dialog box options, refer to Figure 4.

<b>VOI Region</b>	Applies the algorithm to the volume delineated by the VOI.
<b>Mask Image</b>	Applies the algorithm to the volume within a chosen mask. You must apply the mask, first.
<b>Intensity Level</b>	Applies the algorithm to the region with the chosen intensity level. By default, the intensity level value is set to 50.
<b>Blur by (Std.Dev.)</b>	Blurs the image to improve the extracted surface. The value specifies the standard deviation of the Gaussian filter used to regularize the image.
<b>Decimate Surface</b>	If selected, reduces the count of triangle facets, which results the more compact algorithm.
<b>Choose</b>	Use this button to select a catalogue where you would like to store your results and choose a file format (*.txt, *.sur, or *.wrl).
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes it.
<b>Help</b>	Displays online help for this dialog box.



**Figure 4. The Extract Surface dialog box options**





## EXAMPLES

### 1. To apply the algorithm to a VOI region:

- 1 Open an image of interest.
- 2 Delineate a VOI on the image, and then call the Algorithms> Extract Surface (marching cubes).
- 3 In the Extract Surface dialog box that appears, select the VOI Region option.
- 4 Select the surface file type as an output, and then specify the file name and location.
- 5 Press OK. The algorithm begins to run, and a progress bar appears showing the status.
- 6 When the algorithm finishes running and the progress bar disappears, click the Volume Tri-planar View icon on the MIPAV toolbar.
- 7 The Resample dialog box appears.

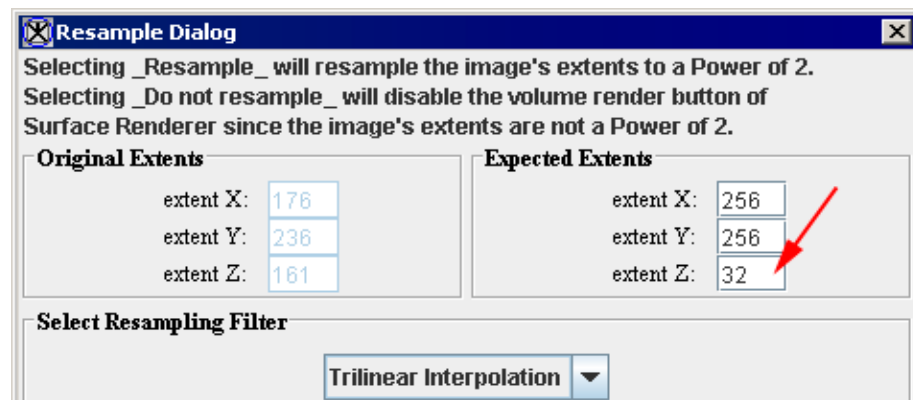


Figure 5. The Resample dialog

- 8 In the Expected Extents box, specify the value for the Z direction. The value must be in a power of two (32, 64, 128, 256). Refer to Figure 5. The bigger number you choose, the higher resolution you receive.<sup>1</sup>

1. Please, note that the Resampling is a lengthy process. Give some consideration to values you choose for Z. The algorithm's speed also depends on the MIPAV memory allocation settings. Resample size also depends on the amount of memory on the graphics card.

- 9** Press Resample to proceed.
- 10** The 3D viewer opens for the selected image.
- 11** Click the Add Surface to Viewer icon and add the surface file from Step 5 to the Surface list. Then, select the file name and press Add.
- 12** The surface will be added to the image. To adjust the display of the image, use the Surface options, such as color and opacity. Refer to Figure 6.

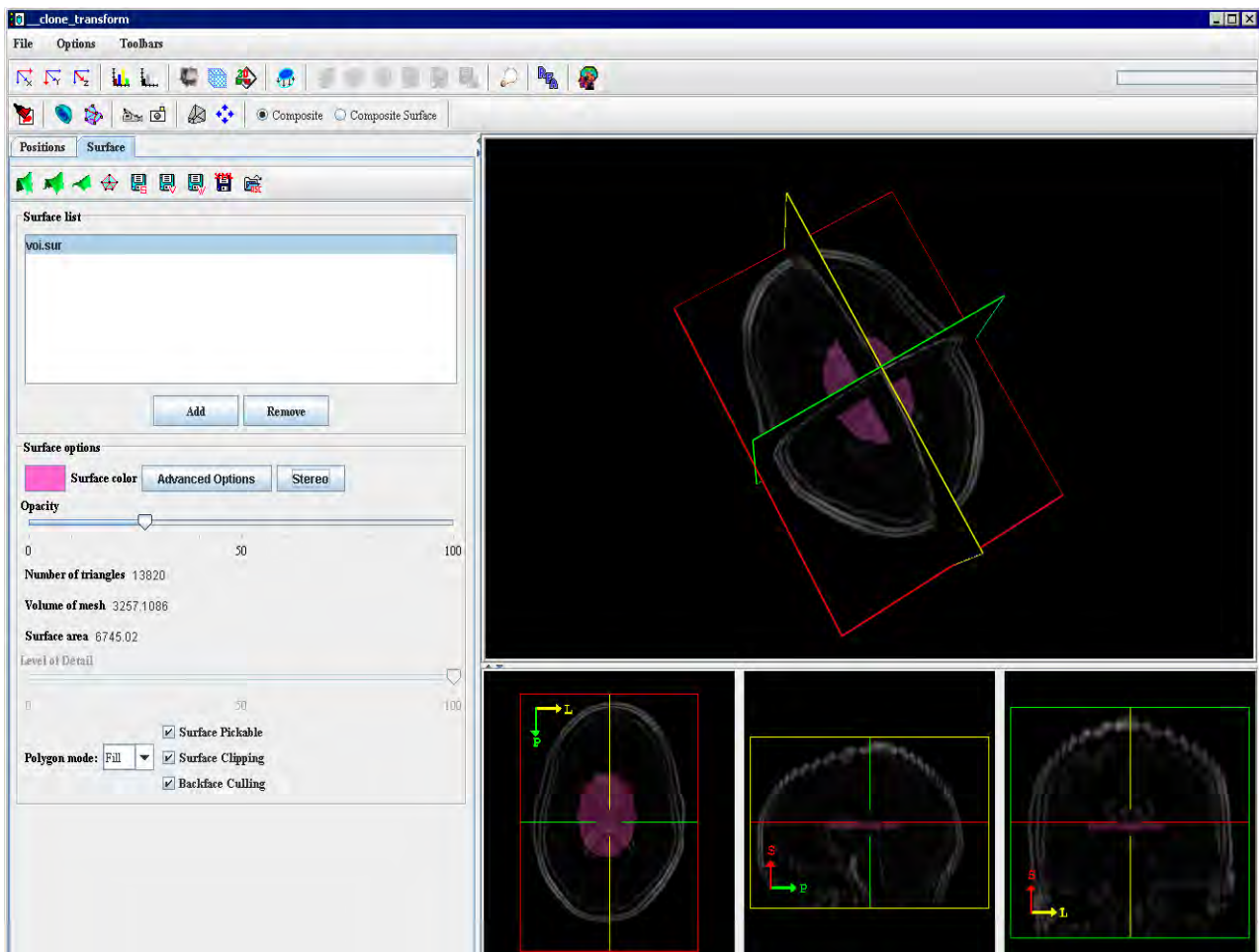


Figure 6. The view of the surface extracted using the VOI Region option

## 2. To use the Mask Image option:

- 1** Open an image of interest.
- 2** Create a mask.

- 3 Call the Algorithms> Extract Surface (marching cubes).
- 4 In the dialog box that appears, select the Mask Image option.
- 5 Select the surface file as an output, and then specify the file name and location. Press OK.
- 6 When the algorithm finishes running, click the Volume Tri-planar View icon on the MIPAV toolbar and follow the steps 6--12 from the first example on page 401. Refer to Figure 7.

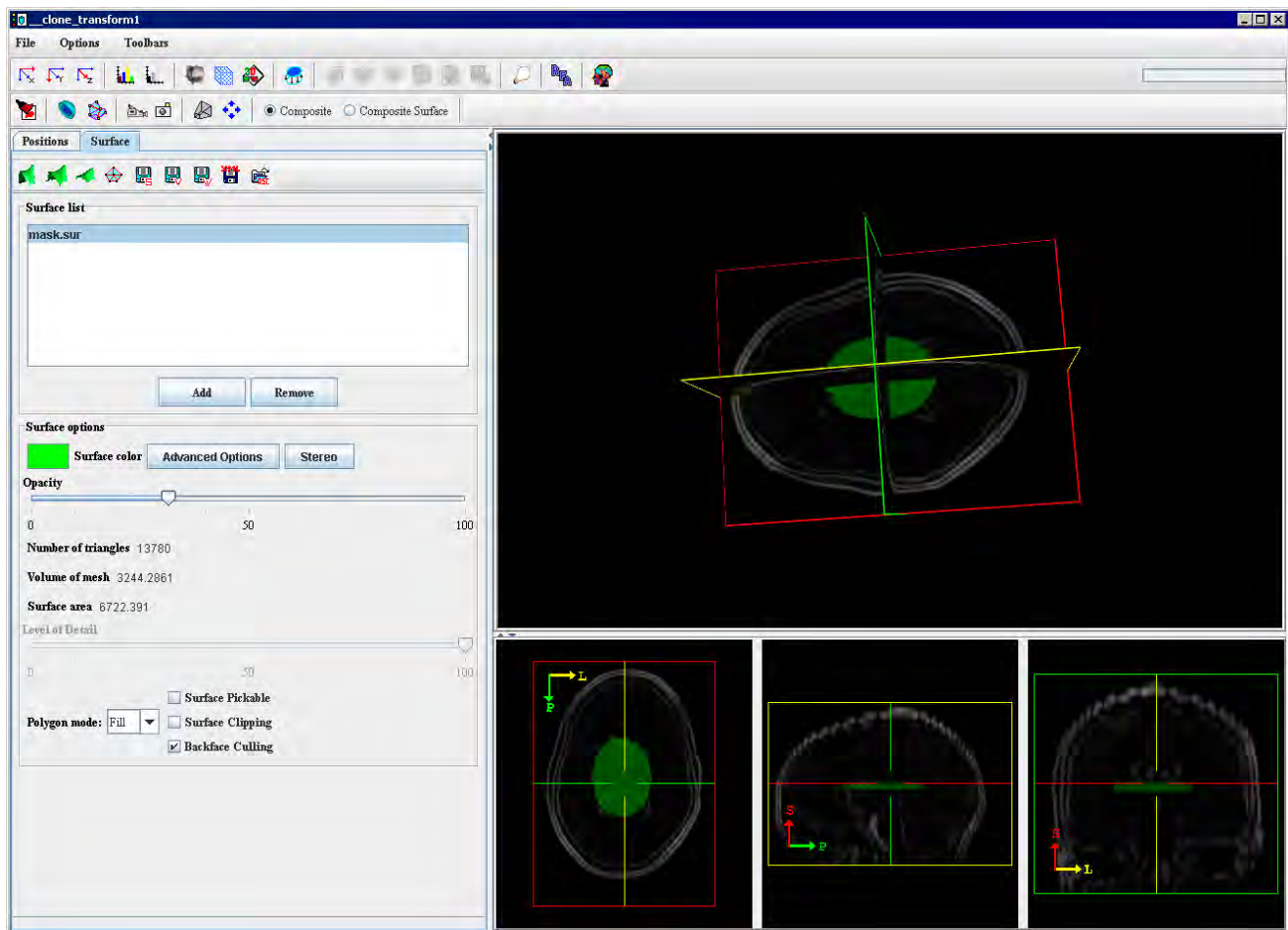


Figure 7. The view of the surface extracted using the Mask Image option

### 3. To use the Intensity Level option:

- 1 Open an image of interest.
- 2 Select the intensity level value. You can understand the intensity level by mousing over the image. The intensity value for the selected pixel is shown on the main MIPAV toolbar.

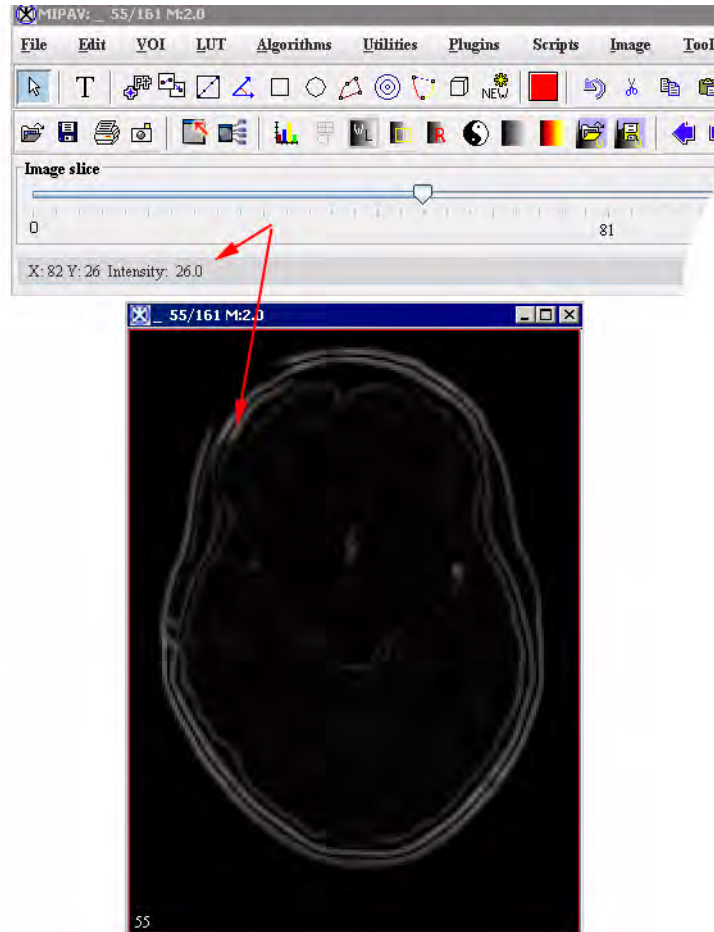


Figure 8. The intensity level is shown on the main MIPAV toolbar

- 3 Call the Algorithms> Extract Surface (marching cubes).
- 4 In the dialog box that appears, select the Intensity Level option.
- 5 Enter the intensity value that you would like to apply. Refer to Figures 8 and 9.

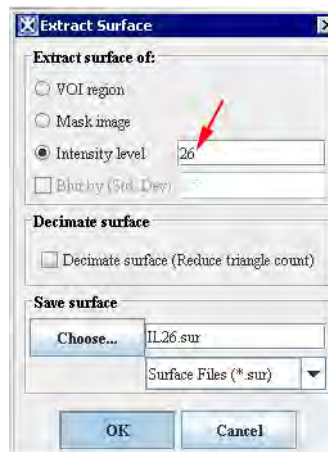


Figure 9. Enter the intensity level value here

- 6 Select the surface file as an output, and specify the file name and location. Press OK to run the algorithm.
- 7 When the algorithm finishes running, click the Volume Tri-planar View icon on the MIPAV toolbar and follow the steps 7–12 from the first example on page 401.

Figure 10 shows the surface extracted using the Intensity Level option.

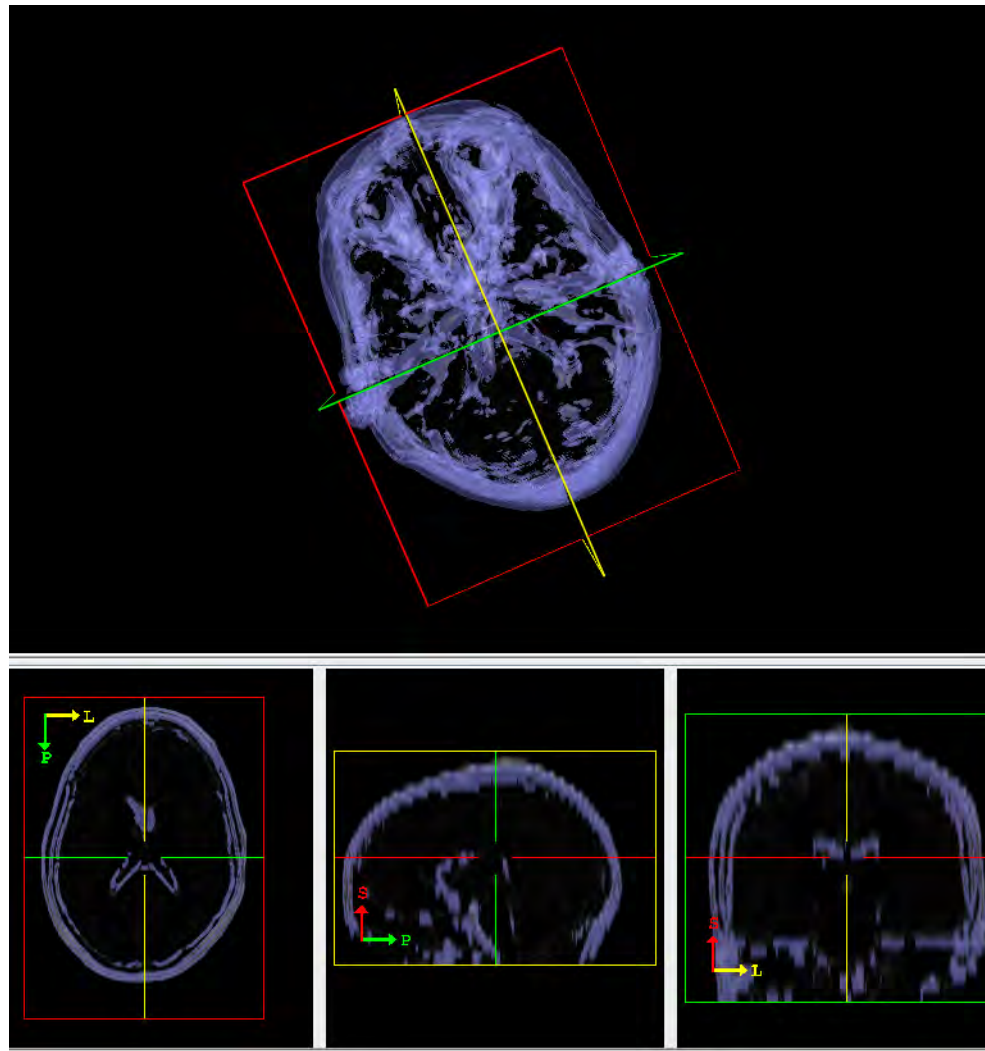


Figure 10. The view of the surface extracted using the Intensity Level option

## Mask

*This algorithm generates a mask, with a specified intensity, of the region inside or outside the contoured VOI groups that are delineated on an image. This algorithm is an alternative to the other mask-generation methods discussed in Volume 1 of the MIPAV User's Guide.*

**Tip:** If you have already used contours to delineate a VOI, you might want to use this method because it provides a quick means to create a mask.

## Background

When you apply this algorithm, enter the intensity value that fills the mask (contour). Then, indicate whether you want the algorithm to fill the area inside or outside of the contoured areas. When this algorithm runs, it determines which areas are contoured and fills the areas inside or outside of the contour with the specified intensity value (refer to Figure 1).

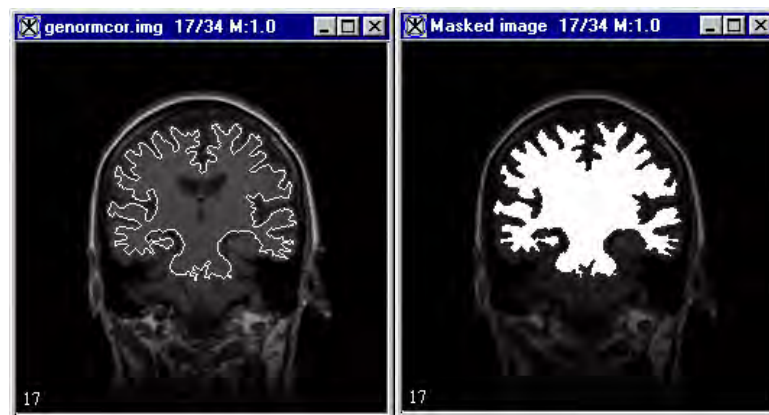


Figure 1. Mask algorithm processing

---

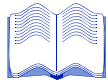
## IMAGE TYPES

You can apply this algorithm to 2D, 2.5D, 3D, and 4D images. However, you cannot apply the algorithm to RGB images.

---

## SPECIAL NOTES

If the intensity is higher or lower than the original extremes, the LUT is modified so that the new intensity is incorporated. After this algorithm is run, you may need to manually adjust the LUT so that you can see the structures clearly.



## REFERENCES

There are no references for this algorithm.

## Applying the Mask algorithm

To run this algorithm, complete the following steps:

- 1** Open an image, in Default view, in an image window.
- 2** If not done already, contour the desired VOIs. MIPAV applies the mask to every VOI in the dataset, regardless of group affiliation. For more guidance on contouring a VOI, refer to volume 1 in the *MIPAV User's Guide*.
- 3** In the MIPAV window, select Algorithms > Mask. The Mask Image dialog box (Figure 2) appears.




<p><b>Value used to fill VOI (s)</b></p>	<p>Indicates the intensity value that replaces the existing value inside or outside the region bound by the VOI contours.</p> <p>To replace the intensity value inside the VOI region, type a value in this text box and then select the Interior/exterior fill check box.</p> <p>To replace the intensity value outside the VOI region, simply type a value in this text box.</p>	
<p><b>Interior/exterior fill</b></p>	<p>If selected, changes the intensity values of the region bound by the VOIs to the intensity value indicated in Value used to fill VOI(s).</p>	
<p><b>New image</b></p>	<p>If selected, indicates that the results of the algorithm appear in a new image window.</p>	
<p><b>Replace image</b></p>	<p>If selected, indicates that the results of the algorithm replace the current active image.</p>	
<p><b>OK</b></p>	<p>Performs the Mask algorithm on the selected images based on your choices in this dialog box.</p>	
<p><b>Cancel</b></p>	<p>Disregards changes you made in this dialog box and closes the dialog box.</p>	
<p><b>Help</b></p>	<p>Displays online help for the algorithm.</p>	

Figure 2. Mask Image dialog box

- 4 Enter the parameters and indicate whether to fill the region inside or outside of the region bound by the VOI contours. Enter the destination information (refer to the previous table for more information).
- 5 When complete, click OK.

A status window appears with the following message: “Masking” to indicate that the software is generating the mask by replacing the intensity levels. MIPAV then remaps the intensity values in the dataset to the LUT. The new masked image appears.



**Tip:** If the resulting masked image looks significantly lighter or darker than the original image, the new intensity value that you specified may be much higher or lower than the majority of other intensity values in the dataset. To change the LUT, do one of the following: (1) Click Quick LUT. Move the cursor to the resultant masked image and draw a box around an area where the intensity value was not changed. MIPAV uses these values to remap the image data to the LUT. (2) Adjust the LUT in the Lookup Table window. For more information on how to do this, refer to the *MIPAV User's Guide, Volume 1*.

---

---

## Microscopy: Colocalization Orthogonal Regression

*This algorithm quantifies automatically the amount of colocalization in images. Recently, this method was used to quantify protein-protein interactions from colocalization data analysis of two-color confocal microscope images. The method was accurate enough to determine dissociation rates of two interacting proteins labeled with fluorescence reporters in live cells. This method may be helpful in answering other biological questions that involve protein-protein interactions or co-compartmentalization.*

---

**Disclaimer:** Colocalization cannot distinguish interaction from co-compartmentalization for compartments below light resolution (i.e., distances less than 200 nm).

---

The theory of colocalization assumes that a high level of similarity between two spatial patterns implies proximity of the two proteins, which implies direct or indirect protein binding, or co-compartmentalization, of the proteins.

Specifically, the algorithm creates a 2D histogram from two colors of a single image or from two black-and-white images. It uses an orthogonal line fit of the histogram data to generate a correlation line through the histogram. Upper right region rectangles with lower left corners on the correlation line are used as colocalization regions, regions where both of the two colors are significantly above background. The colocalization frame is tied to the original or modified source image or images, therefore only the pixels located in the colocalization region are displayed. A user-movable point allows for the display of source image pixels and statistics associated with different upper right colocalization regions. When not in free range mode, the point must move along the correlation line. In free range mode, the point can move anywhere in the histogram buffer.

In summary regarding the usage of this algorithm, one must make sure that the images acquired have low noise levels and no bleed through and that the optical setup used for each color leads to the same point spread function (PSF) and is free of registration errors. Assumptions made in this approach

are that pixel intensities linearly relate to concentration of molecular species being measured and that a pixel is either colocalized or not.

## Background

Based on the type of image, when this algorithm is used, either the Orthogonal Regression Colocalization dialog box for color images or the Orthogonal Regression Colocalization dialog box for grayscale images (Figure 1 on page 421) opens.

As an option, the Colocalization Orthogonal Regression algorithm can perform background level subtraction on images. If the Subtract average VOI level check box is selected, then the algorithm calculates the average pixel levels in the yellow VOI in one-color images or in the two yellow VOIs in two black-and-white images. It then subtracts these two background levels from the 1 or 2 source images. If a registration was also selected, the background determinations occur before registration, and subtraction occurs after the registration. After the subtraction is performed, any pixels with negative values are set to zero.

As an option this algorithm can perform image registration before performing colocalization. If you select the Registration before Colocalization check box on the Orthogonal Regression Colocalization dialog box, this algorithm runs another algorithm—the Optimized Automatic Registration (OAR) 2D algorithm—first. In this registration, both images or colors are moved halfway to a common center point. The registration is always done on the entire image. The only registration parameter that you can vary is the cost function. Although correlation ratio is the default cost function, you can select least squares, normalized cross correlation, or normalized mutual information instead.

After registration, this algorithm performs colocalization on the registered image or images rather than on the original image. After subtraction, this algorithm performs colocalization on the subtracted image or images rather than on the original image. If the maximum data range is decreased, then the number of histogram bins decreases if necessary so as not to exceed the actual number of channels in the data range.

The amount of colocalization can be measured with the linear correlation coefficient.

Let  $L$  be the linear correlation coefficient

Let the two channels be  $R$  and  $G$

Let the average values be  $\bar{R}$  and  $\bar{G}$

Then

$$L = \frac{\sum_i ((R_i - \bar{R}) \times (G_i - \bar{G}))}{\sqrt{\sum_i ((R_i - \bar{R})^2) \times \sum_i ((G_i - \bar{G})^2)}}$$

This is equivalent to the more traditional form of the equation:

$$\frac{N \sum R_i G_i - \sum R_i \sum G_i}{\sqrt{N \sum R_i^2 - (\sum R_i)^2} \times \sqrt{N \sum G_i^2 - (\sum G_i)^2}}$$

where  $N$  is the number of pixels.

The linear correlation coefficient ranges from -1 to 1 with 0 the value for random patterns and 1 the value for 100-percent colocalization. Negative values of the coefficient are not used for colocalization since that would indicate an anticlocalized situation with a bright pixel in one channel implying a dim pixel in another channel.

**Note:** The medical literature on colocalization refers to the traditional linear correlation coefficient as *Pearson's coefficient*.

This algorithm uses the linear least squares method of calculating the best line fit for  $\text{green} = a \times \text{red} + b$ . It uses an orthogonal line fit where the minimized distance measurements are orthogonal to the proposed line. This is an orthogonal regression as opposed to a traditional direct regression of dependent  $y$  variable green on independent red variable  $x$ , which would minimize the vertical distances between the points and the fitted line. An inverse regression of red on green would minimize the horizontal distances between the points and the fitted line. All three lines pass through the point  $(\bar{x}, \bar{y})$ , but the three lines only coincide if the correlation coefficient equals -1 or 1. Use of a direct regression only makes sense if one variable is dependent and one variable is independent. Since

the red and green dyes used in colocalization are best thought of as two independent variables, the orthogonal regression is most appropriate.

---

**Note:** Orthogonal regression is also called *total least squares*.

---

Points that are part of the dataless background in both images are excluded from the regression line calculations. Let  $R$  be the red channel and  $G$  be the green channel. If  $R_i < \text{background1}$  and  $G_i < \text{background2}$ , the point  $i$  is not included in the regression line calculations. If the image is not a floating point image,  $\text{background1} = 1.0$  and  $\text{background2} = 1.0$ . In floating point images, the image minimums are used as backgrounds so that no points are excluded. Note that there are two VOIs: the yellow background subtraction VOI and a nonyellow VOI used to delineate the region of the image included in calculations. You can select to use points from the entire image, only points from the contour VOI region, or only points allowed by a mask file.

---

**Tip:** If VOI region is selected, then, after the algorithm is run, you can move the VOI by dragging it with the mouse to another region on the image. When the mouse button is released, the calculation is performed again updating the histogram. This allows the calculation to be performed with the same contour VOI in different positions.

---

The optional second iteration, thresholding, is different from the first iteration thresholding. In the first iteration, points are only excluded if they are below both  $\text{background1}$  and  $\text{background2}$ . However, in the second iteration, points are excluded if they are below either  $\text{lastPositive1}$  or  $\text{lastPositive2}$ . The ( $\text{lastPositive1}$ ,  $\text{lastPositive2}$ ) is the last point going from top to bottom on the first iteration regression line before an  $L$ -shaped subthreshold region with a zero or negative linear correlation coefficient is found. In the second iteration, the  $L$ -shaped subthreshold region determined from the first iteration is excluded from the regression line calculations. The default is to not perform a second iteration.

The slope and offset of the orthogonal regression line can be obtained from the eigenvector of the smallest eigenvalue of the 2 by 2 matrix:

$$\begin{bmatrix} (count - 1) \times \text{sample } y \text{ variance} & - (count - 1) \times \text{sample } xy \text{ covariance} \\ - (count - 1) \times \text{sample } xy \text{ covariance} & (count - 1) \times \text{sample } x \text{ variance} \end{bmatrix}$$

The eigenvector has the components (direction  $x$ , direction  $y$ ).

$$\text{slope of the line} = \frac{\text{direction } y}{\text{direction } x} = \frac{Vy - Vx + \sqrt{(Vy - Vx)^2 + 4Cxy^2}}{2Cxy}$$

where

$S$  = slope of the line

$V$  = variance

$C$  = covariance

$$\text{offset} = \bar{y} - S \times \bar{x}$$

Let  $n$  be the normal vector to the orthogonal regression line.

Let  $\theta$  be the angle from the  $x$  axis to the normal.

$$n = \bar{x} \times \cos \theta + \bar{y} \times \sin \theta$$

where

$$\theta = \arctan(\text{slope})$$

The minimized residue, the mean square error (MSE), is given by the following:

$$\begin{aligned}
 MSE &= \text{variance } x \times (\cos \theta)^2 + \text{covariance } xy \times \sin 2\theta + \text{variance } y \times (\sin \theta)^2 \\
 &= (\text{variance } x + 2\text{slope} \times \text{covariance } xy + \text{slope}^2 \times \text{variance } y) / (1 + \text{slope}^2)
 \end{aligned}$$

A histogram buffer is generated with bin1 counts along the  $x$  axis and bin2 counts along the  $y$  axis. The Orthogonal Regression Colocalization dialog box allows you to specify the two bin numbers. The default is the range of the data (max - min +1) or 256, whichever is less, for nonfloating point images and 256 for floating point images.

The algorithm calculates the linear correlation coefficients for all pixels whose values are either below threshold in buffer or are below  $a \times \text{threshold} + b$  in secondBuffer. This is the  $L$ -shaped subthreshold region. The algorithm calculates along the color that has the greatest range along the line segment. The line segment's (color1,color2) point just above the point where the linear correlation coefficient becomes negative or zero is taken in a nonrigorous fashion as the threshold point separating upper right rectangular colocalized data from the lower left  $L$ -shaped noncolocalized data.



The algorithm calculates the percent colocalization area, the percent red colocalization, and the percent green colocalization. Each point has a red value = threshold and a green value = secondThreshold. Thus, each point defines an upper right rectangular region characterized by  $red \geq threshold$  and  $green \geq secondThreshold$ .

The percent colocalization area is 100 times the number of pixels belonging to the upper right rectangular region divided by the number of pixels in the entire image or selected VOI.

If Limit colocalization to pixels $\geq$ threshold is . . .		
Percent	Selected	Not selected (default)
Red colocalization	$100 \times \frac{R}{R_T}$ <p>where  <math>R</math> = Sum of the red values in the upper right rectangular region  <math>R_T</math> = Sum of the red values <math>\geq</math> red threshold in the image or selected VOI</p>	$100 \times \frac{R}{R_I}$ <p>where  <math>R</math> = Sum of the red values in the upper right rectangular region  <math>R_I</math> = Sum of the red values in the image or the selected VOI</p>
Green colocalization	$100 \times \frac{G}{G_T}$ <p>where  <math>G</math> = Sum of the green values in the upper right rectangular region  <math>G_T</math> = Sum of the green values <math>\geq</math> green threshold in the image or selected VOI</p>	$100 \times \frac{G}{G_I}$ <p>where  <math>G</math> = Sum of the green values in the upper right rectangular region  <math>G_I</math> = Sum of the green values in the image or the selected VOI</p>

**Note:** Only pixels with at least one of the two levels above background that you specify in the Orthogonal Regression Colocalization dialog box are included in any of the above sums.

---

## IMAGES TYPES

You can apply this algorithm to both color and black-and-white 2D or 3D images.

---

## SPECIAL NOTES

Limitations to colocalization include the following:

- The inability to distinguish binding proteins from nonbinding proteins located within a cellular compartment smaller than the resolution of light microscopes (rough 200 nm)
- All current colocalization methods, including this one, force each pixel to be classified as either entirely colocalized or entirely noncolocalized signals. Clearly, this is not the case. Unfortunately, a method that treats each pixel as the sum of colocalized and noncolocalized signals has not been developed yet.
- An important note regarding quantitative colocalization is the need to use low-noise images to make accurate measurements since two identical signals become more and more uncorrelated as their signal-to-noise ratio decreases. The amount of noise in a region of interest can be determined by measuring the Pearson correlation coefficient of two consecutive acquisitions of the same channel. For example, one could try to keep all images with a correlation above 90 percent (e.g., increasing laser power, using more averages). Also, in order to compute accurately each colocalized fraction, the background must be subtracted from each channel.
- Other important notes regarding the usage of this algorithm: One must make sure that the images acquired have low-noise levels and no bleedthrough, and the optical setup used for each color leads to the same PSF and is free of registration errors. Assumptions made in this approach are that pixel intensities linearly relate to concentration of molecular species being measured and that a pixel is either colocalized or not.

---

## REFERENCES

See the following references for more information about this algorithm:

Sylvain V. Costes, Dirk Daelemans, Edward H. Cho, George Pavlakis, and Stephen Lockett, "Protein-protein interaction quantified by microscopic co-localization in live cells."

Gishan Dissanaikie and Shiyun Wang, Equations for slope and offset of the orthogonal least squares problem, "A Critical Examination of Orthogonal Regression and an application to tests of firm size interchangeability."

E. M. M. Manders, J. Stap, G. J. Brakenhoff, R. Van Driel, and J. A. Aten, "Dynamics of three-dimensional replication patterns during the S-phase, analysed by double labelling of DNA and confocal microscopy," *Journal of Cell Science*, 103:857-862, 1992.

E. M. M. Manders, F. J. Verbeek, and J. A. Aten, "Measurement of co-localization of objects in dual-colour confocal images," *Journal of Microscopy*, Vol. 169, Pt. 3:375-382, March 1993.

Eigenvalue software for solving the orthogonal least squares problem is found at the internet site [www.magic-software.com](http://www.magic-software.com).

Jan A. Van Mieghem, Hadar I. Avi-Itzhak, and Roger D. Melen, Equations for slope and mean square error of the orthogonal least squares from "Straight Line Extraction Using Iterative Total Least Squares Methods," *Journal of Visual Communication and Image Representation*, Vol. 6, No. 1, pp. 59-68, March 1995.

---

## Applying the Microscopy: Colocalization Orthogonal Regression algorithm

To run this algorithm, complete the following steps:

- 1** Open either one color or two black-and-white images.
- 2** Select Algorithms > Microscopy > Colocalization > Orthogonal Regression.

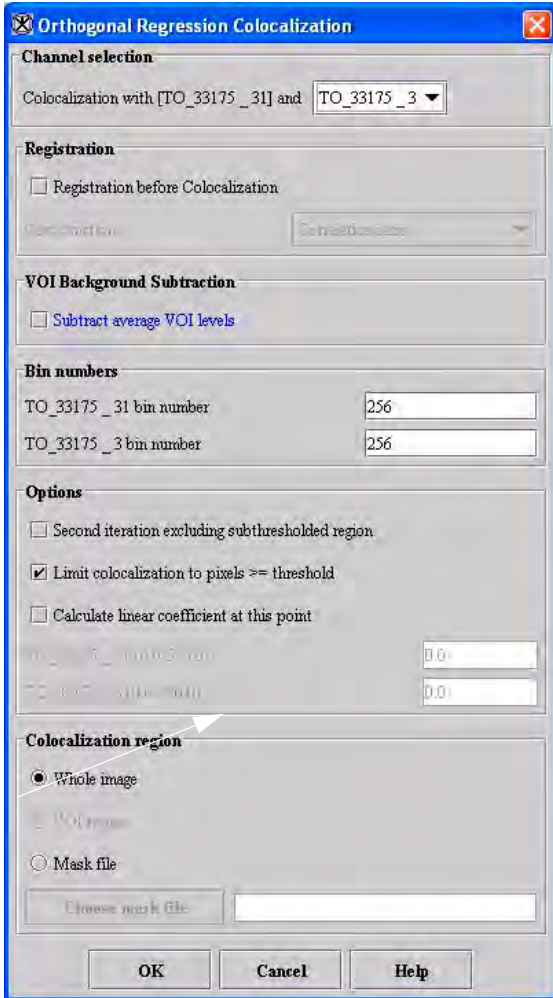
Depending on whether the image is grayscale or color, the Orthogonal Regression dialog box for grayscale images (Figure 1) or the Orthogonal Regression box for color images (Figure 2) opens.

- 3** Complete the information in the dialog box.
- 4** Click OK.

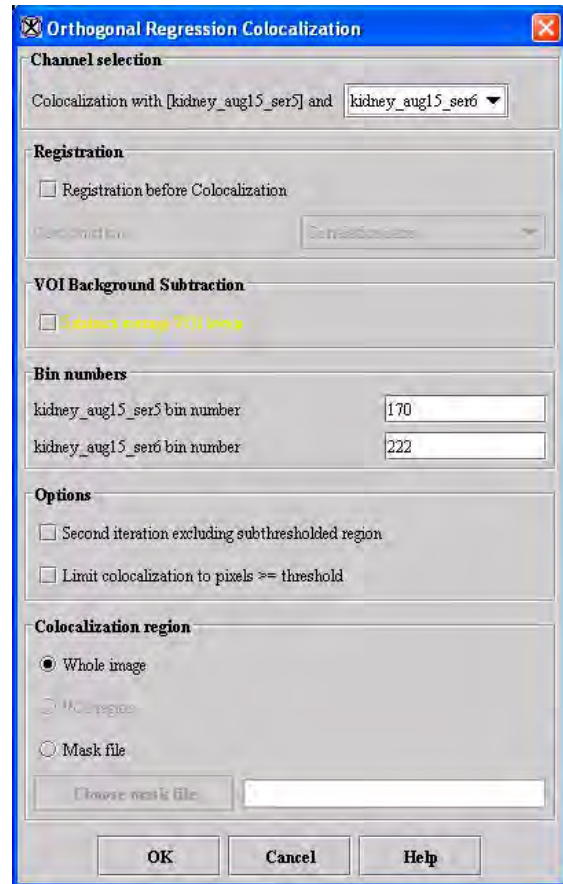
The algorithm begins to run, and a progress bar appears with the status. When the algorithm finishes running, the progress bar disappears, and a 2D histogram with a correlation line and statistics for the threshold point appears in a new window.

- 5** Use the mouse to move the point to see the statistics for different upper right rectangular colocalization regions. This colocalization frame is tied to the original or modified color source image or to the two original or modified grayscale source images. Only those pixels belonging to the upper right colocalization regions are visible in the source images. If you want to move the point off the line, select Free range mode.

If you created a VOI on the source image and selected the VOI region in the dialog box, then you can drag the VOI with the mouse. When you release the mouse, the algorithm recalculates using the new VOI position and updates the histogram.



(A) Orthogonal Regression Colocalization dialog box for color images



(B) Orthogonal Regression Colocalization dialog box for grayscale images

Figure 1. Orthogonal Regression Colocalization dialog boxes for (A) color and (B) grayscale images

<b>Channel selection</b>	<p><b>Color images:</b> Specifies the colors to use in color images.</p> <ul style="list-style-type: none"> <li>• If only two of the channels contain data, then only a label appears. For example, if only red and green are present, then the label Colocalization of red with green appears.</li> <li>• If all three color channels have data, then red, green, and blue check boxes are present. Select two of the three check boxes. By default red and green are selected.</li> </ul> <p><b>Grayscale images:</b> Specifies the second grayscale image to be used. Select one from the list box.</p>
<b>Registration before colocalization</b>	<p>Performs registration on an entire image before performing colocalization. When you select Registration before colocalization, the Cost function box becomes available.</p> <p>By default, registration is not selected.</p>
<b>Cost function</b>	<p>Specifies the cost function to be used during registration. You can specify correlation ratio (default), least squares, normalized cross correlation, or normalized mutual information.</p> <p>To make this box available, you must first select Registration before colocalization.</p>
<b>VOI background subtraction</b>	<p>Calculates the average background levels in the yellow VOI and subtracts them from the rest of the image if Subtract average VOI level is selected. The subtraction results are clamped to be <math>\geq</math> zero. The default is no subtraction.</p>
<b>Bin numbers</b>	<p>Specifies the number of histogram bins to be used in the 2D histogram.</p> <p>The default is the range of the data or 256, whichever is less, for nonfloating point images and 256 for floating point images.</p>
<b>Options</b>	<p><b>Second iteration excluding subthresholded region</b>—Excludes the L-shape subthreshold region determined from the first iteration from the regression line calculations.</p> <p>By default, this check box is clear.</p> <hr/> <p><b>Limit colocalization to pixels <math>\geq</math> threshold</b>—Uses, in red colocalization, only pixels with red values <math>\geq</math> red threshold in the denominator sum and, in green colocalization, only pixels with green values <math>\geq</math> green threshold in the denominator sum. The default value is not selected.</p>

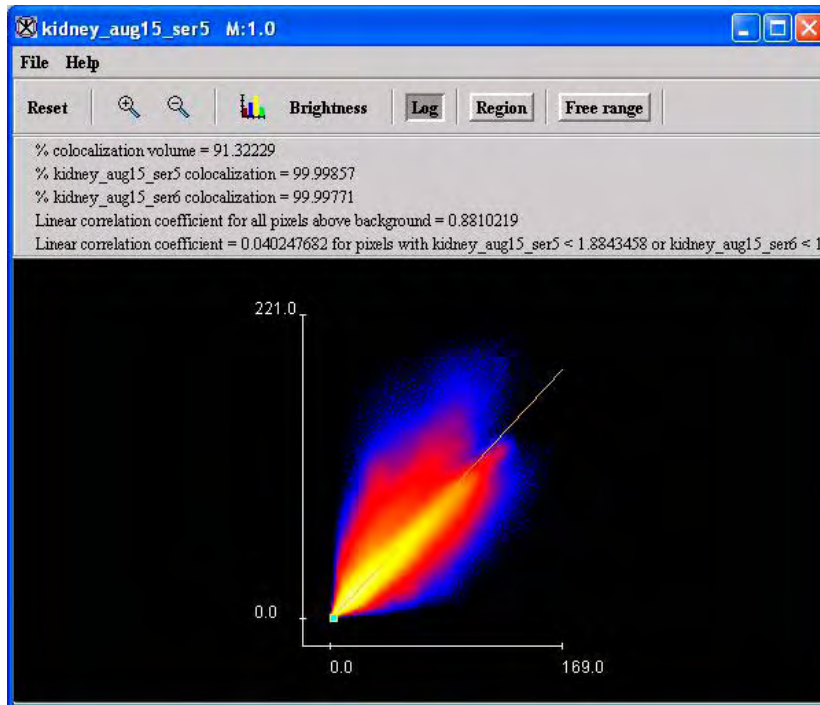
Figure 1. Orthogonal Regression Colocalization dialog boxes for (A) color and (B) grayscale images (continued)

<b>Colocalization region</b>	<b>Whole image</b> —Makes calculations on the whole image. By default, this option is selected.
	<b>VOI region</b> —Makes calculations on only the pixels in the VOI. To enable this option, a contour VOI must be present in the image.  After this algorithm run and the histogram is produced, if you drag the VOI with the mouse and then release the mouse button, the system recalculates using the new VOI position.
	<b>Mask file</b> —Makes calculations on only the pixels in the mask.  Selecting this option enables the Choose mask file button.
	<b>Choose mask file</b> —Allows you to select a mask file.
<b>Free range</b>	Toggles between requiring the point to be in the least squares line and allowing the point to be anywhere in the histogram buffer. By default, the free range mode is not selected.
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

**Figure 1. Orthogonal Regression Colocalization dialog boxes for (A) color and (B) grayscale images (continued)**

## COLOCALIZATION HISTOGRAM WINDOW

The Colocalization Histogram window (Figure 2) contains a 2D histogram with a correlation line and a mouse-movable point on the correlation line. As the point is



<b>File</b>	<b>Close Colocalization</b> —Closes the colocalization window.
<b>Help</b>	<b>About Colocalization</b> —Displays online help for this window.
<b>Reset</b>	Moves the movable point to the position where the linear correlation coefficient of the L-shaped subthreshold region has gone from negative or zero to positive.
<b>Magnify image 2.0X</b>	Magnifies the histogram by 2.
<b>Magnify image 0.5X</b>	Magnify the histogram by one-half.

Figure 2. 2D Histogram window



<b>Displays Lookup Table (LUT)</b>	Displays the Lookup Table (LUT) dialog box, which allows you to specify the LUT for this histogram.
<b>Brightness</b>	Displays the Brightness/Contrast dialog box, which allows you to adjust the brightness and contrast in the histogram.
<b>Log</b>	Toggles the histogram between showing counts and $\log_{10}(1 + \text{histogram counts})$ . The default is no log function.
<b>Region</b>	Toggles the histogram between not showing lines and showing the bottom and left lines for the upper right colocalization region. The default is not to show the lines.

**Figure 2. 2D Histogram window (continued)**

moved, the statistics at the top of the frame change to reflect the changed upper right rectangular colocalization region whose lower left corner is selected by the point. The colocalization frame is tied to a version of the source color image or to the versions of the two source black-and-white images, so that only those source image pixels in the upper right colocalization region are visible in the source images.

The percent colocalization area, the percent red colocalization, and the percent green colocalization change with the point location. Each point has a red value = threshold and a green value = secondThreshold. Thus, each point defines an upper right rectangular region characterized by red  $\geq$  threshold and green  $\geq$  secondThreshold.

The percent colocalization area is given by 100 times the number of pixels belonging to the upper right rectangular region divided by the number of pixels in the entire image or the selected VOI. If limit colocalization to pixels  $\geq$  threshold is not selected, the percent red colocalization (refer to Figure 3) is given by 100 times the sum of the red values in the upper right rectangular region divided by the sum of the red values in the image or the selected VOI. The percent green colocalization is given by 100 times the sum of the green values in the upper right rectangular region divided by the sum of the green values in the image or the selected VOI.

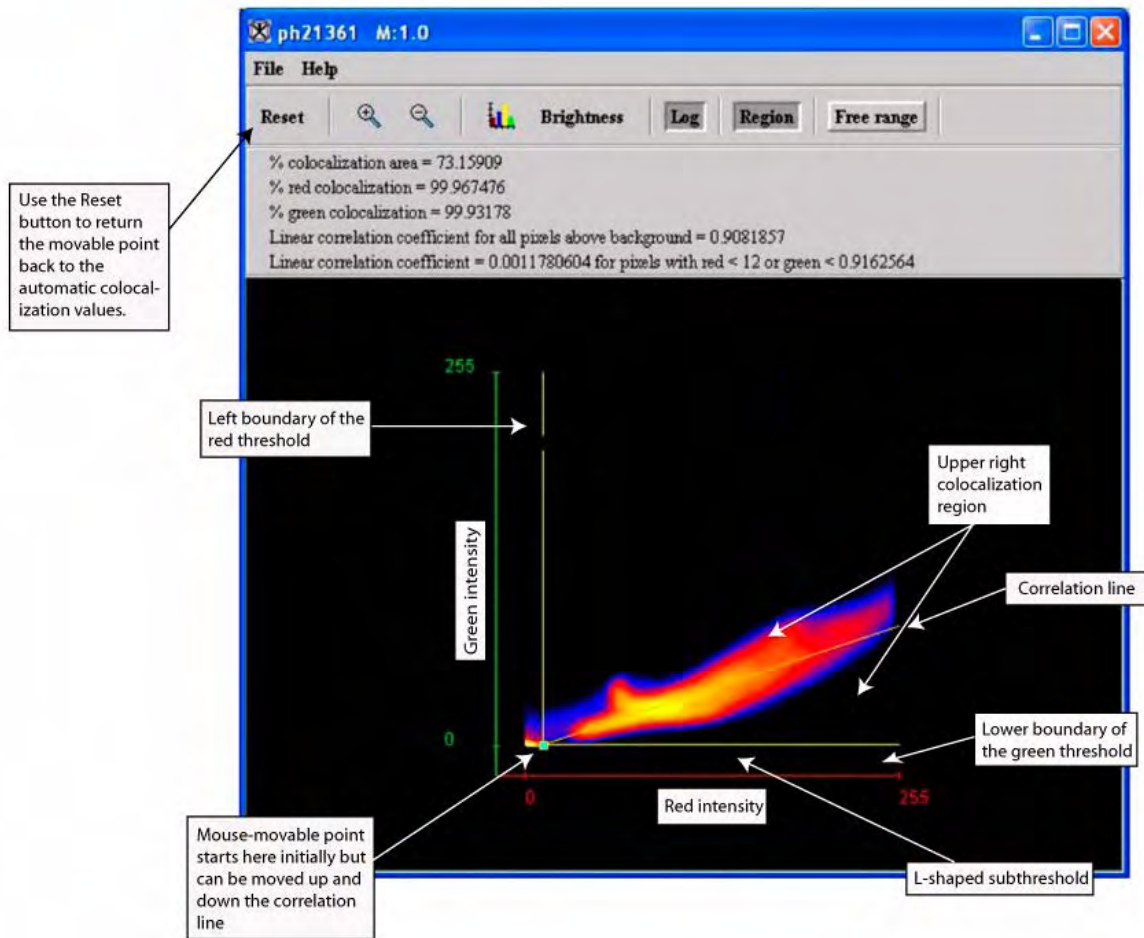


Figure 3. Colocalization Histogram window showing a histogram for a color image in which red and green color localization was done

Note that only pixels above at least one of the two backgrounds are included in any of the above sums.

If limit colocalization to pixels  $\geq$  threshold is selected, the percent red colocalization is given by 100 times the sum of the red values in the upper right rectangular region divided by the sum of the red values  $\geq$  the red threshold in the image or selected VOI. The percent green colocalization is given by 100 times the sum of the green values in the upper right rectangular region divided by the sum of the green values  $\geq$  the green threshold in the image or selected VOI.

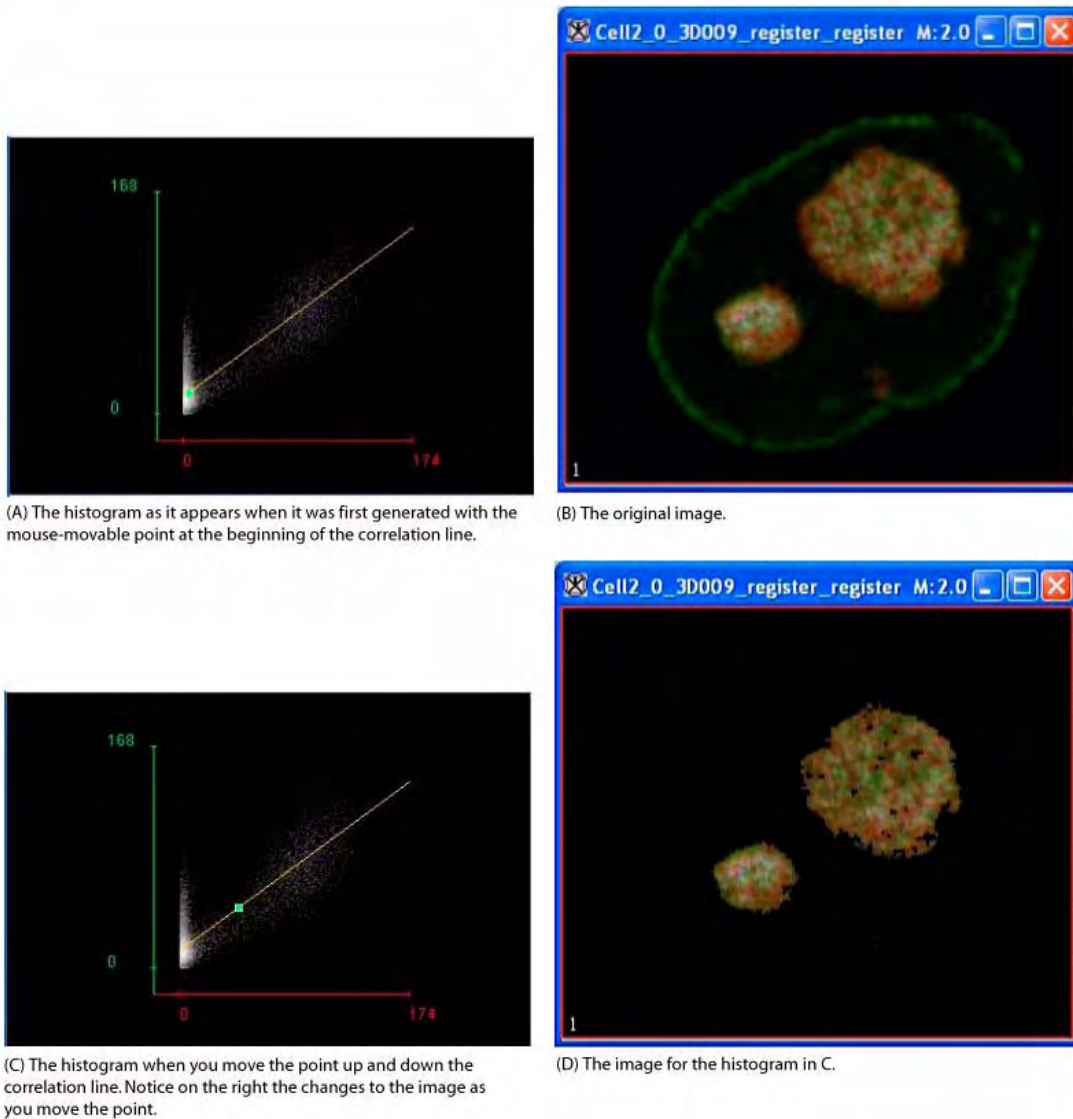
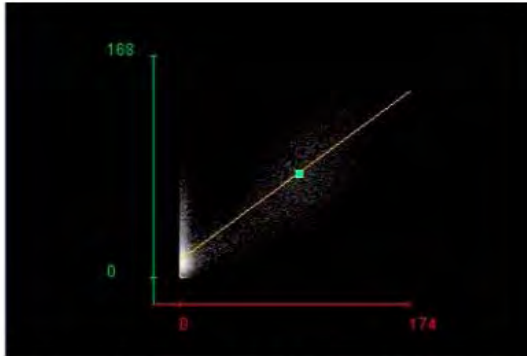


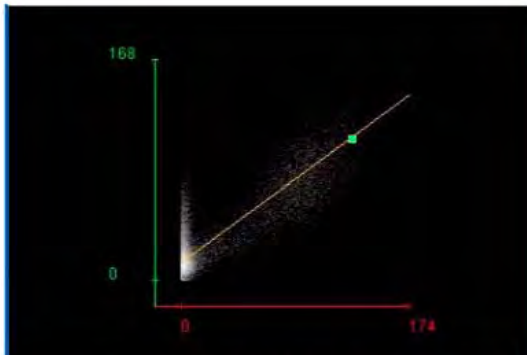
Figure 4. The effect on the image when dragging the mouse-movable point on the histogram



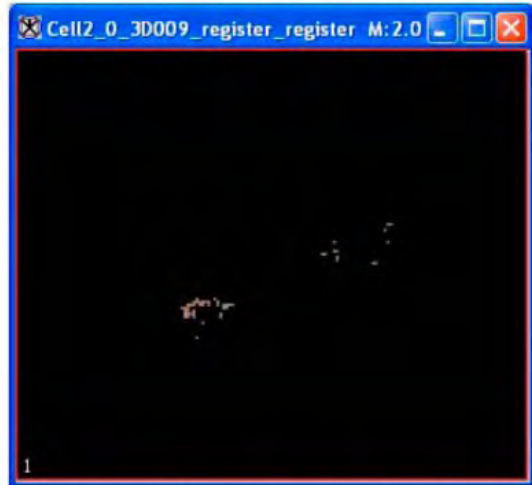
(A) Histogram after moving the point to the middle of the correlation line



(B) Image as it looks after moving the point to the middle of the correlation line on the histogram (notice the reduction on colocalization green and red pixels)



(C) Histogram in which the point has moved higher on the correlation line



(D) Image showing that the number of colocalized green and red pixels decreases when the point is moved higher on the correlation line

Figure 5. The effect on the image when dragging the mouse-movable point on the histogram

## Microscopy: Blind Deconvolution

*Blind Deconvolution is a technique, which permits recovery of the target object from a set of blurred images in the presence or a poorly determined or unknown Point Spread Function (PSF). In optical microscopy, blind deconvolution is used to remove the blurring induced by the limited aperture of the microscope objective. Instead of subtracting blur from the image, a part of the light contribution to the blurred pixels is reassigned to the correct in-focus location in the image.*

### Background

To properly reassign the intensity values in the chosen image, the imaging system must be modeled. The model includes known attributes such as the *objective numerical aperture*, the *refractive index* of the sample immersion medium, and the *wavelength* of the light used to image the sample (either reflected light or fluorescence). Refer to page 434 for more detail.

An optical imaging system is then modeled as a convolution of the target image with a PSF. Here, the PSF is assumed to have a Gaussian shape. This convolution is considered to produce the blurred image. To de-blur the image, the convolution is reversed, and the blurred image is deconvolved with the same PSF. Images can be deconvolved using a PSF that is measured directly by imaging certain types of samples, such as fluorescent beads; or the PSF can be estimated based on the optical parameters of the microscope (imaging system). In *blind deconvolution*, an estimated PSF is used, and both the estimated (de-blurred) image and the PSF image are recovered by the deconvolution process.

Blind deconvolution works through an iterative process. After the deconvolution step, the estimated image produced by the deconvolution is corrected by comparing it to the original blurred image. The same correction is also applied to the PSF image. The corrected estimated image and PSF are later used in the next deconvolution step, and are further refined through the successive iterations. See algorithm details on page 430.

An *initial estimate for the target image* is set to be the converted input image, while the *initial estimate for the PSF* is determined as a Gaussian kernel of size 3x3 for 2D images or 3x3x3 for 3D images. Once the estimate and PSF images have been initialized the algorithm computes new

estimates with the iterative process as shown below. Refer to equations 1 – 3:

EQUATION 1

$$\lambda^{(k+1)}(x, y, z) = \lambda^{(k)}(x, y, z)[\mu(x, y, z) / (\lambda^{(k)}(x, y, z) \bullet h^{(k)}(x, y, z)) \bullet h^{(k)}(-x, -y, -z)]$$

EQUATION 2

$$h^{(k+1)}(x, y, z) = \lambda^{(k)}(x, y, z)[\mu(x, y, z) / (\lambda^{(k)}(x, y, z) \bullet h^{(k)}(x, y, z)) \bullet \lambda^{(k)}(-x, -y, -z)]$$

EQUATION 3

$$h^{(k+1)}(x, y, z) = \text{constraints}(h^{(k+1)}(x, y, z))$$

Here:

$\mu(x, y, z)$  is the original input image

$\lambda^{(k)}(x, y, z)$  is the estimated input image

$h^{(k)}(x, y, z)$  is the estimated PSF

$h^{(k+1)}(x, y, z)$  is an unconstrained PSF

$\lambda^{(k)}(x, y, z) \bullet h^{(k)}(x, y, z)$  represents convolution

This algorithm uses a numerical method to calculate a *maximum-likelihood estimation* of the input image using the steps described in equations 1 – 3.

- 1** An initial estimate of the *input image* is made;
- 2** This estimate is, then, convolved with a *theoretical PSF* calculated based on the optical parameters of the imaging system. See “Computing convolution” on page 431;
- 3** The resulting *blurred estimate* is compared with the input image;

- 4 After that, a correction is computed and employed to generate a new estimate. The same correction is also applied to the PSF, generating a new PSF estimate.
- 5 The PSF estimate is constrained to form the new PSF estimate for the next iteration. Refer to “PSF constraints” below;
- 6 In the further iterations, the PSF estimate and the image estimate are updated together.

---

## COMPUTING CONVOLUTION

Convolution is computed by taking the Fast Fourier Transformation (FFT) of the images, multiplying in Fourier space, and then taking the inverse FFT. Refer to Section “Fast Fourier Transformation (FFT)” on page 191 for more information about FFT.

If the extents of the input image are not a power of 2, the nearest power of 2 for each dimension is estimated and the input image is resampled to the power of 2 for efficient computation of FFT.

---

## PSF CONSTRAINTS

It is possible to incorporate some prior knowledge about the PSF into the algorithm. It has been shown that the solution to the PSF has to be constrained in some way so that reconstructions had resemblance to the true object and true PSF. Hence, the following constraints are applied to the PSF image before the next iteration:

*Unit summation* – all intensity values in the PSF image must sum to 1.

*Hourglass* – limits the energy of the PSF so that portions of the background intensity, which lie outside an hourglass (or disk for 2D), are not assigned to the PSF.

*Band-limit and missing cone* – constraint the Fourier transform of the PSF (the optical transfer function) to be band-limited.

*Non negativity* – ensures that the PSF has no negative values.

## IMAGE TYPES

You can apply this algorithm to color and grayscale, 2D and 3D images.

**Note:** For color images, the input color image is, first, converted into grayscale. After that, the deconvolution process is performed on the grayscale representation of the image. And in the final processing step, the color values are restored by iterating once through the deconvolution process for each source channel using the estimated image and estimated PSF of the grayscale representation. All images are converted to floating point representation for processing. Refer to Figure 1.

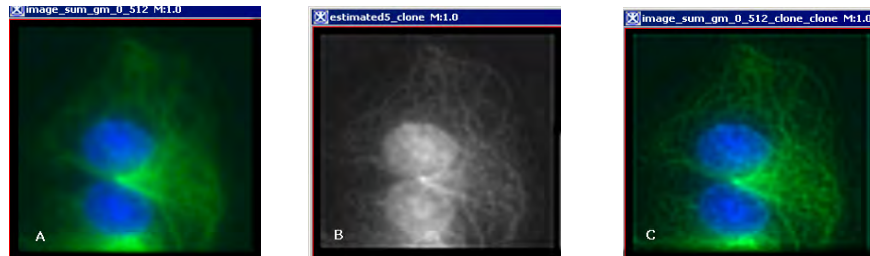


Figure 1. The original color image (A), the intermediate grayscale image (B), and the final estimate color image after 10 iterations (C).

## NOTES

Blind deconvolution eliminates the need to acquire an empirical PSF, and can be used for both microscopy and non-microscopy images. Refer to Figure 2 that shows the Jupiter image.



Figure 2. The original image of Jupiter in grayscale (A), the estimate image after five iterations (B), and the final estimate after 10 iterations (C).



---

## REFERENCES

Refer to the following sources for more information about blind deconvolution.

Holmes, T.J., Maximum Likelihood Image Restoration Adapted for Noncoherent Optical Imaging, *JOSA-A*, 5(5): 666-673, 1988.

——, Bhattacharyya, S., Cooper, J., Hanzel, D., Krishnamurthi, V., Lin, W., Roysam, B., Szarowski, D., Turner, J., Light Microscopic Images Reconstructed by Maximum Likelihood Deconvolution, Ch. 24, *Handbook of Biological Confocal Microscopy*, J. Pawley, Plenum, 1995.

——, Liu, Y., Image Restoration for 2D and 3D Fluorescence Microscopy, *Visualization in Biomedical Microscopy*, A. Kriete, VCH, 1992.

——, Blind Deconvolution of Quantum-Limited Noncoherent Imagery, *JOSA-A*, 9: 1052 - 1061, 1992

---

## USEFUL LINKS

<http://www.mediacy.com/aqi/index.asp>

<http://micro.magnet.fsu.edu/primer/digitalimaging/deconvolution/deconvolutionhome.html>

<http://micro.magnet.fsu.edu/primer/digitalimaging/deconvolution/deconintro.html>

<http://micro.magnet.fsu.edu/primer/digitalimaging/deconvolution/deconalgorithms.html>

<http://www.hamangia.freereserve.co.uk/how/index.html>

## Applying the Maximum Likelihood Iterative Blind Deconvolution algorithm

To apply the algorithm, complete the following steps:

- 1 Open an image of interest.
- 2 From the main MIPAV menu select Algorithms > Microscopy > Restoration > Maximum Likelihood Iterated Blind Deconvolution.
- 3 The Maximum Likelihood Iterated Blind Deconvolution dialog box appears.
- 4 Complete the information in the dialog box.

<b>Number of Iterations</b>	Enter the number of times to iterate through the blind-deconvolution process. The default value is 10 iterations.	
<b>Display intermediate image every</b>	Enter the frequency of display for intermediate images here. The default value is 5, which means that the intermediate images will be displayed every five iterations. Displaying intermediate results allows the user to save a series of images and adjust the PSF parameters to produce the better result. Note that the best result may also come from an earlier iteration.	
<b>Objective Numerical Aperture</b>	Enter the numerical aperture of the microscope objective used in the imaging system. The default value is 1.4.	
<b>Wavelength (nm)</b>	Enter the wavelength of the light used to image the sample, or the wavelength of the light fluorescence, in nanometers. The default wavelength is 520 nanometers.	
<b>Refractive Index</b>	This is the refractive index of the sample immersion medium. The default value is 1.518 (the refractive index of common microscope immersion oils).	

Figure 3. The Maximum Likelihood Iterated Blind Deconvolution dialog box

<b>Use microscope settings</b>	Uncheck this box to turn off the microscope-specific optical parameters. However, if the box is checked and the theoretical PSF derived from the microscope parameters contains all zero values, the algorithm will automatically disable the checkbox. Refer to Section "PSF constraints" for more information. By default, the box is checked on.
<b>Resample to the power of 2</b>	
<b>Original Extents (extent X, extent Y, and extent Z)</b>	Displays the original extents of the input image. If the image is not a 3D image, then the extent Z field is empty. A user cannot edit these values.
<b>Expected Extents (extent X, extent Y, and extent Z)</b>	Displays the expected extents of the image to the nearest power of 2. If the image is not a 3D image, then the extent Z field is empty and non-editable. It is recommended to not change the expected extents values as these are the most efficient values to the nearest power of 2.
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes this dialog box.
<b>Help</b>	Displays online help for this dialog box.

Figure 3. The Maximum Likelihood Iterated Blind Deconvolution dialog box (continued)

- 5 Press OK to run the algorithm.
- 6 The algorithm begins to run, and a pop-up window appears with the status: *Computing iterative blind deconvolution*. See Figure 4.

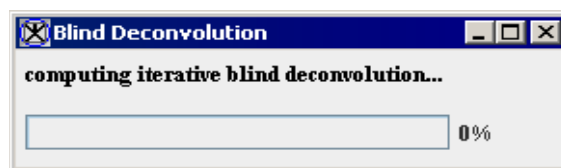


Figure 4. A status pop-up window.

- 7 While running, the algorithm produces several intermediate images or estimates which appear on the screen. The number of estimates depends on two parameters: 1) the number of iterations and 2) the frequency of appearing of intermediate images. These parameters are set in the *Number of Iterations:* and *Display intermediate images every:* fields of the dialog box.

- 8** If the *Use Microscope Settings:* box was checked, but the PSF derived from the microscope parameters contains all zero values, the algorithm automatically displays the warning as shown in Figure 5, and then, disables the checkbox. Refer to “PSF constraints” on page 431 to learn more about PSF constraints.

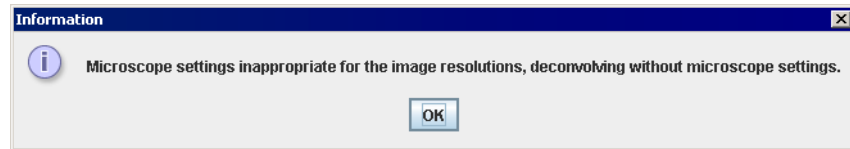


Figure 5. The PSF warning.

- 9** When the algorithm finishes running, the status window closes, and the intermediate and final results appear in new windows.

### Selecting the algorithm parameters

Careful parameter selection must be done for the Blind Deconvolution algorithm to produce good results. To find an optimal set of parameters values, apply this algorithm repeatedly on the same image (or a set of images) that you want to process with different parameter values and with a reasonable number of intermediate images selected.

Figure 6 below shows images that were produced from running the Blind Deconvolution algorithm with the default parameters against a color microscopy image. The final image was produced after 50 iterations. The intermediate images appear every five iterations.

However, the initial image was in color the intermediate images (estimates) appear in grayscale. Refer to page 434 for details. Also note that the name of the intermediate image includes the word “estimate” and the number of the iteration.




---

**Note:** *Hourglass constraint:* and *Band-limit and missing cone constraint:* mentioned on page 431 uses the Objective Numerical Aperture, Wavelength, and Refractive Index parameters provided by the user to determine the size and shape of the PSF.

---

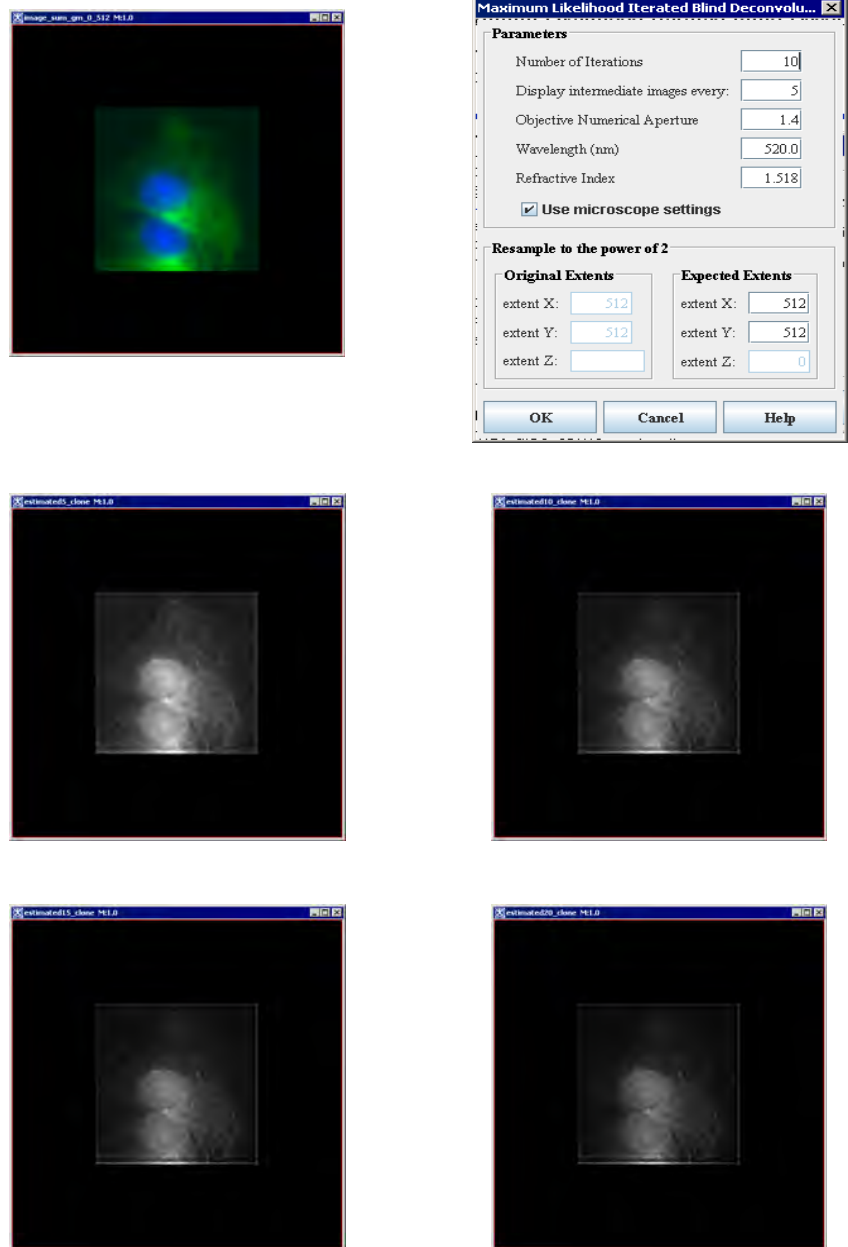
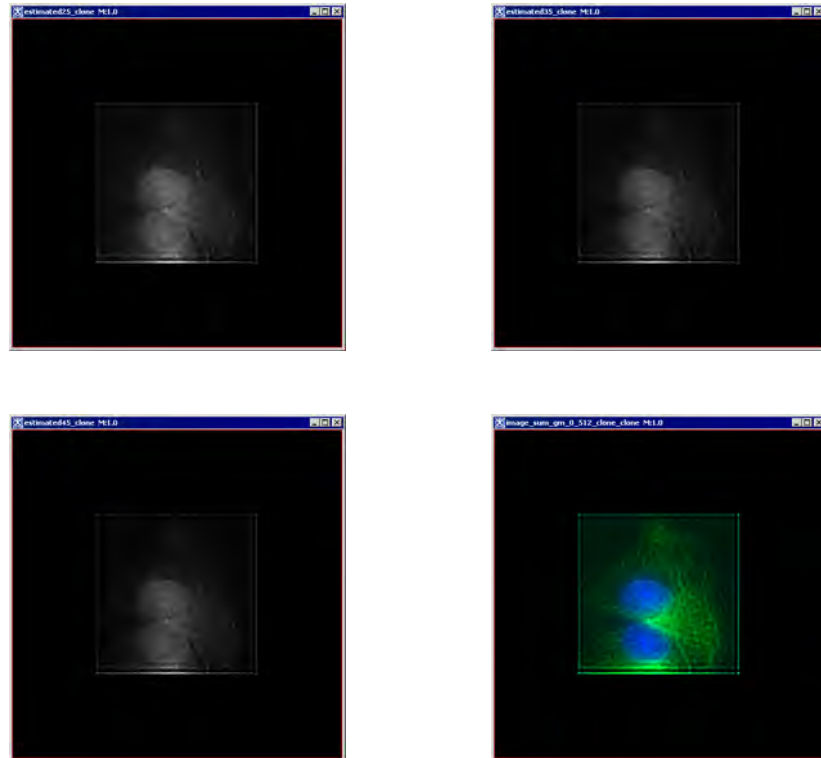


Figure 6. The Blind Deconvolution algorithm was applied on the color image (first slide) with the default parameters values (shown on the second slide). The intermediate grayscale images are also shown.



**Figure 6.** The Blind Deconvolution algorithm was applied on the color image (first slide) with the default parameters values (shown on the second slide). The intermediate grayscale images are also shown.

## Microscopy: FRAP (Fluorescence Recovery After Photobleaching)

*This algorithm allows the determination of an association rate, a dissociation rate, and a diffusion transfer coefficient by measuring the recovery of fluorescence in a photobleached area by diffusion from the nonphotobleached surround. The first images in a series of images over time are generally not photobleached. Photobleaching is then used to create a narrow band, small circle, or other small VOI where almost all of the fluorescence is removed.*

### Background

The normalized recovery of fluorescence over time  $t$  is modeled as

$$\text{bottom} + \text{span} \times (1 - \Gamma e^{\alpha t} - (1 - \Gamma)e^{\beta t}) .$$

A complete photobleaching would reduce the fluorescence to zero. However, a fluorescence level of *bottom* rather than zero may be measured after photobleaching. As time approaches infinity, the terms  $\Gamma e^{\alpha t}$  and  $(1 - \Gamma)e^{\beta t}$  go to zero, and the measured fluorescence approaches bottom plus span, which is the mobile fraction.

The algorithm uses curve fitting to determine alpha, beta, gamma, bottom, and span. It then uses alpha, beta, gamma, and the ratio of the photobleached region fluorescence after bleaching to before bleaching to determine the association rate, the dissociation rate, and the diffusion transfer coefficient.

The mathematics of FRAP is based on compartmental modeling in which the compartments must be homogeneous. That is, the fluorescence within a compartment must not have a spatial gradient. If a compartment is not homogeneous, the algorithm cannot be used. More advanced algorithms must then be used to divide the compartments into multiple compartments

---

or introduce partial differentials with respect to distance.

As an option, you may choose to perform registration before running the FRAP algorithm. During the registration, MIPAV registers slices to the first slice after photobleaching. Although the default cost function is correlation ratio, you can select instead least squares, normalized cross correlation, or normalized mutual information. By default the registered image appears in a frame.

---

**Note:** MIPAV runs FRAP on the registered image rather than on the original image.

---

The algorithm uses only one color in a color image. If a color image is registered, the registration is based only on this selected color, but all colors in the registered image undergo the same transformation.

---

**Recommendation:** Because registration takes far more time than the FRAP calculation, if more than one FRAP calculation is a possibility, save the registered image to use in future FRAP calculations.

---

For images that have a photobleached narrow band with two neighboring unbleached regions, MIPAV uses the equations described in *Using FRAP and Mathematical Modeling to Determine the In Vivo Kinetics of Nuclear Proteins* by Carrero, McDonald, et al. (refer to “References” on page 452). Similar equations were derived for images that have a photobleached circle inside an unbleached region, and solutions were also implemented for the pure 1D diffusion case and for the single exponential model.

To perform FRAP on an image, you select Algorithms >Microscopy >FRAP. The Fluorescence Recovery after Photobleaching dialog box appears. The first task is to delineate the following VOIs on the image:

- A red required photobleached VOI
- A green required whole organ VOI
- An optional blue background VOI

In delineating VOIs for FRAP, it is not necessary to select the New VOI icon. The procedure is to simply select the Add required photobleached VOI option on the FRAP dialog box and then, after selecting an Ellipse, Rectangle, Polyline, or Levelset icon in the VOI toolbar in the MIPAV



window, draw the VOI on the image. The required photobleached VOI appears in red on the image.

You return to the FRAP dialog box to create the next required VOI: the whole organ VOI. Select the appropriate option on the FRAP dialog box—in this case. Add required whole organ VOI—and then select the Ellipse, Rectangle, Polyline, or Levelset icon in the VOI toolbar and draw the VOI on the image. This VOI appears in green on the image.

As an option, you can create a third VOI by selecting Add optional background VOI on the FRAP dialog box and repeating the previous steps for drawing the VOI, which appears in blue, on the image.



**Note:** You must create these VOIs on the image *after* the Fluorescence Recovery after Photobleaching (FRAP) dialog box appears using the instructions provided in the section on “Applying the FRAP algorithm” on page 453. VOIs created outside the dialog box may not have the right colors.

The VOIs must all have curves present in the first slice after photobleaching (there is no reason to put curves in any other slice). As it runs, the algorithm propagates the photobleached and whole organ VOIs to the other image slices. The background VOI is not propagated to other slices. The photobleached VOI should be contained within the whole organ VOI. The whole organ region has a greater average intensity than the photobleached region, and the optional background region has a smaller average intensity than the photobleached region.

The algorithm uses the average of the background VOI from the first slice after photobleaching to obtain the background constant, which is subtracted from the photobleached intensity and whole organ intensity measurements. It then obtains the ratio  $F_a/F_0$ , in which  $F_a$  is the first slice after photobleaching and  $F_0$  is the slice just before photobleaching.

Even with low illumination during the recovery phase, fluorescence loss through photobleaching is expected to occur over the course of the recovery curve. For this loss the time that matters is the exposure time for each slice picture and not the absolute time at which the exposure was made. Since the exposure time for each slice is the same, the time here is proportional to the number of slices. The background corrected photobleached values are divided by the background corrected whole organ values to correct for this

loss of fluorescence.

The algorithm then normalizes the values of the (corrected photobleached/corrected whole organ) by dividing by the value of this ratio just before photobleaching. These normalized photobleached ratios are fitted to a slightly modified version of equation 19 in the Carrero article:

$$R(t, \alpha, \beta, \Gamma) = \text{bottom} + \text{span} \times (1 - \Gamma e^{\alpha t} - (1 - \Gamma)e^{\beta t}) \quad \text{with } 0 > \alpha > \beta \text{ and } 1 \geq \Gamma \geq 0.$$

The mobile fraction equals bottom + span. In this curve fitting the time used is the time elapsed since photobleaching. Alpha, beta, gamma, and the ratio of the whole organ region fluorescence after bleaching to before bleaching are output. The dissociation rate, the association rate, and the diffusion transfer coefficient are obtained from alpha, beta, gamma, and the *afterBeforeRatio*.

For narrow band 2D, the effective diffusion coefficient in micrometers<sup>2</sup> per second is found from the diffusion transfer coefficient. The software measures the photobleached width *w* as the minimum dimension of the photobleached VOI bounding box and measures the whole organ length *l* along the same direction.

$$D_{eff} = \text{diffusion transfer coefficient} \times w \times \frac{(l-w)}{4.0}$$

For 1D pure diffusion fitting, the following equation applies:

EQUATION 1

$$I(t) = I(\text{final}) \left( 1 - \frac{w^2}{\left( \sqrt{w^2 + 4\pi t D_{eff}} \right)^2} \right)$$

where

*t* is time

*I* is the photobleached fluorescent intensity

*w* is the width of the bleaching along the membrane in micrometers (um)

$D_{eff}$  is the effective 1D diffusion to coefficient in micrometers<sup>2</sup> per second

Fitting data for the above equation solves for  $\frac{D_{eff}}{w^2}$ .

Then:

$$D_{eff} = \frac{D}{w^2} \times p \times p$$

You can select an optional check box on the FRAP dialog box to double check the Gauss-Newton nonlinear fitting results with a simple grid search pattern. The software calculates the Sum of Squares of Errors (*SSE*) at every legal point on a grid and then finds the sum of the squares of the fitted photobleached data minus the actual photobleached data for every slice starting with the first slice after photobleaching.

For the 2D narrow band and 2D circle photobleached models, the algorithm uses a 3D grid with 201 alpha points, 201 beta points, and 101 gamma points. It then varies both alpha and beta from 0.05 times the values found in the nonlinear fit to 20 times the values found in the nonlinear fit. It geometrically increments the values so that each value is about 1.03 times as great as the previous value and arithmetically increments gamma by 0.01 from 0 to 1.

The algorithm finds the global SSE minimum and, if present, any local SSE minimums, which are the lowest point in a 5 by 5 by 5 cube. The 3D space is searched with the restriction that beta <= alpha. Points with beta > alpha are not included in the 3D search and are simply filled in with SSE values equal to 1.1 times the SSE maximum found over the permissible space. The search is conducted with the bottom and span values kept fixed at the values found by the nonlinear fit. Since these values are very likely to have been accurately determined, this should not be a problem. In any event a search over a 5D space would be very time consuming. A 201 by 201 by 101 3D error image is created to display the calculated SSE values. The error image name is the source image name with *\_err* appended on the end.

Point VOIs appear at the locations of the global minimum and at local minimums if any exist. The algorithm stores point VOIs in a folder titled *defaultVOI\_sourceImageName\_err*. For the pure 1D fit and the single exponential fit, the search is simply one dimensional; therefore, the

software does not create an error image. For the pure 1D fit,  $201 \frac{D_{eff}}{w^2}$  values going from 0.05 times the nonlinear fit value to 20 times the nonlinear fit value are used to calculate SSE. For the single exponential fit,  $201 \frac{1}{2}$  half values going from 0.05 times the nonlinear fit value to 20 times the nonlinear fit value are used to calculate sse.

In the derivation for the case in which the photobleached area is a circle, there are only two compartments:

- 0, the inside of the circle
- 1, the outside of the circle

The equations are:

EQUATION 2

$$\frac{du0f}{dt} = -D1 \times u0f + D2 \times u1f - ka \times u0f + kd \times u0b$$

$$\frac{du1f}{dt} = D1 \times u0f - D2 \times u1f - ka \times u1f + kd \times u1b$$

$$\frac{du0b}{dt} = ka \times u0f - kd \times u0b$$

$$\frac{du1b}{dt} = ka \times u1f - kd \times u1b$$

$$D1 = \left( \frac{Fa}{(Fa + F0 - Fa)} \right) Dt = \left( \frac{Fa}{F0} \right) Dt$$

$$D2 = \left( \frac{(F0 - Fa)}{(Fa + F0 - Fa)} \right) Dt = \left( \frac{(F0 - Fa)}{F0} \right) Dt$$

As expected

$$D1 + D2 = Dt$$

$$u = (u0f, u1f, u0b, u1b)$$

$$u0 = \left( 0, \left( \frac{kd}{(ka + kd)} \right) \left( \frac{Fa}{F0} \right), 0, \left( \frac{ka}{(ka + kd)} \right) \left( \frac{Fa}{F0} \right) \right)$$

The eigenvalues are found by setting the determinant to zero (0):

EQUATION 3

$$\begin{bmatrix} -D1 - ka - e & D2 & kd & 0 \\ D1 & -D2 - ka - e & 0 & kd \\ ka & 0 & -kd - e & 0 \\ 0 & ka & 0 & -kd - e \end{bmatrix}$$

This gives the equation:

$$e^4 + (D1 + D2 + 2ka + 2kd) \times e^2 + (D1 \times ka + 2D1 \times kd + D2 \times ka + 2D2 \times kd + ka^2 + 2ka \times kd + kd^2 \times e^2 + D1 \times ka \times kd + D1 \times kd^2 + D2 \times ka \times kd + D2 \times kd^2) \times e = 0$$

Using  $D1 + D2 = D$ , the above equation factors as:

$$e \times (e + ka + kd) \times (e^2 + (D + ka + kd) \times e + D \times kd) = 0$$

The quadratic gives the roots alpha and beta with

$$\alpha = -s_1 + s_2$$

$$\beta = -s_1 - s_2$$

where

$$s_1 = \frac{(D + ka + kd)}{2}$$

$$s_2 = \frac{\sqrt{((D + ka + kd)^2 - 4D \times kd)}}{2}$$

The eigenvalue  $e = 0$  gives the eigenvector:

$$1$$

$$\left(\frac{D1}{D2}\right)$$

$$\left(\frac{ka}{kd}\right)$$

$$\left(\frac{(D1 \times ka)}{(D2 \times kd)}\right)$$

The eigenvalue  $e = -ka - kd$  gives the eigenvector:

$$1$$

$$\left(\frac{D1}{D2}\right)$$

$$-1$$

$$-\left(\frac{D1}{D2}\right)$$

The eigenvalue  $e = \alpha$  gives the eigenvector:

The eigenvalue  $e = \beta$  gives the eigenvector:

1

$$\frac{D1 \times (kd + \alpha)}{(D2 \times kd + \alpha \times kd + \alpha \times D2 + \alpha \times ka + \alpha^2)}$$

$$\frac{ka}{(kd + \alpha)}$$

$$\frac{D1 \times ka}{(D2 \times kd + \alpha \times kd + \alpha \times D2 + \alpha \times ka + \alpha^2)}$$

$$\frac{D1 \times ka}{(D2 \times kd + \beta \times kd + \beta \times D2 + \beta \times ka + \beta^2)}$$

Let the eigenvectors be  $v_1, v_2, v_3,$  and  $v_4$ . Let the matrix  $V$  be given by:

$$\begin{bmatrix} v_{11} & v_{21} & v_{31} & v_{41} \\ v_{12} & v_{22} & v_{32} & v_{42} \\ v_{13} & v_{23} & v_{33} & v_{43} \\ v_{14} & v_{24} & v_{34} & v_{44} \end{bmatrix}$$

Let the vector  $C$  be

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

Remembering that  $u_0$  is the vector of initial conditions, we wish to solve:

$$VC = u_0$$

Adding equations 1 and 3:

$$c1 + c2 + c3 + c4 = 0(1)$$

$$\left(\frac{ka}{kd}\right) \times c1 - c2 + \left(\frac{ka}{(ka + kd + \alpha)}\right) \times c3 + \left(\frac{ka}{(kd + \beta)}\right) \times c4 = 0(3)$$

yields equation 5:

EQUATION 4

Adding equations 2 and 4:

$$\left(\frac{(ka + kd)}{kd}\right) \times c1 + \left(\frac{(ka + kd + \alpha)}{(kd + \alpha)}\right) \times c3$$

$$\left(\frac{(ka + kd + \beta)}{(kd + \beta)}\right) \times c4 = 0(5)$$

EQUATION 5

yields equation 6:

EQUATION 6

Multiplying 5 by  $\frac{-D1}{D2}$  and adding to 6 yields:

$$\left(\frac{D1 \times (ka + kd)}{D2 \times kd}\right) \times c1$$

$$+ \left(\frac{D1 \times (ka + kd + \alpha)}{D2 \times kd + \alpha \times kd + \alpha \times D2 + \alpha \times ka + \alpha^2}\right) \times c3$$

$$+ \left(\frac{D1 \times (ka + kd + \beta)}{D2 \times kd + \beta \times kd + \beta \times D2 + \beta \times ka + \beta^2}\right) \times c4 = \frac{Fa}{F0}(6)$$

EQUATION 7



$$\begin{aligned} & \frac{D1}{D2} \times c1 + \frac{D1}{D2} \times c2 \\ & + \left( \frac{D1 \times (kd + \alpha)}{(D2 \times kd + \alpha \times kd + \alpha \times D2 + \alpha \times ka + \alpha^2)} \right) \times c3 \\ & + \left( \frac{D1 \times (kd + \beta)}{(D2 \times kd + \beta \times kd + \beta \times kd + \beta \times D2 + \beta \times ka + \beta^2)} \right) \times c4 \\ & = \left( \frac{kd}{(ka + kd)} \right) \left( \frac{Fa}{F0} \right) (2) \end{aligned}$$

$$\begin{aligned} & \left( \frac{D1 \times ka}{(D2 \times kd)} \right) \times c1 + \left( \frac{-D1}{D2} \right) \times c2 \\ & + \left( \frac{D1 \times ka}{(D2 \times kd \times \alpha \times kd + \alpha \times D2 + \alpha \times ka + \alpha^2)} \right) \times c3 \\ & + \left( \frac{D1 \times ka}{(D2 \times kd + \beta \times kd + \beta \times D2 + \beta \times ka + \beta^2)} \right) \times c4 \\ & = \left( \frac{ka}{(ka + kd)} \right) \left( \frac{Fa}{F0} \right) (4) \end{aligned}$$

which simplifies to:

EQUATION 8

Multiplying equation 1 by  $\frac{-D1}{D2}$  and adding to equation 2 yields:

EQUATION 9

$$\left[ \frac{D1 \times (ka + kd + \alpha)}{(D2 \times kd + \alpha \times kd + \alpha \times D2 + \alpha \times ka + \alpha^2)} - \left( \frac{D1 \times (ka + kd + \alpha)}{D2 \times (kd + \alpha)} \right) \right] \times c3$$

$$+ \left[ \frac{D1 \times (ka + kd + \beta)}{(D2 \times kd + \beta \times kd + \beta \times D2 + \beta \times ka + \beta^2)} - \frac{D1 \times (ka + kd + \beta)}{(D2 \times (kd + \beta))} \right] \times c4 = \frac{Fa}{F0}$$

$$\left[ \frac{-D1 \times \alpha \times (ka + kd + \alpha)^2}{(D2 \times (kd + \alpha) \times (D2 \times kd + \alpha \times kd + \alpha \times D2 + \alpha \times ka + \alpha^2))} \right] \times c3$$

$$+ \left[ \frac{-D1 \times \beta \times (ka + kd + \beta)^2}{(D2 \times (kd + \beta) \times (D2 \times kd + \beta \times kd + \beta \times D2 + \beta \times ka + \beta^2))} \right] \times c4 = \frac{Fa}{F0} (7)$$

Multiplying equation 8 by  $\frac{-(ka + kd + \alpha)}{(kd + \alpha)}$  and adding to equation 7 yields:

$$\left[ \frac{-\alpha \times D1 \times (ka + kd + \alpha)}{(D2 \times (D2 \times kd + \alpha \times kd + \alpha \times D2 + \alpha \times ka + \alpha^2))} \right] \times c3$$

$$+ \left[ \frac{-\beta \times D1 \times (ka + kd + \beta)}{(D2 \times (D2 \times kd + \beta \times kd + \beta \times D2 + \beta \times ka + \beta^2))} \right] \times c4 = \left( \frac{kd}{(ka + kd)} \right) \times \frac{Fa}{F0}$$

EQUATION 10

$$c4 = \alpha \times \left( \frac{Fa}{F0} \right) \times D2 \times (D2 \times kd + \beta + kd + \beta \times D2 + \beta \times ka + \beta^2)$$

$$\times \frac{(kd + \beta)}{[(ka + kd) \times \beta \times (\beta - \alpha) \times D1 + (ka + kd + \beta)]}$$

This simplifies to:

EQUATION 11

$$c4 = \frac{\left(\frac{Fa}{F0}\right) \times D2 \times kd + (-D + ka + kd - 2S2)}{[D \times (ka + kd) \times 4S2]}$$

Substituting c4 back into equation 8 yields c3:

EQUATION 12

$$c3 = \frac{\beta \times \left(\frac{Fa}{F0}\right) \times D2 \times (kd + \alpha) \times (D2 \times kd + \alpha \times kd + \alpha \times D2 + \alpha \times ka + \alpha^2)}{[\alpha \times (\alpha - \beta) \times D1 \times (ka + kd) \times (ka + kd + \alpha)]}$$

This simplifies to:

EQUATION 13

$$c3 = \frac{\left(\frac{Fa}{F0}\right) \times D2 \times kd \times (D - ka - kd - 2S2)}{[D \times (ka + kd) \times 4S2]}$$

Putting c3 and c4 in equation 5 and solving for c1 yields:

EQUATION 14

$$c1 = \frac{\left(\frac{Fa}{F0}\right) \times D2 \times kd}{[D \times (ka + kd)]}$$

Putting c1, c3, and c4 into equation 4 and solving for c2 yields c2 = 0.

Thus, the solution is:

EQUATION 15

$$\begin{aligned} u0f &= c1 + c3e^{(\alpha \times t)} + c4e^{(\beta \times t)} \\ &= \left[ \left(\frac{Fa}{F0}\right) \times \frac{D2 \times kd}{(D \times (ka + kd))} \right] \times \left[ \frac{1 + D - ka - kd - 2S2}{4S2e^{\alpha \times t}} + \frac{(-D + ka + kd - 2S2)}{4S2e^{\beta \times t}} \right] \end{aligned}$$

The above equation is normalized by dividing by

EQUATION 16

$$\frac{\left(\frac{F_a}{F_0}\right) \times D \times kd}{(D \times (ka + kd))}$$

The coefficient of  $\exp(\alpha \times t)$  in the normalized equation equals  $-\gamma$

so  $\Gamma = \frac{(-D + ka + kd + 2S^2)}{4S^2}$  and the normalized equation can be written

as:

EQUATION 17

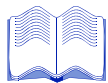
$$u_{of\ normalized} = 1 - \Gamma e^{(\alpha \times t)} - (1 - \Gamma)e^{(\beta \times t)}$$

## IMAGE TYPES

You can apply this algorithm to both color and black-and-white 3D images. However, with color images, the algorithm uses only one of the colors.

## SPECIAL NOTES

This algorithm can only be used when the fluorescence is homogeneous in a compartment. If the fluorescence has spatial gradients across a compartment, a model using partial derivatives with respect to distance should be employed.



## REFERENCES

- Axelrod, D., D. E. Koppel, J. Schlessinger, E. Elson, and W. W. Webb. "Mobility Measurement by Analysis of Fluorescence Photobleaching Recovery Kinetics." *Biophysical Journal* 16 (1976):1055–1069.
- Carrero, Gustavo, Darin McDonald, Ellen Crawford, Gerda de Vries, and Michael J. Hendzel. "Using FRAP and Mathematical Modeling to Determine the In Vivo Kinetics of Nuclear Proteins." *Methods* 29 (2003):14–28.
- Lippincott-Schwartz, J., J. F. Presley, K. J. M. Zaal, K. Hirschberg, C. D. Miller, and J. Ellenberg. "Monitoring the Dynamics and Mobility of Membrane Proteins Tagged with Green Fluorescent Protein." *Methods in Cell Biology* 58 (1999):261–281.

Phair, Robert D. "Practical Kinetic Modeling of Large Scale Biological Systems." <http://www.bioinformaticservices.com/bis/resources/cybertext/IBcont.html>.

## Applying the FRAP algorithm

This section explains how to run the FRAP algorithm and how to modify the appearance of the resulting graph windows.

### To run the FRAP algorithm

- 1** Open a 3D color or 3D black-and-white image.
- 2** Select Algorithms > Microscopy > FRAP. The Fluorescence Recovery after Photobleaching dialog box (Figure 3) opens.

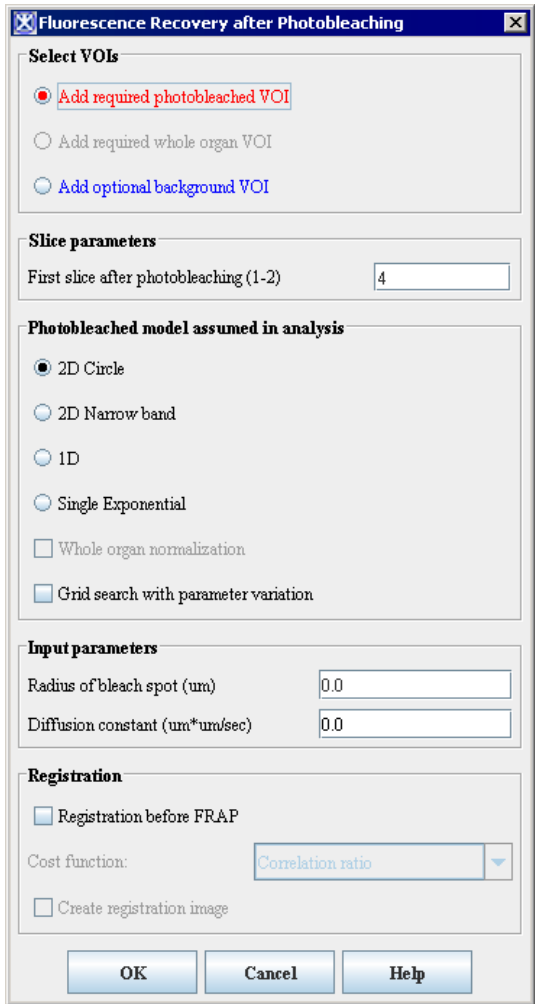
<p><b>Add required photobleached VOI</b></p>	<p>Applies a mandatory photobleached VOI, which appears in red, to the first image slice after photobleaching. Make sure that the VOI delineates the photobleached area. By default, this radio button is selected.</p>	
<p><b>Add required whole organ VOI</b></p>	<p>Adds a mandatory whole organ VOI, which appears in green, to the first image slice after photobleaching. This VOI should contain the entire photobleached VOI and should be brighter than the photobleached VOI.</p>	
<p><b>All optional background VOI</b></p>	<p>Adds an optional background VOI, which appears in blue, to the first image after photobleaching. This VOI should delineate a featureless area that is darker than the photobleached area.</p>	
<p><b>Color selection (only appears for color images)</b></p>	<p>Select one of the available colors—red, green, or blue—listed that corresponds to the fluorescence undergoing photobleaching. By default, red is selected. However, if red is single valued, the default is green.</p>	
<p><b>First slice after photobleaching</b></p>	<p>Type the slice number of the first slice after photobleaching. If the image's file header contains this information, the slice number appears. Otherwise, the default is 4.</p>	
<p><b>2D narrow band</b></p>	<p>Assumes a narrow photobleached band surrounded by a large unbleached region on each side. Unless a photobleached region circle is found in the file header, the default value is 2D narrow band.</p>	
<p><b>2D circle</b></p>	<p>Assumes a small circle in the midst of a large unbleached region. If the file header for the image contains information that the photobleached region is a circle, the default is 2D circle.</p>	
<p><b>1D</b></p>	<p>Assumes pure 1D diffusion.</p>	
<p><b>Single exponential</b></p>	<p>Assumes recovery according to a single exponential.</p>	

Figure 1. Fluorescence Recovery after Photobleaching dialog box

<b>Grid search with parameter variation</b>	Conducts a double check by grid search.
<b>Registration before FRAP</b>	Registers the image slices that are not aligned. Select this item so that slice alignment occurs before FRAP. The default is no registration.
<b>Cost function</b>	Specifies the registration cost function. You may select correlation ratio (the default), least squares, normalized cross correlation, or normalized mutual information. To use this item, you must first select Registration before FRAP.
<b>Create registration image</b>	Creates the registered image in a separate image window. To use this item, you must first select Registration before FRAP. By default, this item is selected.
<b>OK</b>	Applies the algorithm according to the specifications made in the dialog box.
<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes this dialog box.
<b>Help</b>	Displays online help for this dialog box.

Figure 1. Fluorescence Recovery after Photobleaching dialog box (continued)

- 3 Select for color images the fluorescence color in the Color selection panel.







**Note:** Only the color that you select has a role in the FRAP mathematics.

- 4 Read to see, if an information panel is present, if either the shape of the photobleached region or the first slice after photobleaching is specified.
- 5 Go to the MIPAV window and adjust the image slice slider to display the first slice after photobleaching.
- 6 Type the number of the first slice after photobleaching in the text box in the Slice parameters panel.



**Remember:** The slice number appears on the title bar of the image window. As you move the image slice slider in the MIPAV window to the right, the slice number increases. As you move it to the left, the slice number decreases.

- 7 Select Add required photobleached VOI in the Select VOIs panel in the FRAP dialog box.
- 8 Go to the MIPAV window, and select one of these icons from the VOI toolbar:




- Draw rectangle VOI 
- Draw ellipse VOI 
- Draw polygon/polyline VOI 
- Draw levelset VOI 

**9** Draw the VOI on the image. The VOI appears in red.



**Note:** In drawing these VOIs, you do not need to first select the New VOI icon.

**10** Select Add required whole organ VOI in the Select VOIs panel in the FRAP dialog box.

**11** Create a whole organ VOI by selecting , , , or  to draw a rectangle, ellipsoidal, polygon/polyline, or levelset VOI on the image around the whole organ region in the first slice after the photobleaching. A green VOI appears on the image.



**Figure 2.** An image with a required photobleached VOI in red and a whole organ VOI in green









**Whole organ region:** The whole organ region should entirely contain the photobleached region and be brighter than the photobleached region.

**12** Go to the next step, or, if you wish, create an optional background VOI using the following directions:

**a** Select Add optional background VOI in the Select VOIs panel in the FRAP dialog box.

**b** Go to the MIPAV window and select , , , or  to draw a rectangle, ellipsoidal, polygon/polyline, or levelset VOI on the image around the background region in the first slice after the photobleaching. A blue VOI appears on the image.



**Background region:** The background region should not contain any structures and should be darker than the photobleached region.

**13** Register the image slices as an option by doing the following:

**a** Select Registration before FRAP.

**b** Select a cost function in the Cost function list. Although correlation ratio is the default cost function, you can also select least squares, normalized cross correlation, or normalized mutual information.

**c** Select or clear Create registration image. Since registration is far more time consuming than calculating FRAP, selection is the default.

**14** Select the photobleached model assumed in analysis:

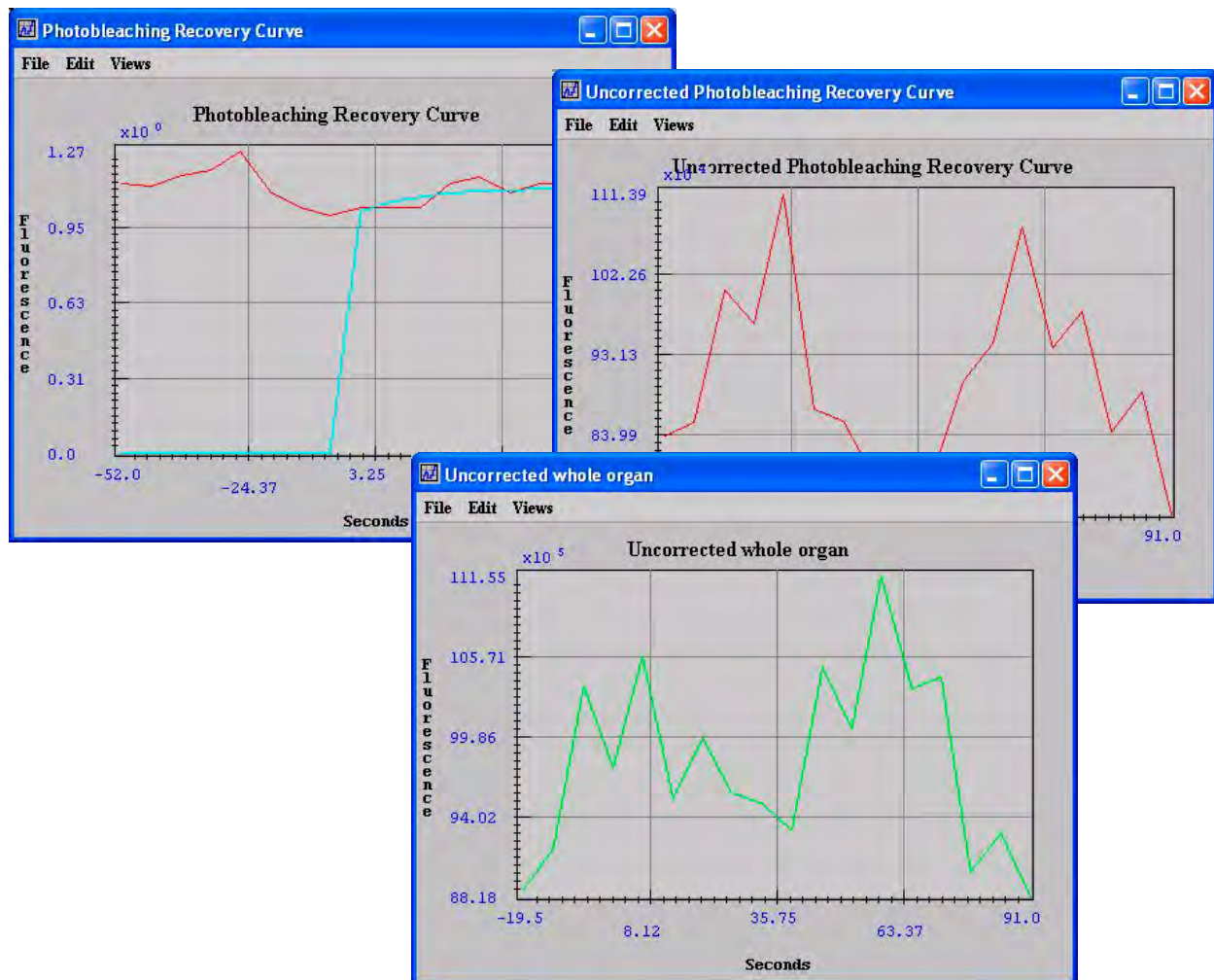
- 2D narrow band
- 2D circle
- 1D
- Single exponential

**15** Select Grid search with parameter variation if you want to double check the nonlinear fitting engine with a grid search.

**16** Click OK.

The algorithm begins to run and a progress bar appears with the status. When the algorithm finishes running, the progress bar disappears, and the following windows appear:

- Photobleaching Recovery Curve window (Figure 3), which displays a graph of the fitting of the normalized photobleached region recovery curve (the calculated parameters appear on the Data page of the Output window)



**File**      **Open Graph (CtrlO)**—Allows you to open a previously saved graph. When you select this command, the Open Graph Data dialog box appears.

**Figure 3. Graph windows: Photobleaching Recovery Curve, Uncorrected Photobleaching Recovery Curve, and Uncorrected Whole Organ windows**

	<b>Save Graph (CtrlS)</b> —Allows you to save a graph as a .PLT file. When you select this command, the Save dialog box appears.
	<b>Print Graph (CtrlP)</b> —Allows you to print the graph. When you select this command, the Print dialog box appears.
	<b>Close Graph (CtrlX)</b> —Closes the graph immediately.
<b>Edit</b>	<b>Delete Function</b> —Allows you to delete a specified function on the graph unless the graph contains only one function.
	<b>Copy Function</b> —Allows you to copy the function that you specify.
	<b>Paste Function</b> —Allows you to paste a previously copied function to the graph.
<b>Views</b>	<b>Modify Graph Features</b> —Allows you to change all of the graph features, such as the names and colors of the functions, the background color of the graph, the ranges used, whether the gridlines appear, and so on. When you select this command, the Modify Graph window opens.
	<b>Reset Range to Default</b> —Changes the range displayed in the graph to the default range.
	<b>Reset Graph to Original</b> —Changes the range displayed in the graph to the original range.
<b>Graph</b>	Displays the amount of fluorescence over time (seconds).

**Figure 3. Graph windows: Photobleaching Recovery Curve, Uncorrected Photobleaching Recovery Curve, and Uncorrected Whole Organ windows (continued)**

- Uncorrected Photobleaching Recovery Curve window (Figure 3)
- Uncorrected Whole Organ window (Figure 3)

If you selected Create registration image in the FRAP dialog box, an image window with a registered source image appears.



**Note:** All of the three graph windows—the Photobleaching Recovery Curve, Uncorrected Photobleaching Recovery Curve, and Uncorrected Whole Organ windows—have the same menus and commands that work in the same way.

If you selected Grid search with parameter variation in the FRAP dialog box and either 2D narrow band or 2D circle, a 201 by 201 by 101 3D error image is created to display the sum of squared error values.

## To print graphs

In a graph window:

- 1 Select File > Print, or press Ctrl P on the keyboard. A Print dialog box (Figure 4) appears.

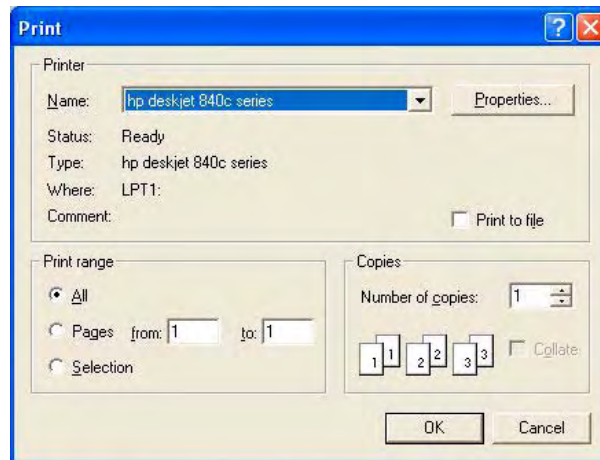


Figure 4. Print dialog box

- 2 Select the printer, print range, and number of copies.
- 3 Click OK.

## To save graphs

- 1 Select File > Save, or press Ctrl S on the keyboard. The Save dialog box appears.
- 2 Select a directory in which to store the graph.
- 3 Type a name for the graph, including the .PLT extension, in the File Name box.
- 4 Click OK.

## To open previously saved graphs

- 1 Select File > Open, or press Ctrl O on the keyboard. The Open dialog box appears.
- 2 Navigate to the directory where the graph is stored.

**3** Select the name of the graph.

**4** Click OK.

### To close graphs

Select File > Close, or press Ctrl X on the keyboard. The graph window closes.

### To change the appearance of the graph window

Using the Modify Graph Features command on the Views menu of each of the graph windows, you can change how the data appears in each graph. When you select Views > Modify Graph Features, the Modify Graph dialog box opens.



**Figure 5. Modify Graph Features command on the Views menu in a graph window**

The Modify Graph dialog box (Figure 6) allows you to change the following:

- The appearance of the graph
- Whether legends for functions appear on the graph
- The color and visibility of functions and whether points appear on the functions
- Mathematical adjustments to the functions

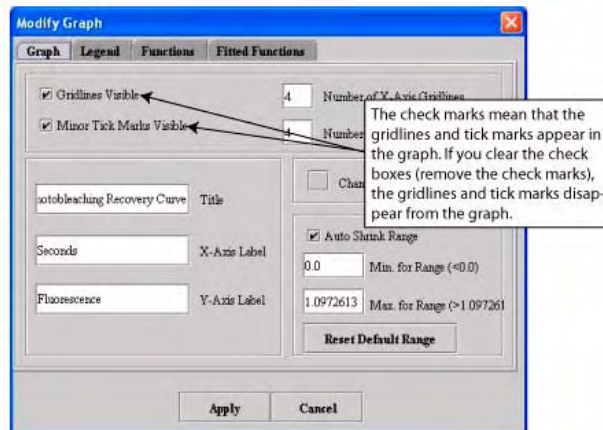


Figure 6. Modify Graph dialog box showing the Graph page

## To change the displayed titles of graphs

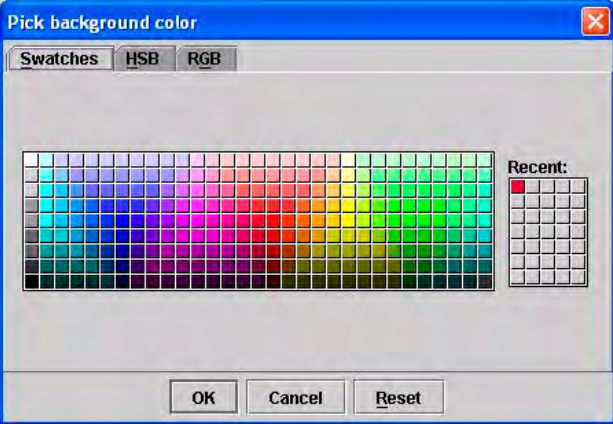
Changing the title of graphs is very easy.

- 1 Select Views > Modify Graph Features in the graph window. The Modify Graph Features dialog box opens.
- 2 Type the new title in Title. You can simply type over the current title of the graph.
- 3 Click Apply.

## To change the background color of graphs

- 1 Select Views > Modify Graph Features in the graph window. The Modify Graph dialog box opens.
- 2 Click Change background color on the Graphs page. The Pick Background Color dialog box (Figure 7) opens.
- 3 Select a color on the one of the pages in the dialog box: Swatches, HSB, or RGB page.
- 4 Click OK. The Pick Background Color dialog box closes.

**5 Click Apply. The Modify Graph dialog box closes. The background of the graph window should now be the color you selected.**

<p><b>Change background color</b></p>	<p>Allows you to change the background color of the grid. By default, the background color is light gray. When you click this button, the Pick Background Color dialog box opens.</p> 
<p><b>Auto shrink range</b></p>	<p>Allows you to adjust the display of the Y axis of the graph. By default, this check box is selected.</p>
<p><b>Minimum for range</b></p>	<p>Specifies the minimum value for the Y axis, which must be 0.0 or less. The default value is 0.0.</p>
<p><b>Maximum for range</b></p>	<p>Specifies the maximum value for the X axis, which must be 1.0 or more. The default value is 1.0.</p>
<p><b>Reset default range</b></p>	<p>Reinstates the default range for the graph, which means that the value of Minimum for range returns to 0.0 and the value for Maximum for range returns to 1.0.</p>
<p><b>Apply</b></p>	<p>Applies immediately the changes to the graph according to the specifications in this dialog box but keeps this dialog box open.</p>
<p><b>Cancel</b></p>	<p>Disregards any changes that you made in the dialog box and closes this dialog box.</p>
<p><b>Help</b></p>	<p>Displays online help for this dialog box.</p>

**Figure 7. Pick Background Color dialog box**

**To return the graph window to its default settings**

If you made changes in the way a graph appears in the Modify Graph dialog box and now would like to return to the default settings, select in the graph window View > Reset Graph to Original or press Ctrl Z on the keyboard.

## To return the range displayed in the graph window to its default settings

The Modify Graph dialog box allows to change the range that is displayed in a graph window. If, after modifying the range, you want to reinstate the minimum and maximum values in the range for a graph, do either of the following:

*In the graph window*

Select Views > Reset Range to Default.

*In the Modify Graph dialog box*

- 1** Select Views > Modify Graph Features in the graph window. The Modify Graph dialog box opens.
- 2** Select Reset default range.
- 3** Click Apply.

The range returns to its default settings.

## To display legends for functions

- 1** Select Views > Modify Graph Features in the graph window. The Modify Graph dialog box opens.
- 2** Click Legend. The Legend page (Figure 8) appears.
- 3** Select Show legend.
- 4** Accept or change the names of the functions in the text boxes.
- 5** Click Apply. The legends should appear in the graph window.



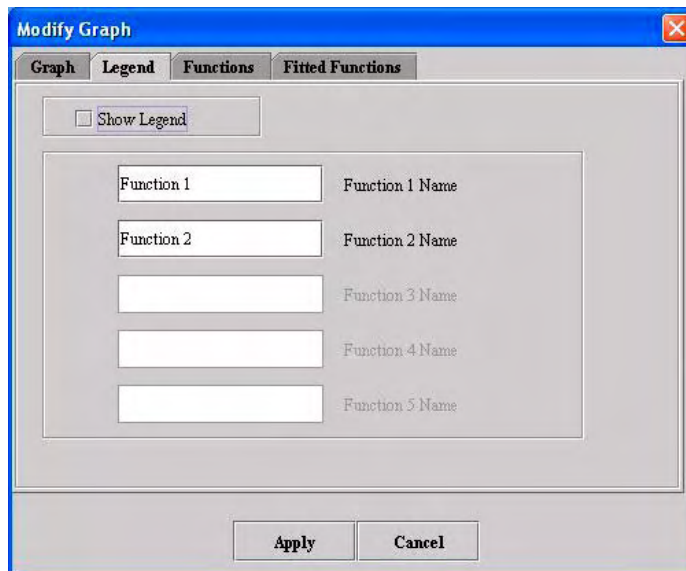


Figure 8. Legend page in the Modify Graph dialog box

### To change the appearance of functions

- 1 Select Views > Modify Graph Features in the graph window. The Modify Graph dialog box appears.
- 2 Click Functions. The Functions page (Figure 9) appears.

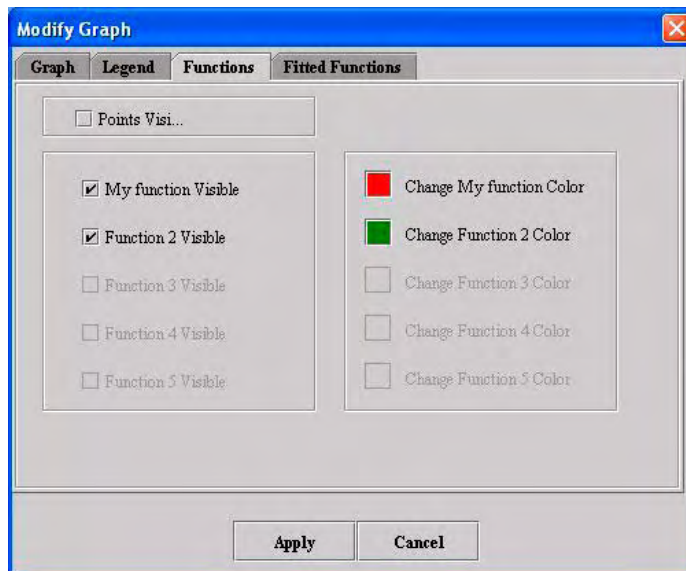


Figure 9. Functions page in the Modify Graph dialog box

- 3** Select Points visible if you want to display all of the points on the functions.
- 4** Change the color of each function by clicking Change function color.
- 5** Click Apply. In the graph window, the appearance of the functions should display according to your specifications.

### To apply mathematical changes to the functions

- 1** Select Views > Modify Graph Features in the graph window. The Modify Graph dialog box appears.
- 2** Click Fitted Functions. The Fitted Functions page (Figure 10) appears.
- 3** Select Fit linear or Fit exponential. The default choice is None.
- 4** Select the functions to which these changes should be made.
- 5** Click Apply. The functions should change according to your specifications.

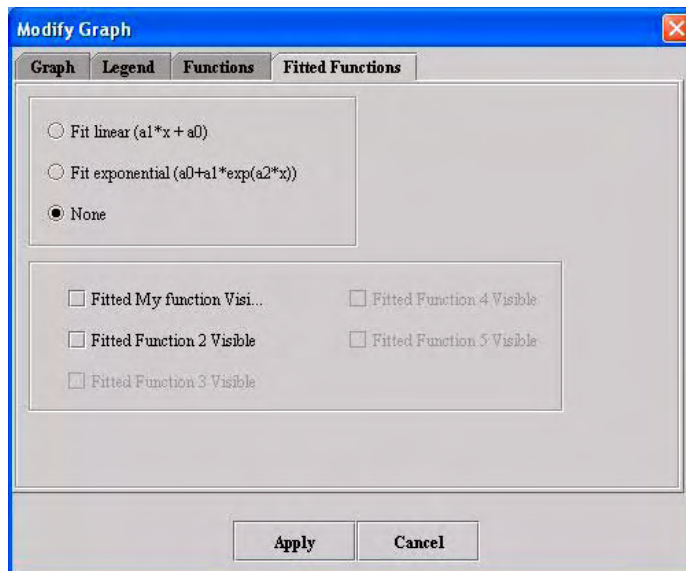


Figure 10. Fitted Functions page in the Modify Graph window

## Microscopy: Colocalization Orthogonal Regression (FRET)—Acceptor Photobleaching

This algorithm uses acceptor photobleaching to compare the proximity of fluorescent-labeled molecules in two 2D images over distance:

- An image *before* acceptor photobleaching, also referred to as the prebleached image
- An image *after* acceptor photobleaching, which is the postbleached image

### Background

FRET refers to the nonradiative transfer of energy from an excited state donor fluorescent molecule to a nearby acceptor fluorescent molecule. The energy transfer efficiency  $E$ , defined as the number of energy transfer events divided by the number of photons absorbed by the donor, is related to the distance  $R$  between the acceptor and donor by:

$$E = \frac{1}{\left[1 + \left(\frac{r}{R_0}\right)^6\right]}$$

EQUATION 1

where  $R_0$ , the Forster critical distance, is the distance at which  $E = 0.5$ .

We also have

$$E = 1 - \frac{FDA}{FD}$$

EQUATION 2

where  $FDA$  is the donor fluorescence in the presence of an acceptor and  $FD$  is the donor fluorescence in the absence of the acceptor.

The equivalent of an acceptor's absence can be created by photobleaching the acceptor. Thus, a method using equation 2 can be performed on a single sample by measuring the donor fluorescence before and after photobleaching the acceptor molecules.

Because FRET falls off as the sixth power of the distance between the donor and the acceptor, no FRET occurs for distances greater than  $2RO$ . Since  $RO$  is on the order of 10 to 70 Angstroms, by performing FRET measurements it is possible to distinguish proteins that are merely nearby in the same compartment from those proteins that are interacting with each other.

To access the FRET algorithm, you first open the two images in question and then select Algorithm > Microscopy > FRET in the MIPAV window.

Since unbleached acceptors quench the donor fluorescence with FRET and bleached acceptors do not quench donor fluorescence, the postbleached image donor fluorescence is greater than the prebleached image donor fluorescence.

In the Fluorescence Resonance Energy Transfer dialog box, you may choose as an option to register the two images before running the FRET algorithm. During registration MIPAV registers the pre bleached image to the post bleached image and uses correlation ratio as the default cost function. However, other cost functions available for use include least squares, normalized cross correlation, or normalized mutual information. After registration MIPAV runs FRET on the *registered* pre bleached image rather than on the original pre bleached image.

To run the algorithm, you must first delineate a VOI on the post bleached image. This requires that you select Add required donor fluorescence VOI in the dialog box and then return to the MIPAV window to choose an ellipse VOI, rectangle VOI, polyline VOI, or levelset VOI with which to draw the VOI. You can then, as an option, also identify a background VOI on the image by selecting Add optional background VOI and repeat the steps performed in drawing the required donor VOI. The VOIs must all be placed in the post bleached image, and the background region has a smaller average intensity than the donor region.

As an option, you can use both a signal normalization VOI and a background VOI. However, to use a signal normalization VOI, you must have a background VOI.

For color images, select the color corresponding to the donor fluorescence. For example, if the donor fluorescence is blue, select blue. If the donor fluorescence is red, select red.



**Note:** Only one color is used from a color image.

If a background VOI is present and no signal VOI is present, the average of the prebleached background is subtracted from the average of the prebleached donor region. The average of the postbleached background is subtracted from the average of the postbleached donor region. Then, the energy transfer efficiency is calculated as (background subtracted postbleached donor intensity - background subtracted prebleached donor intensity)/background subtracted postbleached donor intensity.

Let  $b$  and  $s$  be the prebleached background and signal values, and let  $b_2$  and  $s_2$  be the postbleached background and signal values. Then, the following equation is used to linearly scale a donor value from the prebleached image into a donor value in the postbleached image range:

$$L = \left( \frac{s_2 - b_2}{s_1 - b_1} \right) \times P + \frac{(b_2 \times s_1 - b_1 \times s_2)}{(s_1 - b_1)}$$

where

L = Linearly scaled prebleached donor

P = Prebleached donor

Then

$$\text{Efficiency} = \frac{(\text{Postbleached donor} - \text{Linearly scaled prebleached donor})}{\text{Postbleached donor}}$$

---

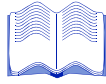
## IMAGE TYPES

You can apply this algorithm to two 2D images or one 2-slice 3D image.

---

## NOTES

None.



## REFERENCES

Refer to the following references for more information about the FRET algorithm.

Gordon, Gerald W., Gail Berry, Xiao Huan Liang, Beth Levine, and Brian Herman. "Quantitative Fluorescence Resonance Energy Transfer Measurements Using Fluorescence Microscopy." *Biophysical Journal* 74(May 1998):2702–2713.

Kenworthy, Anne K. "Imaging Protein-Protein Interactions Using Fluorescence Energy Transfer Microscopy." *Methods* 24(2001):289–296.

## Applying the FRET algorithm

To use this algorithm, do the following:

- 1 Open two 2D images or one 2-slice 3D image that contain fluorescent-labeled components. The Load Bleached ROI message (Figure 1) appears.

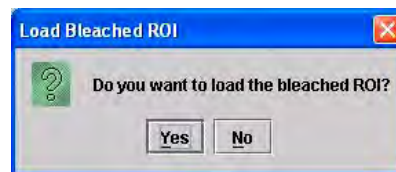




Figure 1. Load Bleached ROI message

- 2 Click Yes. MIPAV loads the images along with the bleached region of interest (ROI).
- 3 Select the prebleached image (refer to “Selecting Image Slices Quickly” below).
- 4 Select Algorithms > Microscopy > FRET in the MIPAV window. The FRET dialog box (Figure 2) appears.

## Selecting Image Slices Quickly

If the images or images slices are in the same file, you can quickly jump from one image or slice to another by using , the Decrements image slice icon, or , the Increments image slice icon.

- 5 Select the postbleached image.
- 6 Select—for color images—the fluorescence color in the Color selection panel.



**Note:** The Color selection panel does not appear in the dialog box if the images are grayscale or black and white. Also, the number of colors that you can select in the Color selection panel depends on the number of colors in the images. If, for example, only red and green appear in the images, then those are the only colors listed. If three colors are present, then red, green, and blue appear.



<b>Prebleached image</b>	Specifies the image <i>before</i> acceptor photobleaching.
<b>Postbleached image</b>	An image <i>after</i> acceptor photobleaching.
<b>Add required donor fluorescence VOI</b>	Adds a mandatory VOI to the postbleached image.
<b>Add optional background VOI</b>	Adds an optional background VOI on the postbleached image.
<b>Add optional signal normalization VOI</b>	Requires that a background VOI be used. Adds an optional signal normalization VOI on the postbleached image.
<b>Color selection (only appears for color images)</b>	Select one of the available colors—red, green, or blue—that corresponds to the fluorescence-labeled component being quantified in the images. By default, red is selected. Note that only colors that appear in the images are listed in this panel.
<b>Registration before FRET'</b>	Registers the prebleached image to the postbleached image. The default is no registration. Selecting this check box enables the Cost function list.
<b>Cost function</b>	Specifies the registration cost function. You may select correlation ratio (the default), least squares, normalized cross correlation, or normalized mutual information. To use this item, you must first select Registration before FRET.
<b>Create registration image</b>	Creates the registered image in a separate image window. To use this check box, you must first select Registration before FRET. By default, this check box is selected when registration is selected.
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes this dialog box.
<b>Help</b>	Displays online help for this dialog box.

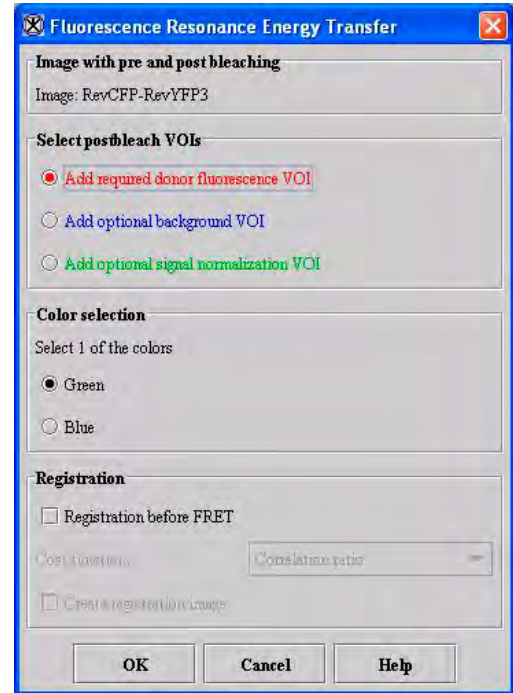






Figure 2. Fluorescence Resonance Energy Transfer (FRET) dialog box

## 7 Select Add required donor fluorescence VOI.

**8** Go to the MIPAV window, and select one of these icons from the VOI toolbar:

- Draw rectangle VOI 
- Draw ellipse VOI 
- Draw polygon/polyline VOI 
- Draw levelset VOI 

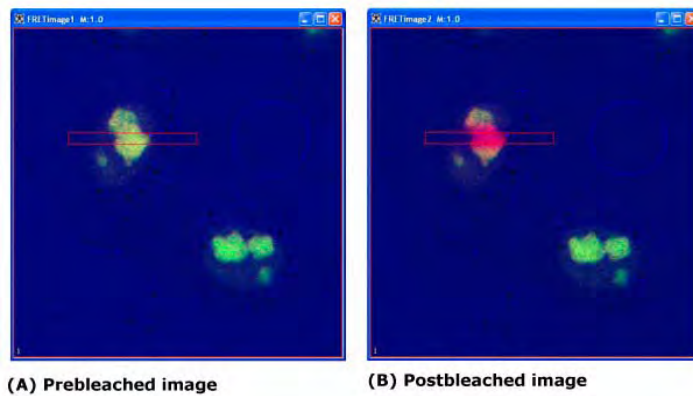






Figure 3. Example of (A) prebleached image and (B) postbleached image after FRET algorithm was applied

**9** Draw the VOI on the postbleached image. The VOI appears in red.



**Note:** In drawing these VOIs, you do not need to first select the New VOI icon.

**10** Go to step 12, or, if you wish, create an optional background VOI using the following directions:





- a** Select Add optional background VOI in the Select postbleach VOIs panel in the dialog box.
- b** Go to the MIPAV window and select , , , or  to draw a rectangle, ellipsoidal, polygon/polyline, or levelset VOI on the image. A blue VOI appears on the postbleached image.



**Background region:** The background region should not contain any structures and should be darker than the donor region.

**11** Go to the next step, or, if you wish, create an optional signal normalization VOI.

**a** Select Add optional signal normalization VOI in the Select postbleach VOIs panel in the dialog box.

**b** Go to the MIPAV window and select , , , or  to draw a rectangle, ellipsoidal, polygon/polyline, or levelset VOI on the image. A green VOI appears on the postbleached image.



**Note:** To use an optional signal normalization VOI, you must create a background VOI.

**12** Register the images as an option by doing the following:

**a** Select Registration before FRET.

**b** Select a cost function in the Cost function list. The default cost function is correlation ratio, but you can select least squares, normalized cross correlation, or normalized mutual information.

**c** Select or clear Create registration image. Since registration is far more time consuming than calculating FRET, if you selected the Registration before FRET check box, the Create registration image check box is selected by default.

**13** Click OK. The algorithm begins to run, and a progress bar appears momentarily with the status.

When the algorithm finishes running, the progress bar disappears. The data appears on the Data page in the Output window (Figure 4).

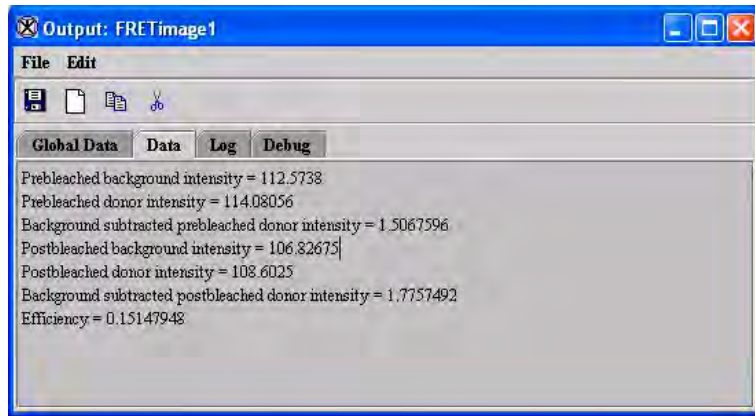


Figure 4. FRET data in the Data page of the Output window



**Note:** If you chose to register the images, the registered image appears in a separate image window, and the FRET data appears on the Data page in the Output window.

## Microscopy: Fluorescent Resonance Energy Transfer (FRET) Bleed Through and Efficiency

This section provides information on and discusses how to use the following two FRET algorithms:

- FRET Bleed Through algorithm
- FRET Efficiency algorithm

Used consecutively, the FRET Bleed Through algorithm uses two sets of three 2D images and the FRET Efficiency algorithm uses one set of three 2D images to measure effects dependent on the proximity of fluorescent-labeled molecules.

**Note:** You must first run the FRET Bleed Through algorithm twice: once on acceptor-dyed images and once on donor-dyed images. Using the results achieved from running this algorithm, you then use the FRET Efficiency algorithm to process images that were dyed with both the donor and acceptor dyes to obtain the FRET efficiency.

### Background

*Fluorescent resonance energy transfer* (FRET) refers to the nonradiative transfer of energy from an excited fluorochrome, called a *donor*, to a nearby fluorescent molecule, called an *acceptor*. The FRET technique measures the fluorescence signals of the donor, the acceptor, and the FRET signal. If FRET occurs, the donor channel signal is quenched and the acceptor channel signal is sensitized or increased.

The energy transfer efficiency  $E$  is conventionally defined as the number of energy transfer events divided by the number of photons absorbed by the donor, is related to the distance  $R$  between the acceptor and donor by:

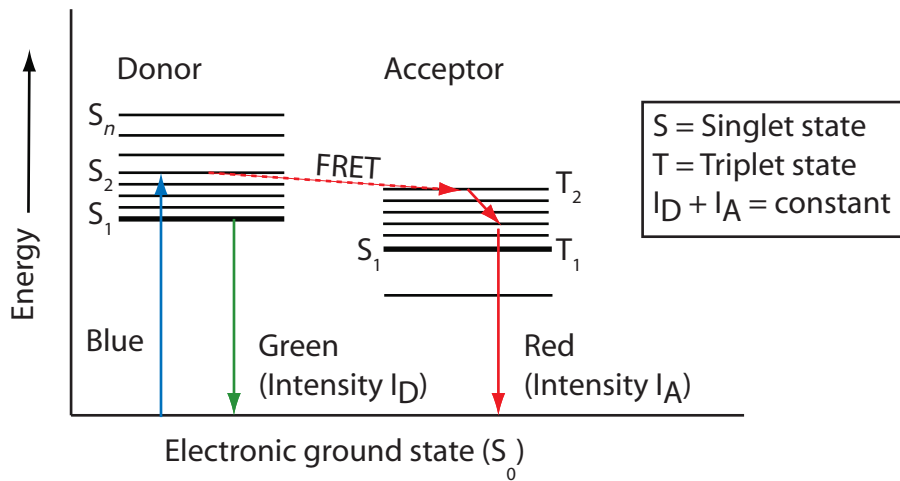
$$E = \frac{1}{\left[1 + \left(\frac{r}{R_0}\right)^6\right]}$$

EQUATION 1

where  $R_0$ , the Forster critical distance, is the distance at which  $E = 0.5$ .

Because FRET falls off as the sixth power of the distance between the donor and the acceptor, virtually no FRET occurs for distances greater than  $2R_0$ . Since  $R_0$  is on the order of 10 to 70 Angstroms, by performing FRET measurements it is possible to distinguish proteins that are merely nearby in the same compartment from those proteins that are interacting with each other. Virtually no FRET occurs for distances that are greater than  $2R_0$ .

Figure 1 schematically illustrates the process of fluorescence and FRET. When a fluorescent molecule (donor) is excited (using a blue photon in this example), the molecule goes into an excited state ( $S_1$ ) for a few nanoseconds. During this time the molecule loses some of its energy after which it returns to the ground state ( $S_0$ ) by emission of a less energetic photon (green photon in this example). If another fluorescence molecule (acceptor), which is excited by green photons and emits red photons, is very close by, then it is likely that the donor nonradiatively transfers its excitation energy to the acceptor and returns to the ground state without emitting a photon. As a consequence, the acceptor becomes excited and later returns to the ground state by giving off a photon of lower energy (red photon in this example).



**Figure 1. Jablonski diagram for FRET method**

*In acceptor photobleaching, since  $I_D + I_A = \text{constant}$ , if the acceptor is eliminated by photobleaching, then  $I_A$  goes to zero and  $I_D$  increases to constant.*

Measuring FRET using the sensitized emission method requires three sets of 2D images. You must first run the FRET Bleed Through algorithm once on three donor dye only images and once on three acceptor dye only images. You then run the FRET Efficiency algorithm on three images with both donor and acceptor dyes.

These images are:

- Image donor and/or acceptor dyes taken with a donor filter
- Image with donor and/or acceptor dyes taken with a FRET filter
- Image with donor and/or acceptor dyes taken with an acceptor filter

To obtain FRET efficiency, you need, therefore, a set of nine images. Table 1 on page 480 and Figure 2 on page 482 list and show the images that are used as examples in this discussion.

**Table 1. Setup of nine example images**

To obtain the FRET efficiency, you need to have a complete set of nine images modeled after the images listed here.

Algorithm	Dye used on images	Image name*	Filter used	Excitement wavelength		Emission wavelength	
				Donor	Acceptor	Donor	Acceptor
FRET Bleed Through	Acceptor	R3_543F.tif (Source)	Acceptor 1 (FP1)		✓		✓
		R3_488F.tif	FRET	✓			✓
		R3_488Y.tif	Donor 2 (PF2)	✓		✓	
	Donor	Y5_543F.tif	FP2		✓		✓
		Y5_488F.tif	FRET	✓			✓
		Y5_488Y.tif (Source)	FP1	✓		✓	
FRET Efficiency	Acceptor and donor	YR4_543F.tif (Source)	Acceptor fluorescence photobleaching (AFP)		✓		✓
		YR4_488F.tif	FRET	✓			✓
		YR4_488Y.tif	Donor fluorescence photobleaching (DFP)	✓		✓	

\*Refer to Figure 2 on page 482 to see these example images. The naming convention for these images is the following:

- R = Red (acceptor dye)
- Y = Yellow (donor dye)
- F = Filter for detecting acceptor signal
- Y after number = Filter for detecting donor signal
- 488 = Excitation wavelength for donor dye
- 543 = Excitation wavelength for acceptor dye



FP stands for "fluorescence protein". The term "FP1 filter" refers to the pair of filters normally used to acquire an image of fluorescence protein 1 (FP1). The pair of filters consists of an excitation filter and an emission filter. In order to measure bleed through, it is necessary to image each of the proteins



(1 and 2) with the correct filters (1 and 2), wrong filters (i.e., 2 and 1), as well as image each of the proteins with the FRET filter. The FRET filter uses the excitation filter for the donor protein and the emission filter for the acceptor protein. For the bleed-through calculations I avoided using the terms "donor" and "acceptor" and instead use "1" and "2" since the donor and acceptor are used equivalently for the bleed through calculations. In the second part of the calculation to calculate the FRET efficiency, it does make a difference which is the donor and which is the acceptor, therefore I use the terms DFP and AFP.

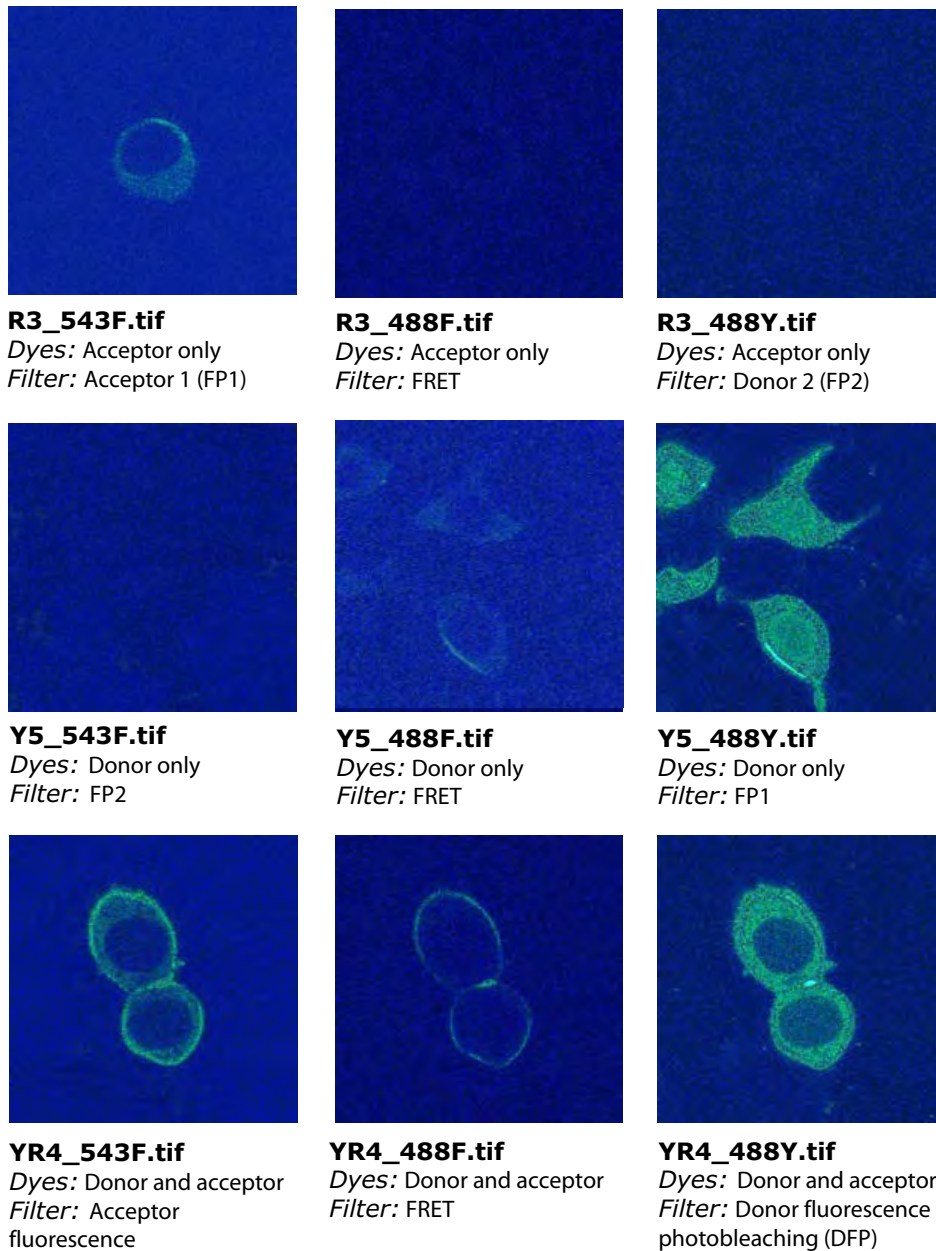


Figure 2. Nine example images used to calculate FRET efficiency

## FRET Bleed Through Algorithm

The FRET Bleed Through algorithm requires the six types of images listed in Table 2. You need to run the algorithm twice: Once on the set of three images with acceptor dye only (the first three images listed in the table); and once on the set of three images with donor dye only (the last three images listed in the table).

**Table 2. Six images used in applying the FRET Bleed Through algorithm**

Sets of images	Example image*	Type of image
Acceptor only dyed images	R3_543F.tif	An acceptor only dyed image taken with an acceptor filter set
	R3_488F.tif	An acceptor only dyed image taken with a FRET filter set
	R3_488Y.tif	An acceptor only dyed image taken with a donor filter set
Donor only dyed images	Y5_543F.tif	An donor only dyed image taken with an acceptor filter set
	Y5_488F.tif	An donor only dyed image taken with a FRET filter set
	Y5_488Y.tif	A donor only dyed image taken with a donor filter set

\*Refer to Figure 2 on page 482 to view these example images.

For both the acceptor dye only run and the donor dye only run, you need to create two or more VOI regions on the source image:

- **A background VOI**—The background region should have a smaller average intensity than the active region. Background VOIs appear in blue.
- **One or more active VOIs**—The area of the active regions are used for the bleed through calculations. Each different active VOI appears in a different nonblue color.

---

**Caution:** Do not use blue as an active VOI color.

---

You must create all of the VOIs on the source image, and the source image must be an image of the FP1 dye taken with a FP1 filter. That is, the source image must be either:

- An image with donor dye only taken with a donor fluorescent filter set
- An image with acceptor dye only taken with an acceptor fluorescent filter set

---

**Note:** Pixels that are saturated in any of the three images are excluded from all calculations.

---

### Performing the Acceptor dye only run

Running the algorithm on images in which only the acceptor dye was used obtains the following values for each active VOI region:

- AFP (acceptor fluorescence photobleaching) to FRET bleed through value
- AFP to DFP (donor fluorescence photobleaching) bleed through value

The algorithm displays these values in the Output window.

### Performing the Donor dye only run

Running the algorithm on a set of three images in which only the donor dye was used obtains the following values for each active VOI region, which are displayed in the Output window:

- DFP to FRET bleed through value
- DFP to AFP bleed through value

These values are displayed in the Output window.

### Moving to FRET Efficiency

Once the four bleed through parameters (AFP to FRET bleed through, AFP to DFP bleed through value, DFP to FRET bleed through, and DFP to AFP bleed through) are obtained, you can then run the FRET Efficiency algorithm. You run the FRET Efficiency algorithm on the last set of three images in Figure 2 on page 482. This last set of images contain both donor and acceptor dyes.

## Calculating the FRET and FP2 values

The FRET Bleed Through algorithm calculates the FRET and FP2 values using the following equations:

$$\text{denom} = \text{mean}(\text{FP1 dye with FP1 filter active VOI}) - \text{mean}(\text{FP1 dye with FP1 filter background VOI})$$

$$\text{FRET bleed-through} = [\text{mean}(\text{FP1 dye with FRET filter active VOI}) - \text{mean}(\text{FP1 dye with FP2 filter background VOI})] / \text{denom}$$

$$\text{FP2 bleed-through} = [\text{mean}(\text{FP1 dye with FP2 filter active VOI}) - \text{mean}(\text{FP1 dye with FP2 filter background VOI})] / \text{denom}$$

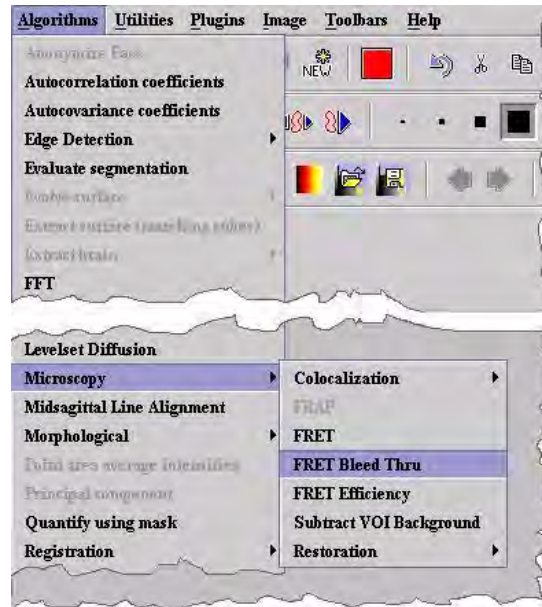
## Applying the FRET Bleed Through algorithm

The images used in the following instructions are the ones listed in Table 2 on page 483 and shown in the first two rows in Figure 2 on page 482.

### PERFORMING THE ACCEPTOR DYE ONLY RUN

To use this algorithm, do the following:

- 1** Select File > Open to open the image file on which the acceptor dye and FP1 filter were used (example image: R3\_543F.tif in Figure 2) (this image is the source image).
- 2** Select Algorithms > Microscopy > FRET Bleed Through (Figure 3).  
 The FRET Bleed Through dialog box (Figure 5 on page 488) opens.



**Figure 3. Algorithms > Microscopy > FRET Bleed Through command**

**Note:** If images on which you're running the algorithm are in color, the FRET Bleed Through dialog box shown in Figure 4A appears. This dialog box includes a Channels group, which allows you to select the color channel on which to run the algorithm. If the images are in grayscale, the Channels group does not appear. Instead, the dialog box shown in Figure 4B opens.

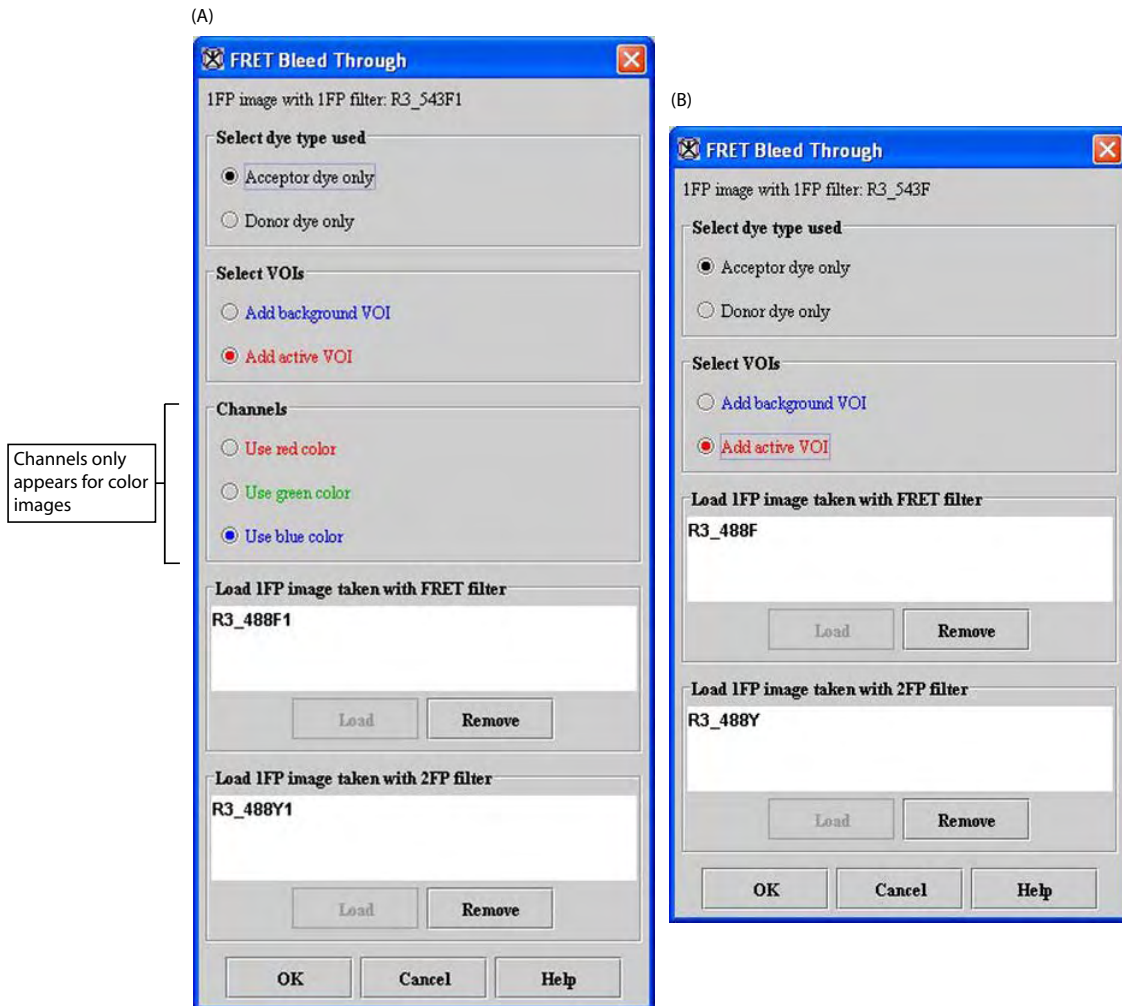










Figure 4. FRET Bleed Through dialog box for (A) color images and (B) grayscale images

- 3 Select Acceptor dye only in Select dye type used.
- 4 Select Add background VOI in Select VOIs.

- 5 Create a background VOI using the ellipse VOI , rectangle VOI , levelset VOI , or polyline VOI  on the image in the background of the image. This VOI appears in blue (Figure 5).
- 6 Select Add active VOI in Select VOIs.
- 7 Create an active VOI using the ellipse VOI , rectangle VOI , levelset VOI , or polyline VOI  in the foreground of the image, and adjust the contours of the VOI to eliminate any image background. This VOI appears in a *nonblue* color (Figure 5).

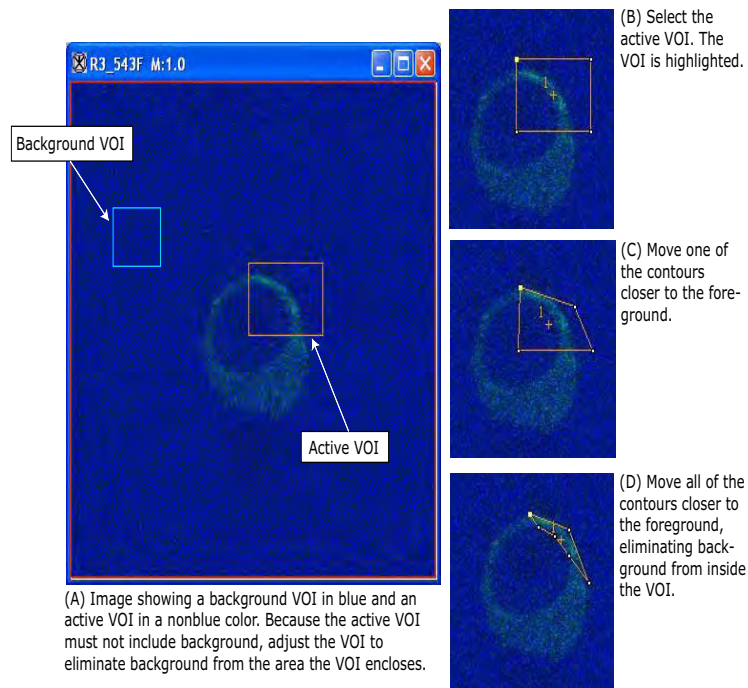







Figure 5. Acceptor only run: FP1 image with a background VOI in blue and an active VOI in orange (example image: R3\_543F.tif)

- 8 Decide whether to create additional active VOIs.



---

If you decide against creating additional active VOIs, go to the next step. If you decide to create additional active VOIs, do the following:

- a** Select , the New VOI icon.
- b** Create another active VOI using the ellipse VOI , rectangle VOI , levelset VOI , or polyline VOI , and adjust the contours of the VOI to eliminate any image background. This VOI appears in a nonblue color that is different from the color of the first active VOI.

---

**Note:** If you do not select , the New VOI icon, MIPAV creates additional contours of the same color for the first active VOI and sums all areas within the contours of this VOI together.

- 
- 9** Select Use red channel, Use green channel, or Use blue channel in Channels.

---

**Remember:** The Channels group only appears if the source image is in color.

- 
- 10** Click Load under Load 1FP image taken with FRET filter box. The Open Image dialog box (Figure 6) appears.

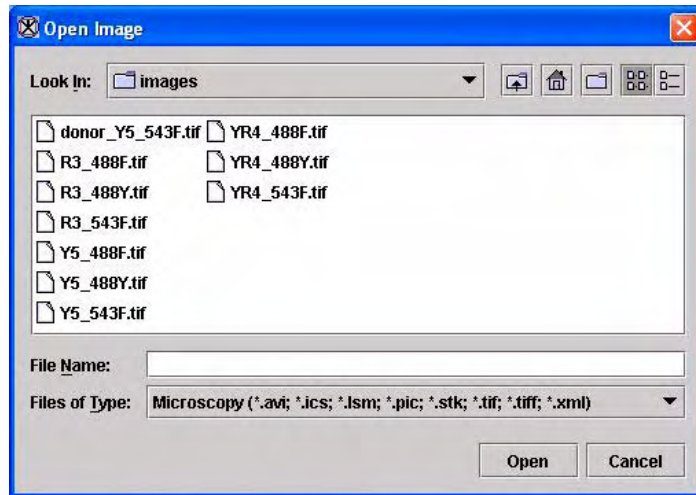


Figure 6. Open Image dialog box

- 11** Navigate to the directory where the image (example image: R3\_488F.tif) is stored, and select the image. (Refer to Figure 2 on page 482 to see this image.)
- 12** Click Open. The name of the image appears in the Load 1FP image taken with FRET filter box (Figure 4 on page 487).

---

**Note:** The Remove button only becomes enabled when an image appears in the box (as it does in Figure 4 on page 487).

---

- 13** Click Load under the Load 1FP image taken with 2FP filter box. The Open Image dialog box (Figure 6) appears.
- 14** Navigate to the directory where the image (example image: R3\_488Y.tif) (refer to Figure 2 on page 482 to see this image) is stored, and select the image.
- 15** Click Open. The name of the image appears in the Load 1FP image taken with 2FP filter box (refer to Figure 4 on page 487).
- 16** Click OK. The algorithm begins to run. When the algorithm finishes, it places the data in the Output window (Figure 7).

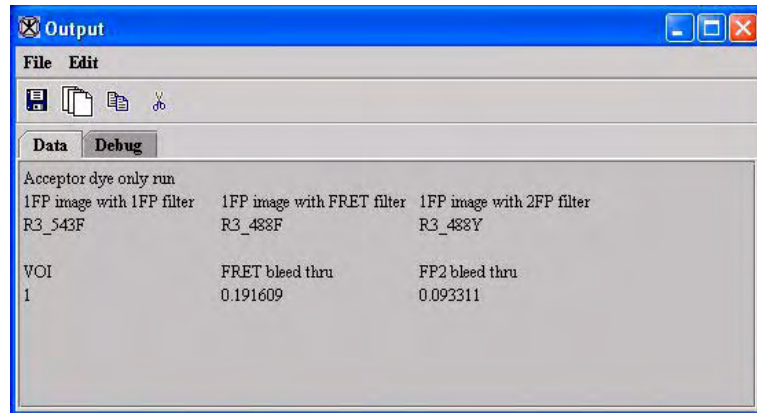


Figure 7. Results in the Output window from the Acceptor dye only run of the FRET Bleed Through algorithm

## PERFORMING THE DONOR DYE ONLY RUN

- 1** Select File > Open to open the image file on which the donor dye and FP2 filter were used (example image: Y5\_488Y.tif in Figure 2 on page 482) (this image is the source image).
- 2** Select Algorithms > Microscopy > FRET Bleed Through. The FRET Bleed Through dialog box (Figure 4 on page 487) opens.
- 3** Select Donor dye only in Select dye type used (Figure 8).

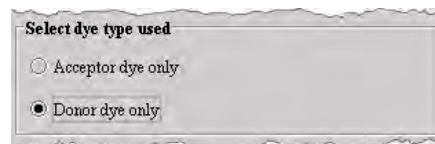


Figure 8. Donor dye only selection

- 4** Select Add background VOI in Select VOIs (Figure 9).



Figure 9. Add background VOI selection














- 5 Create a background VOI using the ellipse VOI , rectangle VOI , levelset VOI , or polyline VOI  on the image in the background of the image. This VOI appears in blue (Figure 11).
- 6 Select Add active VOI in Select VOIs.



Figure 10. Add active VOI selection

- 7 Create an active VOI using the ellipse VOI , rectangle VOI , levelset VOI , or polyline VOI  in the foreground of the image, and adjust the contours of the VOI to eliminate any image background. This VOI appears in a nonblue color (Figure 11).
- 8 Decide whether to create additional active VOIs.
  - If you decide against creating additional active VOIs, go to the next step.
  - If you decide to create additional active VOIs, do the following:
    - a Select , the New VOI icon.
    - b Create another active VOI using the ellipse VOI , rectangle VOI , levelset VOI , or polyline VOI , and adjust the contours of

the VOI to eliminate any image background. This VOI appears in a nonblue color that is different from the color of the first active VOI.



**Note:** If you do not select , the New VOI icon, MIPAV creates additional contours of the same color for the first active VOI and sums all areas within the contours of this VOI together.

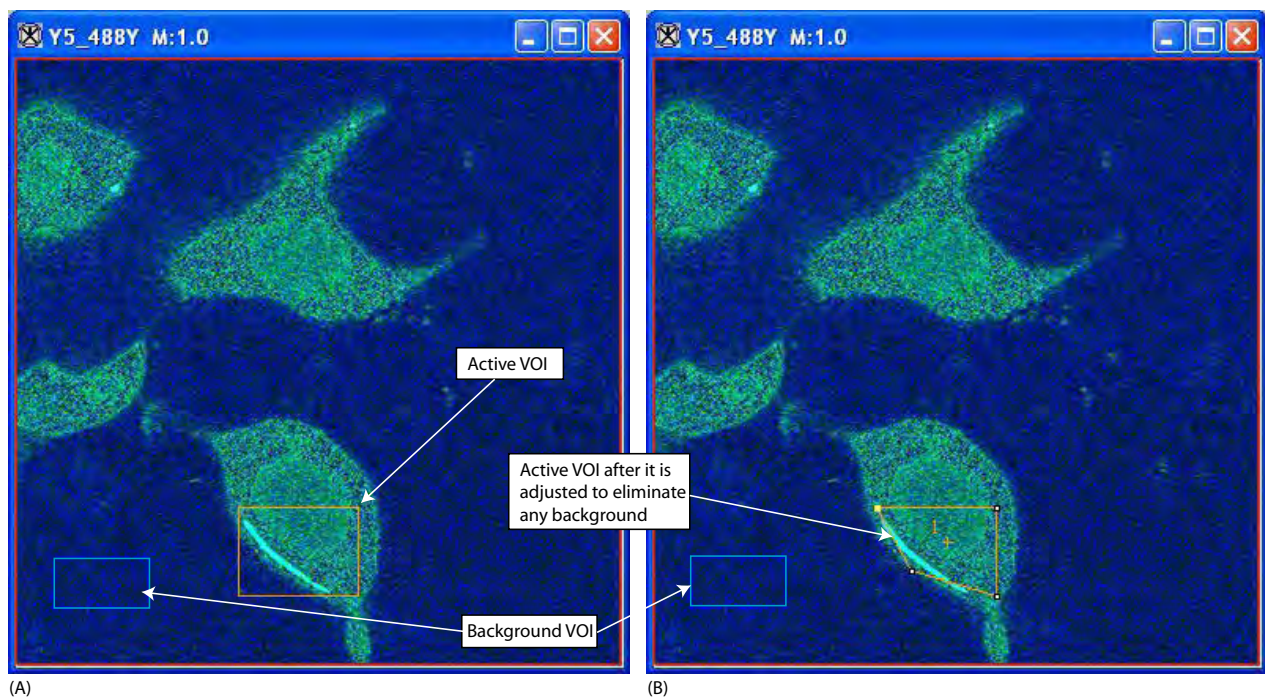


Figure 11. Donor only run: FP1 image with a background VOI in blue and an active VOI in orange (example image: Y5\_488Y.tif)

- 9 Select Use red channel, Use green channel, or Use blue channel in Channels if the images are in color.

**Remember:** The Channels group only appears if the source image is in color.

- 10 Click Load under Load 1FP image taken with FRET filter box. The Open Image dialog box (Figure 12) appears.

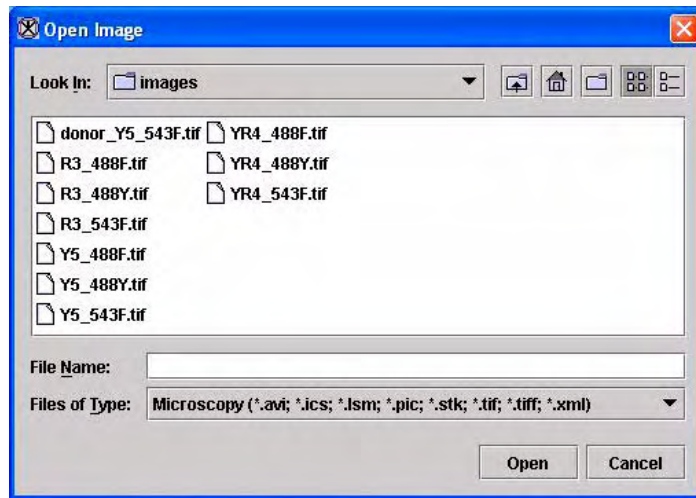


Figure 12. Open Image dialog box

- 11** Navigate to the directory where the image (example image: Y5\_488F.tif) is stored, and select the image. (Refer to Figure 2 on page 482 to see this image.)
- 12** Click Open. The name of the image appears in the Load 1FP image taken with FRET filter box (Figure 13).

---

**Note:** The Remove button becomes enabled when an image appears in the box.

---

- 13** Click Load under the Load 1FP image taken with 2FP filter box. The Open Image dialog box (Figure 12) appears.
- 14** Navigate to the directory where the image (example image: Y5\_543F.tif) is stored, and select the image. The name of the image appears in the Load 1FP image taken with 2FP filter box (Figure 13).



Figure 13. The loaded images in the FRET Bleed Through dialog box

- 15 Click OK. The algorithm begins to run. When the algorithm finishes, it places the data in the Output window (Figure 14).

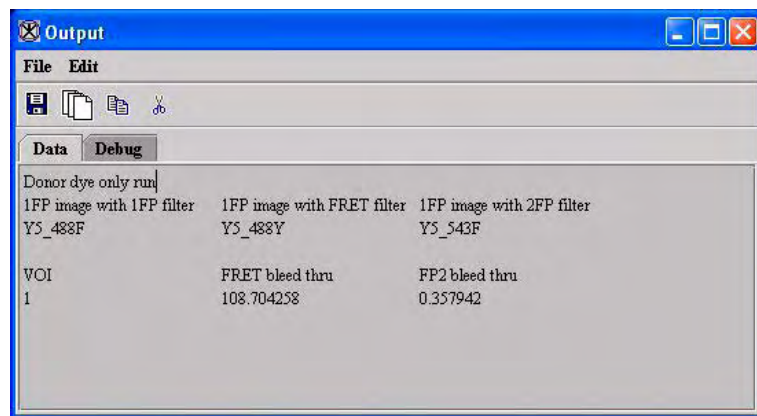


Figure 14. Results in the Output window from the Donor dye only run of the FRET Bleed Through algorithm

<p><b>Select dye type used</b></p>	<p><b>Acceptor dye only</b>—The dye used on images that are processed through the acceptor-dye run of the FRET Bleed Through algorithm. As a result of this run, the algorithm calculates the AFP to FRET bleed through value and the AFP to DFP bleed through value.</p> <hr/> <p><b>Donor dye only</b>—The dye used on images that are processed through the donor dye only run of the FRET Bleed Through algorithm. During this run, the algorithm calculates the DFP to FRET bleed through value and the DFP to AFP bleed through value.</p>	
<p><b>Select VOIs</b></p>	<p><b>Add background VOI</b>—Indicates a region in the image that is the background. Using the ellipse VOI, rectangle VOI, levelset VOI, or polyline VOI, you may create one background VOI. The background VOI appears in blue.</p> <hr/> <p><b>Add active VOI</b>—Indicates one or more regions in the image that are in the foreground. You may create the first active VOI using the ellipse, rectangle, levelset, or polyline VOI. To create more active VOIs, first select New VOI and then select ellipse, rectangle, levelset, or polyline VOI. Each active VOI appears in a different, nonblue color.</p>	
<p><b>Channels</b></p>	<p><b>Use red color</b>—Selects the red channel for the FRET analysis.</p> <hr/> <p><b>Use green color</b>—Selects the green channel for the FRET analysis.</p> <hr/> <p><b>Use blue color</b>—Selects the blue channel for the FRET analysis.</p>	
<p><b>Load FP1 image taken with FRET filter</b></p>	<p>Lists the FRET filtered image that you selected after clicking Load.</p>	

Figure 15. FRET Bleed Through dialog box



<b>Load</b>	Allows you to load an FP1 image that was taken with the FRET filter. Clicking this button causes the Open Image dialog box—from which you can select an image—to appear.
<b>Remove</b>	Removes the image that is selected in the Load FP1 image taken with FRET filter box.
<b>Load FP1 image taken with FP2 filter</b>	Lists the FP2 filtered image that you selected after clicking Load.
<b>Load</b>	Allows you to load an FP1 image that was taken with the FP2 filter. Clicking this button causes the Open Image dialog box—from which you can select an image—to appear.
<b>Remove</b>	Removes the image that is selected in the Load FP2 image taken with FP2 filter box.
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

Figure 15. FRET Bleed Through dialog box (continued)

## FRET Efficiency algorithm

This FRET Efficiency algorithm uses the four bleed through parameters obtained from running the FRET Bleed Through algorithm. It requires a set of three images:

Sets of images	Example image*	Type of image
Donor and acceptor dyed images	YR4_543F.tif	An acceptor- and donor-dyed image taken with an acceptor filter set
	YR4_488F.tif	An acceptor- and donor-dyed image taken with a FRET filter set
	YR4_488Y.tif	An acceptor- and donor-dyed image taken with a donor filter set

\*Refer to Figure 2 on page 482 to view these example images.

- Donor- and acceptor-dyed image taken with a donor fluorescence photobleaching (DFP) filter
- Donor- and acceptor-dyed image taken with a FRET filter

- Donor- and acceptor-dyed image taken with an acceptor fluorescence photobleaching (AFP) filter

The algorithm assumes the presence of two or more VOI regions on the source image:

- **A background VOI**—The background region has a smaller average intensity than the active region. Background VOIs appear in blue.
- **One or more active regions**—The area of the active regions, or VOIs, are used for the efficiency calculations. Each different active VOI appears in a different nonblue color.



**Caution:** Do not use blue as an active VOI color.



**Note:** Pixels with saturation values in any of the three images are excluded from all calculations.

For each active VOI region, this algorithm outputs the FRET efficiency and the adjusted donor and adjusted acceptor intensities.

## Applying the FRET Efficiency algorithm

The images used in these instructions were the ones shown in Table 2 on page 483 and shown in the last row in Figure 2 on page 482. You only need to run the FRET Efficiency algorithm once on the images.

To use this algorithm, do the following:

- 1 Select File > Open to open the image file on which both the acceptor dye and donor dyes were used and DFP filter were used (example image: YR4\_488Y.tif).
- 2 Select Algorithms > Microscopy > FRET Efficiency. The FRET Efficiency dialog box (Figure 16) opens.

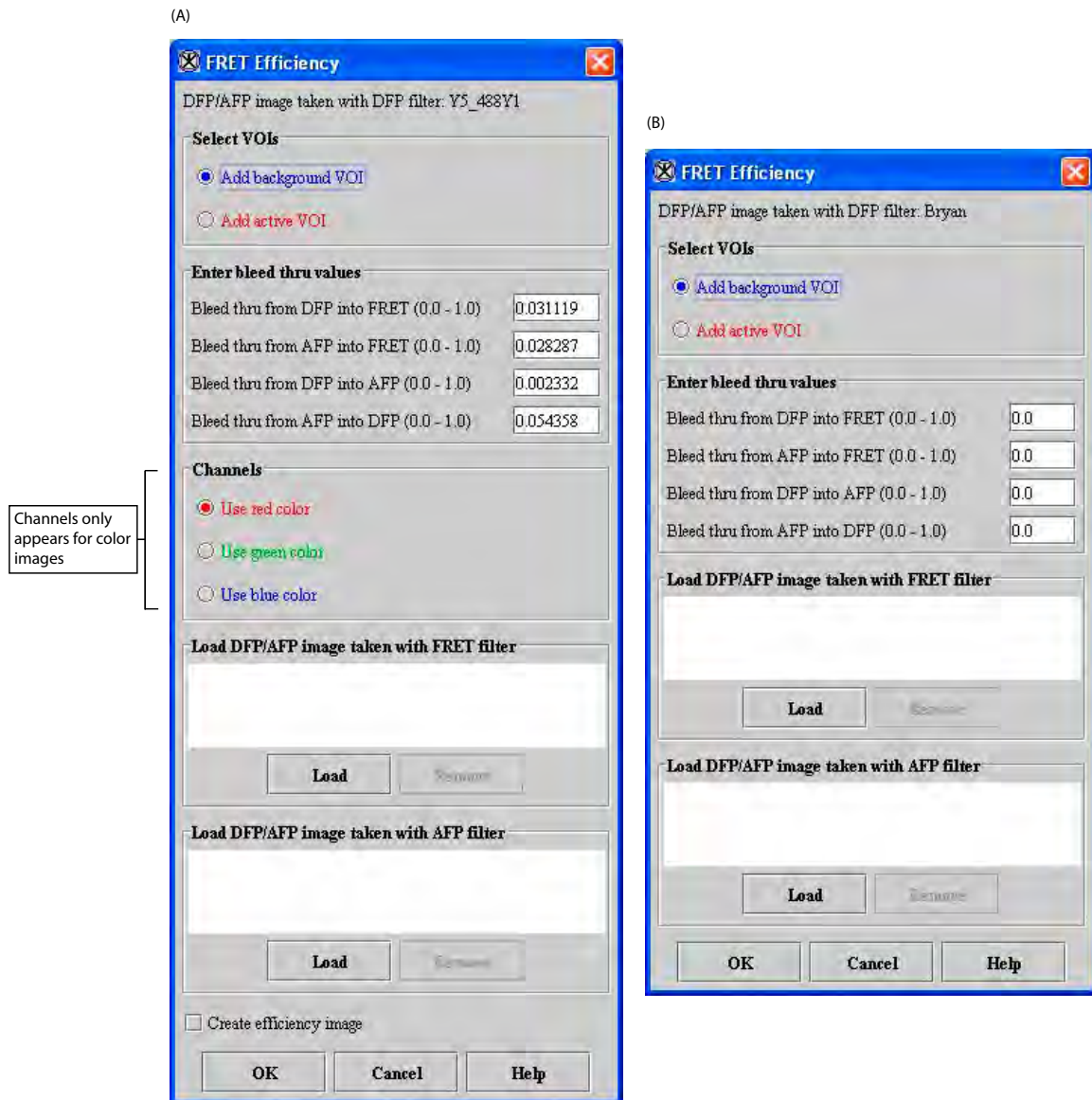






Figure 16. FRET Efficiency dialog box (A) color images and (B) grayscale images

- 3** Select Add background VOI in Select VOIs on the FRET Efficiency dialog box.
- 4** Create a background VOI on the image in the background of the image. This VOI appears in blue lines.





**5** Select Add active VOI in Select VOIs on the FRET Efficiency dialog box.

**6** Create an active VOI using the ellipse VOI , rectangle VOI , levelset VOI , or polyline VOI  in the foreground of the image, and adjust the contours of the VOI to eliminate any image background. This VOI appears in a nonblue color (Figure 11).

**7** Decide whether to create additional active VOIs.

If you decide against creating additional active VOIs, go to the next step. If you decide to create additional active VOIs, do the following:

**a** Select , the New VOI icon.

**b** Create another VOI using the ellipse VOI , rectangle VOI , levelset VOI , or polyline VOI , and adjust the contours of the VOI to eliminate any image background. This VOI appears in a nonblue color that is different from the color of the first active VOI.



**Note:** If you do not select , the New VOI icon, MIPAV creates additional contours of the same color for the first active VOI and sums all areas within the contours of this VOI together.



**Note:** MIPAV prefills the Enter bleed through values. However, if the correct bleed through values are not automatically entered, type them into the fields.

**8** Click Load under Load DFP/AFP image taken with FRET filter box. The Open Image dialog box appears.

**9** Navigate to the directory where the image (example image: YR4\_488F.tif) is stored, and select the image.

**10** Click Open. The name of the image appears in the Load DFP/AFP image taken with FRET filter box.



**Note:** The Remove button becomes enabled when an image appears in the box.

- 11** Click Load under the Load DFP/AFP image taken with AFP filter box. The Open Image dialog box appears.
- 12** Navigate to the directory where the image (example image: YR4\_543F.tif) is stored, and select the image. The name of the image appears in the Load DFP/AFP image taken with AFP filter box.
- 13** Click OK. MIPAV places the data from this second part of the procedure in the Output window (Figure 7).

<b>Select VOIS</b>	<p><b>Add background VOI</b>— Indicates a region in the image that is the background, which has a smaller average intensity than the active region. Using the ellipse VOI, rectangle VOI, levelset VOI, or polyline VOI, you create a background VOI. The background VOI appears in blue.</p> <p><b>Add active VOI</b>— Indicates one or more regions in the image that are in the foreground. You may create the first active VOI using the ellipse VOI, rectangle VOI, levelset VOI, or polyline VOI. To create more active VOIs, select New VOI. Each different active VOI appears in a different nonblue color.</p>	
<b>Bleed through from DFP into FRET (0.0–1.0)</b>	Indicates the bleed through value from DFP into FRET.	
<b>Bleed through from AFP into FRET (0.0–1.0)</b>	Indicates the bleed through value from AFP into FRET.	
<b>Bleed through from DFP into AFP (0.0–1.0)</b>	Indicates the bleed through value from DFP into AFP.	
<b>Bleed through from AFP into DFP (0.0–1.0)</b>	Indicates the bleed through value from AFP into DFP.	

Figure 17. FRET Efficiency dialog box

<b>Channels</b>	<b>Use red color</b> —Selects the red channel.
	<b>Use green color</b> —Selects the green channel.
	<b>Use blue color</b> —Selects the blue channel.
<b>Load DFP/AFP image taken with FRET filter</b>	Lists the FRET filtered image that you selected after clicking Load.
<b>Load</b>	Allows you to load a DFP/AFP image that was taken with the FRET filter. Clicking this button causes the Open Image dialog box—from which you can select an image—to appear.
<b>Remove</b>	Removes the image that is selected in Load DFP/AFP image taken with FRET filter.
<b>Load DFP/AFP image taken with AFP filter</b>	Lists the AFP filtered image that you selected after clicking Load.
<b>Load</b>	Allows you to load a DFP/AFP image that was taken with the AFP filter. Clicking this button causes the Open Image dialog box—from which you can select an image—to appear.
<b>Remove</b>	Removes the image that is selected in Load DFP/AFP image taken with AFP filter.
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes this dialog box.
<b>Help</b>	Displays online help for this dialog box.

Figure 17. FRET Efficiency dialog box (continued)

## IMAGE TYPES

You apply both algorithms to three 2D images.

## NOTES

Only one user-selected color can be used from a color image.

Both algorithms port the MATLAB routines provided by Dr. Stephen Lockett.

---

## REFERENCES

Refer to the following references for more information about FRET.

Gordon, Gerald W., Gail Berry, Xiao Huan Liang, Beth Levine, and Brian Herman.  
"Quantitative Fluorescence Resonance Energy Transfer Measurements Using Fluorescence  
Microscopy." *Biophysical Journal* 74(May 1998):2702–2713.

Kenworthy, Anne K. "Imaging Protein-Protein Interactions Using Fluorescence Energy  
Transfer Microscopy." *Methods*, 24(2001):289–296.

---

## Microscopy (Restoration): Computational Optical Sectional Microscopy (COSM)

This algorithm removes out-of-focus light in 3D volumes collected plane by plane using either widefield or confocal fluorescence microscopes. The implementation of this algorithm, detailed by Conchello and Hansen (1997), incorporates a roughness penalty into a regularized expectation maximization restoration procedure described by Conchello (1995). This implementation also includes an intensity penalty as detailed by Conchello and McNally (1996) (see “References” on page 508).

### Background

Image restoration algorithms are used to compute a more accurate version of the imaged volume by removing artifacts of the imaging system. The sample microscopy imaging system is shown in Figure 1.

For example, an observed image function  $g(x, y, z)$  can be written mathematically as:

$$g(x, y, z) = h(x, y, z) \otimes f(x, y, z) + n(x, y, z)$$

where

$f(x, y, z)$  = true or undistorted image function

$h(x, y, z)$  = transfer function of the imaging system

$n(x, y, z)$  = noise term

$\otimes$  = denotes convolution

The purpose of image restoration is to utilize the observed image  $g(x, y, z)$ , the known transfer function  $h(x, y, z)$ , and the estimate of the noise  $n(x, y, z)$  to compute a better estimate of  $f(x, y, z)$ .



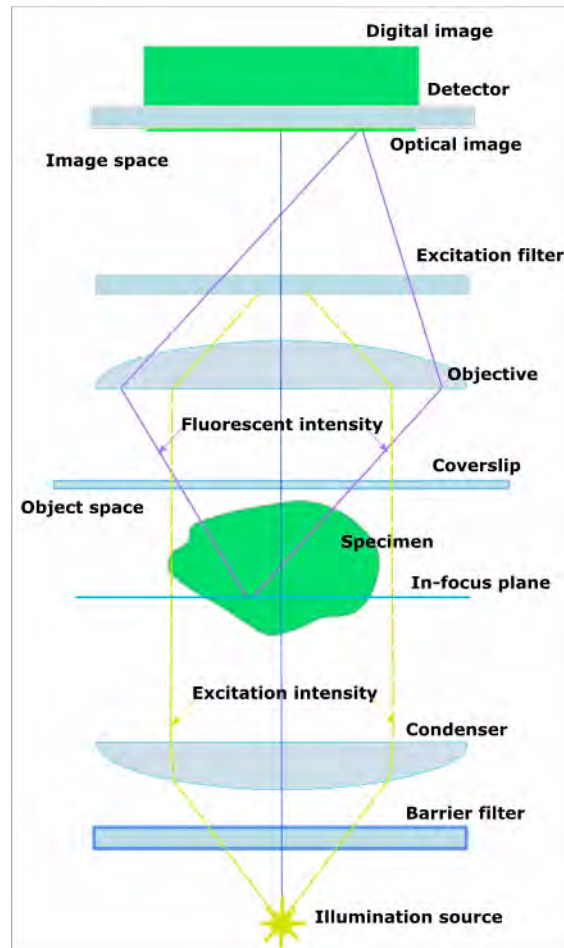
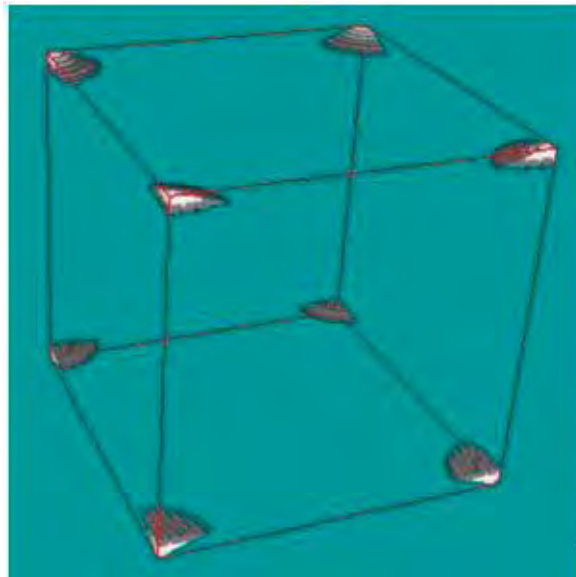


Figure 1. Image formation and recording

To run this algorithm, you, first, need to open an image of a system transfer function ( $h(x, y, z)$ ) of the imaging system and an observed image. One of the first computations performed by this algorithm is to convert the transfer function into a *point spread function* (PSF) by computing its Fourier transform. Because it is customary to refer to the PSF instead of the transfer function, this document follows that convention.

It is necessary that the observed image have the mouse focus when selecting the algorithm from the Algorithms menu. When the algorithm is selected, the XCOSM Expectation Maximum Restoration dialog box opens, presenting several parameters to specify, including the name of the open PSF image.

The PSF utilized by the algorithm is a processed version of the actual PSF by shifting each pixel in it by one-half of the images dimensions. This compensates for the standard shifting operation typically applied while computing the Fast Fourier Transform (FFT). Figure 2 shows the shifted transfer function that is an input into the X\_COSM algorithm.



**Figure 2. The shifted version of the transfer function conventionally, but incorrectly, called the point spread function (PSF)**

The observed image used in this documentation is a microscopic bead, a subject to the blurring introduced by the imaging system. Figure 3 shows a volume rendering of the blurred bead.

The algorithm allows processing lower resolution volumes to remove some of the blur, and thereby, speed up the processing of an image set.

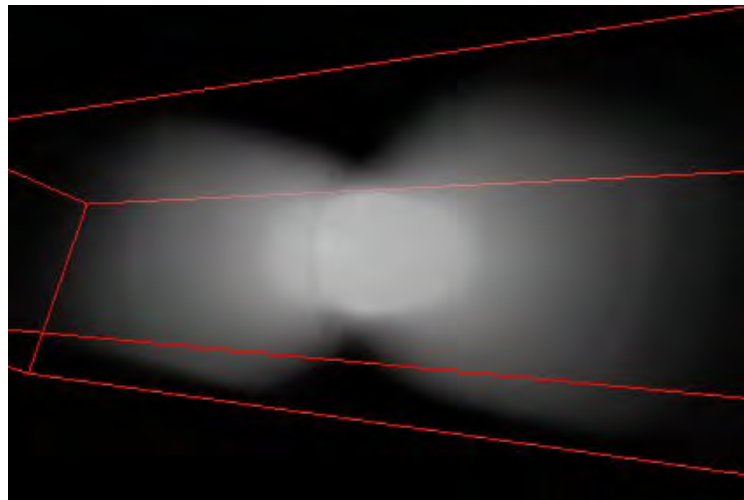


Figure 3. The observed bead used in this documentation. The red lines correspond to the bounding box of the image

---

## ADJUSTING THE SPEED OF THE PROCESSING

The following three parameters that also appear as options in the X\_COSM Expectation Maximum Restoration dialog box can be used to adjust the speed of the processing, depending on the specified values:

- Original size iteration percentage
- Half size iteration percentage
- Quarter size iteration percentage

Each of these boxes specify values that indicate the percentage of iterations that are performed at full resolution, half of the resolution, and quarter of the resolution respectively. The sum of the values in these boxes must add up to 100 percent, or the algorithm computes correct percentages starting with the value for full resolution. See Figure 5.

A factor that affects the speed of image processing is the value specified in the *Number of iterations* box. The length of time the algorithm spends in processing the image depends on how many times it iterates over the image.

In addition, the *Backup number of iterations* parameter specifies the number of iterations that should occur before the algorithm writes the

currently processed image to a disk. The value you specify in this box determines whether the algorithm saves intermediate results or not.

The algorithm allows for either an *intensity penalty* (Conchello and McNally, 1996) or a *roughness penalty* (Markham and Conchello, 1997). These constraints limit the amount of processing when intensity or roughness exceeds a set value, which controls the amount of deblurring that can occur. The penalty value is an exponential value that controls the amount of variation in intensity or roughness that is deblurred.

The final options on the dialog box—Entire image and VOI regions—permit restoring the whole image or those areas of the image that fall within one or more VOIs.

---

## REFERENCES

For information about XCOSM, a version of COSM written for the X-Window system, see <http://www.essrl.wustl.edu/~preza/xcosm/>.

For more information, use the following references.

Conchello, Jose-Angel. "Fluorescence Photobleaching Correction for Expectation-Maximization Algorithm." In *Proceedings of SPIE, 1995: Three-Dimensional Microscopy: Image Acquisition and Processing II*, edited by T. Wilson and C. J. Cogswell, Bellingham, Washington: International Society for Optical Engineering, 1995.

Conchello, Jose-Angel. "Superresolution and Convergence Properties of the Expectation-Maximization Algorithm for Maximum-Likelihood Deconvolution of Incoherent Images." *Journal of the Optical Society of America A*, 15, no. 10 (October 1998):2609-2619.

Conchello, Jose-Angel, John J. Kim, and Eric W. Hansen. "Enhanced Three-Dimensional Reconstruction from Confocal Scanning Microscope Images; II. Depth Discrimination Versus Signal-to-Noise Ratio in Partially Confocal Images." *Applied Optics*, 33, no. 17 (June 1994):3740-3750.

Conchello, Jose-Angel, and James G. McNally. "Fast Regularization Technique for Expectation Maximization Algorithm for Computational Optical Sectioning Microscopy." In *Proceedings of SPIE, 1996: Three-Dimensional Microscopy: Image Acquisition and Processing III*, edited by Carol J. Cogswell, Gordon S. Kino, and Tony Wilson. Washington: International Society for Optical Engineering, 1996.

Markham, Joanne, and Conchello, Jose-Angel. "Trade-offs in Regularized Maximum-Likelihood Image Restoration." In *Proceedings of SPIE, 1997: Three-Dimensional Microscopy: Image Acquisition and Processing IV*, edited by Carol J. Cogswell, Jose-Angel Conchello, and Tony Wilson. Washington: International Society for Optical Engineering, 1997.

Preza, Chrysanthe, Michael I. Miller, and Jose-Angel Conchello. "Image Reconstruction for 3D Light Microscopy with a Regularized Linear Method Incorporating a Smoothness Prior." In *Proceedings of SPIE, 1905(1992): Biomedical Image Processing and Biomedical Visualization*, edited by Raj S. Acharya and Dmitry B. Goldgof. Washington: International Society for Optical Engineering, 1992.

Preza, Chrysanthe, Michael I. Miller, L. J. Thomas Jr., and J.G. McNally. "Regularized Linear Method for Reconstruction of Three-Dimensional Microscopic Objects from Optical Sections." *Journal of the Optical Society of America A*, 9(February 1992):219–228.

---

## IMAGE TYPES

All types of 3D microscopy images that can be opened in MIPAV.

### Applying the X\_COSM algorithm

To run this algorithm, do the following:

- 1 Open an observed image and a PSF image (Figure 4A and B).
- 2 Select the observed image to give it a mouse focus.
- 3 Select the Save User Defined LUT icon, to save the lookup table (LUT) for the observed image.
- 4 Select the PSF image, which gives it the mouse focus.
- 5 Select the Open User Defined LUT icon, which applies the LUT that you saved for the observed image to the PSF image.

---

**Tip:** The PSF image may look as if it is totally blank (Figure 4b). You may not see an image even if you select the Image Slice slider to look at all of the slices in the image. To improve the contrast in the PSF image, select the Quick LUT icon, and drag it over the PSF image to improve the contrast (Figure 4d).

---

- 6 Select the observed image to give it the mouse focus.
- 7 Select Algorithms > Microscopy > Restoration > XCOSM\_EM.

The XCOSM Expectation Maximum Restoration dialog box (Figure 5) opens. Complete the dialog box, or simply accept the default values. See also "Adjusting the speed of the processing" on page 507.

- 8 Click OK. The algorithm begins to run. A message appears showing the status of image processing, which may take a while depending on the number of iterations requested in the XCOSM Expectation Maximum Restoration dialog box. Once processing is complete, a restored image appears. See Figure 4d.

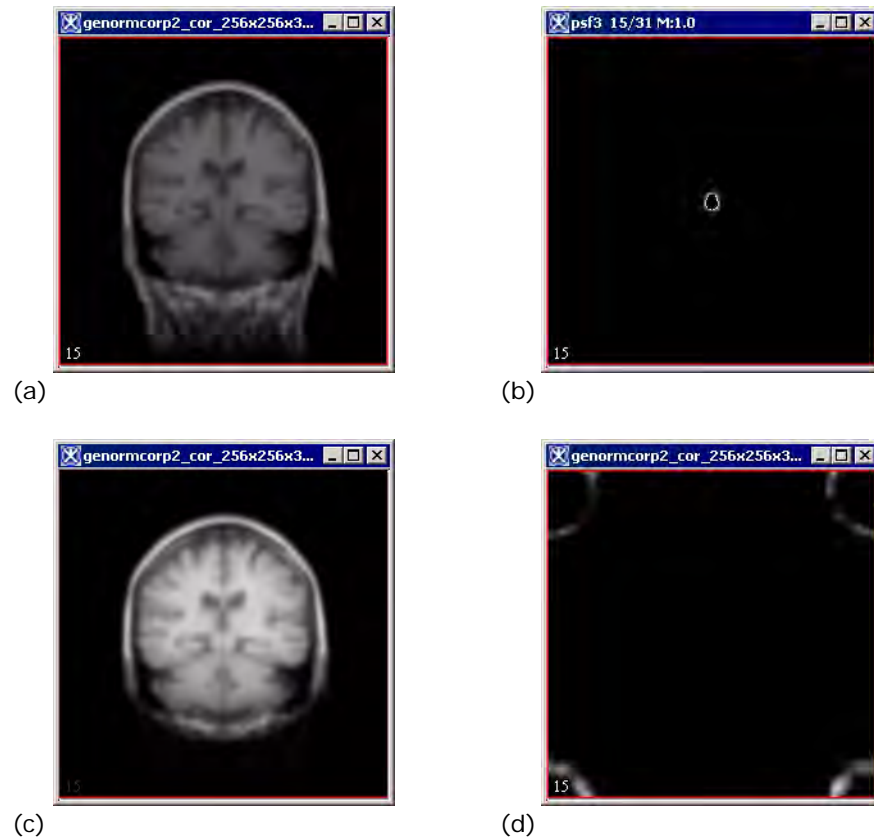


Figure 4. Observed and PSF images: (a) observed image, (b) PSF image as it appears initially, (c) restored image, and (d) PSF image after applying the Quick LUT icon

<p><b>Restoration of [file name] with PSF</b></p>	<p>Indicates the name of the image with the focus (in this example, the image name is <i>bead</i>) and the file name of the image that contains the point spread function of the imaging system (in this case, <i>psfbead</i>).</p> <p>The default image to be restored is the image that had the focus before the image that is to be restored.</p>	
<p><b>Window lower limit in Z</b></p>	<p>Allows restoration to begin at slices other than the first image slice, thus allowing the restoration of a portion of an image.</p>	
<p><b>Estimate decay</b></p>	<p>Indicates to estimate the exponential decay associated with the blur using the value in Decay constant. If you do not select this option, the value in Decay constant is ignored.</p>	
<p><b>Decay constant</b></p>	<p>Uses the specified value for estimating the exponential decay associated with the blur.</p>	
<p><b>Original size iteration percentage</b></p>	<p>Indicates the percentage of iterations to be performed at full resolution.</p> <p>The percentage specified here plus the percentages specified in Half size iteration percentage and Quarter size iteration percentage must add to 100 percent, or the algorithm computes correct percentages starting with the value for full resolution.</p>	
<p><b>Half size iteration percentage</b></p>	<p>Indicates the percentage of iterations to be performed at one-half the resolution.</p> <p>The percentage specified here plus the percentages specified in Half size iteration percentage and Quarter size iteration percentage must add to 100 percent, or the algorithm computes correct percentages starting with the value for full resolution.</p>	
<p><b>Quarter size iteration percentage</b></p>	<p>Indicates the percentage of iteration to be performed at one-quarter the resolution.</p> <p>The percentage specified here plus the percentages specified in Half size iteration percentage and Quarter size iteration percentage must add to 100 percent, or the algorithm computes correct percentages starting with the value for full resolution.</p>	

Figure 5. XCOSM Expectation Maximum Restoration dialog box

<b>Number of iterations</b>	Indicates the number of times the Expectation Maximization algorithm iterates over the image.
<b>Backup number of iterations</b>	Specifies the number of iterations should occur before the algorithm writes the currently processed image to disk. Specifying this value provides a mechanism for saving intermediate results.
<b>Intensity penalty</b>	Limits the amount of processing when intensity exceeds the value specified in the Penalty value box, controlling the amount of deblurring that can occur.
<b>Roughness penalty</b>	Limits the amount of processing when roughness exceeds the value specified in the Penalty value box, controlling the amount of deblurring that can occur.
<b>Penalty value</b>	Specifies a exponent value that controls the amount of variation in intensity or roughness that is deblurred.
<b>Entire image</b>	Restores the entire image.
<b>VOI regions</b>	Restores only those values defined in one or more volumes of interest, or VOIs.
<b>OK</b>	Applies this algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

Figure 5. XCOSM Expectation Maximum Restoration dialog box (continued)



## Midsagittal Line Alignment

*This section presents a detailed description of the Midsagittal Line Alignment algorithm which is used for the automatic detection of the midsagittal line in 2D and midsagittal plane in 3D brain images.*

The midsagittal line alignment is important because the ideal coordinate system attached to the head, in which the inter-hemispheric fissure is conveniently displayed, differs from the coordinate system of the image because of the usual tilt of the patient's head. It means that the midsagittal fissure is generally not displayed in the center of the image lattice where it should be. This might prevent from further visual inspection or analysis of the image, because the homologous anatomical structures or functional areas in both hemispheres are not displayed in the same axial or coronal slice in the 3D image. See Figure 1.

The algorithm searches for the *midsagittal plane* in 3D images or for *midsagittal line* in 2D images.

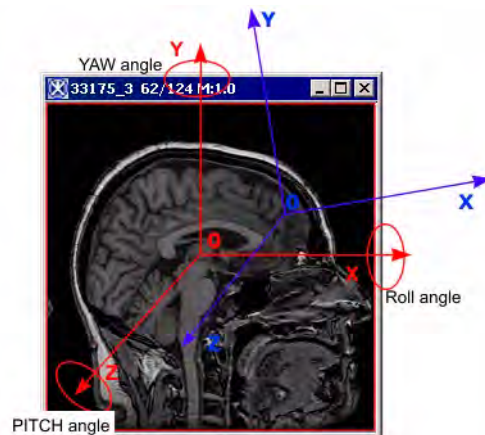


Figure 1. The ideal coordinate system attached to the head in which the fissure is close to the plane  $Z = 0$  (red) and the coordinate system of the image (blue) differ from each other by way of three angles (yaw, roll and pitch) and a 3D translation.

## Background

Midsagittal plane is the plane with respect to which the image exhibits

maximum symmetry. Therefore, the algorithm defines the midsagittal plane as the one that maximizes the similarity between the image and its symmetric. The Cross-Correlation cost function is used as a similarity measure.

Symmetry is measured by flipping the image horizontally and then registering the flipped image against original. The plane that shows maximal symmetry, and thus approaches the cost function global minimum is a midsagittal plane. The algorithm uses a multi-resolution approach to evaluate the cost function. To find a global minimum of the cross-correlation cost function, the algorithm uses the Powell method.

---

## OUTLINE OF THE METHOD:

- 1** The minimum resolution of the image is determined.
- 2** The image is flipped horizontally and the flipped image is saved.
- 3** Both images are, then, resampled and interpolated to create high resolution isotropic voxels. The algorithm uses Trilinear interpolation to subsample the images.
- 4** The center of mass (COM) for both images is computed and one translation step is performed to align the COMs.
- 5** The iterative method is used to register original image against flipped. Refer to “Iterative method used to register original image against flipped” on page 514.

---

## ITERATIVE METHOD USED TO REGISTER ORIGINAL IMAGE AGAINST FLIPPED

- 1** The first estimation of the midsagittal plane is made based on low density and low resolution. Both images –the original and flipped – are subsampled and interpolated to a low resolution (8 times).
- 2** The coarse angle step (45 degrees with 15 degrees step) is used in each dimension when registering the original low resolution image against the flipped image. For each angle configuration, a six degrees of freedom (6-DOF) local optimization is also performed to find the

optimal midsagittal aligning. The parameters of the transformation are then systematically varied, where the cost function is evaluated for each setting. The parameters corresponding to the smallest value(s) of the cost function are, then, used as the initial transformation for the next level in the optimization.

- 3** For each parameter setting corresponding to the top 20% of the cost function minima, the algorithm performs rotations over the fine grid (which is by default 15 degrees with 6 degrees step) and evaluates the cost function.
- 4** Three best minima with corresponding transformation parameters are stored in a vector of minima and then used as initial parameters for the next optimization level.
- 5** The images are resampled and interpolated to a higher resolution (4 times). The transformation corresponding to the smallest value(s) of the cost function is determined and used as the initial transformation for the next level in the optimization.
- 6** Similar processing as the previous two levels except the images are, first, sub-sampled and interpolated to 2 times.
- 7** Finally, 1mm resolution images are used. The algorithm returns the value of the rotation angle that was applied to register the original to flipped image.
- 8** The algorithm transforms the original image back by half of the registration rotation. This finds the estimated midsagittal plane.
- 9** The estimated plane is aligned with the center of the image lattice.

#### For more information of which methods were used to

- Calculate the Center of Mass;
- Subsample and interpolate the images;
- Find the global minima and use it in evaluation of the cost function;

Refer to “Optimized Automatic Registration 3D” on page 596.

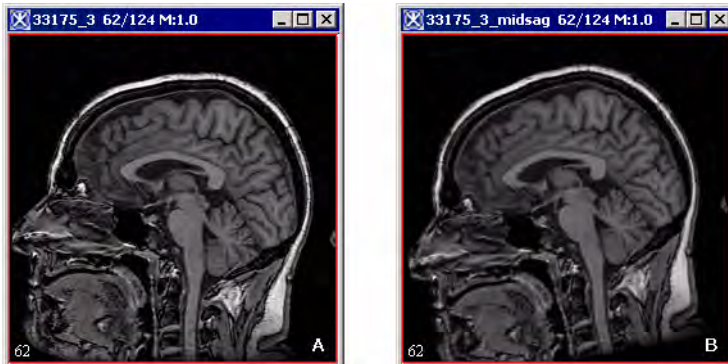


Figure 2. The original image (A) and the image after applying the midsagittal line alignment algorithm (B).

---

## IMAGE TYPES

2D and 3D grayscale and color brain images

---

## RUNNING THE MIDSAGITTAL LINE ALIGNMENT ALGORITHM

To run the algorithm,

- 1 Open an image of interest;
- 2 Call Algorithms>Brain Tools>Midsagittal line alignment.

The algorithm begins to run and several windows appear with the status as shown in Figure 3. When the algorithm finishes running, the aligned image appears in a new image frame as shown in Figure 2-B.

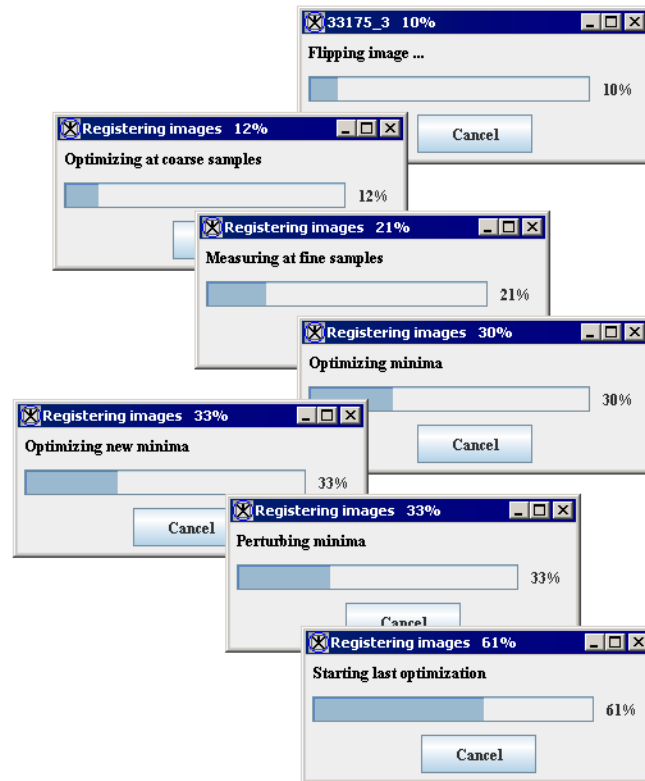


Figure 3. The Midsagittal Line Alignment algorithm is running.

## Morphology

*The following sections present several important algorithms that employ the mathematical morphological operations such as erosion, dilation, opening, closing, particle analysis, calculation the background map, etc.*

### Introduction to the basic morphological operations

In this introductory section we will explain several important concepts in mathematical morphology that are used in MIPAV algorithms. These concepts are as follows: translation, reflection, complement, and difference.

Let's introduce two objects A and B in  $Z^2$  with components  $a = (a_i, a_j)$  and  $b = (b_i, b_j)$  respectively.

The translation of A by  $x = (x_1, x_2)$ , which is denoted as  $(A)_x$ , can be defined as

EQUATION 1

$$(A)_x = \{c \uparrow c = a + x, \text{ for } a \in A\}$$

The reflection of B, denoted as  $B^\wedge$ , is defined as

EQUATION 2

$$\hat{B} = \{x \uparrow x = -b, \text{ for } b \in B\}$$

The complement of object A is

EQUATION 3

$$A^c = \{x \uparrow x \notin A\}$$

The difference of two objects or images A and B, denoted as  $A-B$ , is defined as

$$A - B = \{x \uparrow x \in A, x \notin B\} = A \cap B^c$$

Figure 1 illustrates the morphological operations mentioned above.

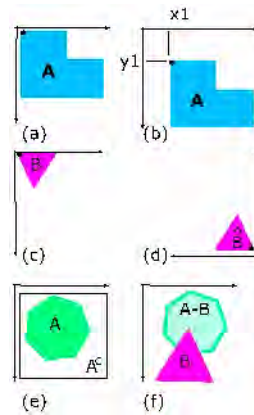


Figure 1. (a) Image A; (b) image A translated; (c) image B; (d) reflection of image B; (e) image A and its complement; (f) the difference of A and B. The origin of the image is shown as a black dot.

## DATA TYPES

All morphological operations can be applied to the images of the following types:

Data type	Description
Boolean	1 bit per pixel/voxel (1 on, 0 off)
Unsigned byte	1 byte per pixel/voxel (0, 255)
Unsigned short	2 bytes per pixel/voxel (0, 65535)

## Background Distance map

The Background Distance map operation converts a binary image (which consists of background and foreground pixels) into an image where every foreground pixel has a value corresponding to the minimum distance from

the background. The algorithm uses the Euclidean distance metric to transform the image.

For 2D images, the algorithm determines the image resolution in X and Y directions, first. Then it identifies all edges pixels. And finally, it creates a distance metric, where each pixel of the metric is associated with a corresponding region of the source image. Each pixel is assigned a calculated distance value corresponding to the Euclidean distance between a center of that pixel and the nearest point of the background. The nearest point is located using the image resolution.

For two 2D points P(x<sub>1</sub>,y<sub>1</sub>) and Q(x<sub>2</sub>,y<sub>2</sub>) the “classical” Euclidian distance is computer as

EQUATION 5

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

The Background Distance map operation when computes the Euclidian distance takes into account the image resolutions in the X and Y dimensions. So that it is calculated as

EQUATION 6

$$\sqrt{(x_2 - x_1)^2 * x_{resolution}^2 + (y_2 - y_1)^2 * y_{resolution}^2}$$

---

## APPLYING BACKGROUND DISTANCE MAP

To apply the algorithm,

- 1** Open an image of interest.
- 2** Call Algorithms>Morphological> Background Distance map.
- 3** In the Background Distance map dialog box that appears,
  - Specify the destination of the Background Distance map image;



- And also specify whether you want to calculate the Background Distance map for the whole image or only for the selected VOIs.

**4** Press OK to run the algorithm.

When the algorithm finishes running, the Background Distance map appears in the specified frame (the default option is a new frame). See also Figure 2.

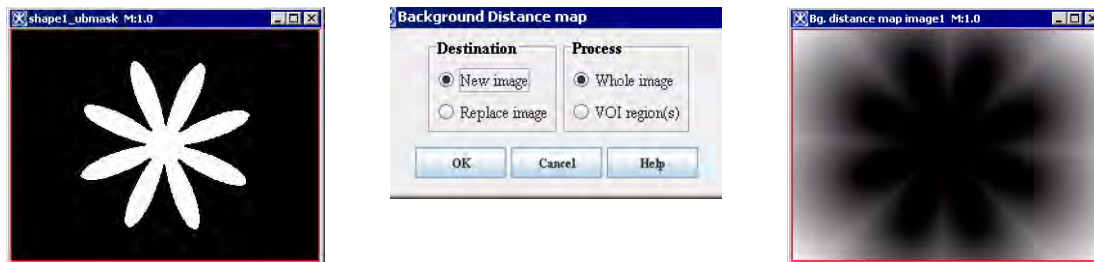


Figure 2. Applying the Background Distance map operation to the image. You can see 1) the original image (left), 2) the dialog box with the default settings (center) and 3) the background distance map (right)

## Close

The closing of an object A by kernel B is defined as the dilation of A by B, followed by the erosion of the result by B. Closing generally smooths sections of contours, fuses narrow breaks, eliminates small holes and fills gaps in the contour. For 2D images it could be expressed using the following equation:

EQUATION 7

$$A \bullet B = (A \oplus B) \ominus B$$

Figure 3 below illustrates closing of an object A with a disk kernel B. Note that the resulted image was also smoothed by the circular kernel B.

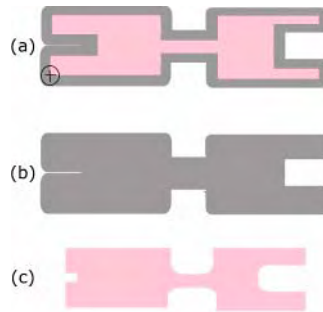


Figure 3. Illustration of closing (a) the original image A, (b) the dilation of image A by kernel B, (c) the erosion of result of (b) by the same kernel B

---

## APPLYING THE CLOSING

To apply the algorithm,

- 1** Open an image of interest.
- 2** Call Algorithms>Morphological >Close.
- 3** In the Open dialog box that appears,
  - Specify the number of dilations (from 1 to 20);
  - Specify the number of erosions (from 1 to 20);
  - Select the structural element or kernel;
  - Specify the destination of the result image;
  - And also specify whether you want apply the closing to the whole image or only for selected VOIs. For the dialog box options refer to Figure 4.
- 4** Press OK to run the algorithm.

When the algorithm finishes running, the result image appears in the specified frame. See also Figure 5.

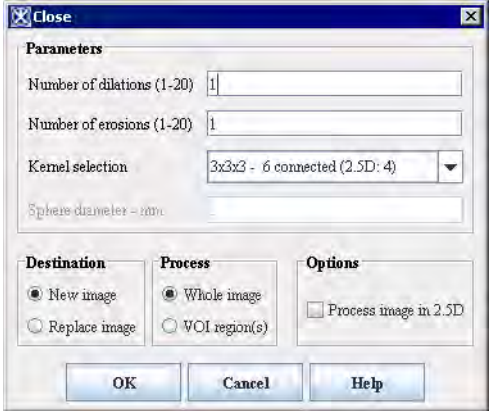
<b>Number of dilations</b>	Indicates the number of dilations to be performed.	
<b>Number of erosions</b>	Indicates the number of erosions to be performed.	
<b>Kernel selection</b>	Use this list box to select the structuring element that will be used for erosion.	
<b>Circle diameter (mm)</b>	<p>This option is available only if you choose the User Sized Circle option in the Kernel Selection box.</p> <p>Enter a value for the circle diameter, here.</p>	
<b>Process image in 2.5D</b>		TBD.
<b>Destination</b>		
<p><b>New image</b> – if selected, indicates that the result image appears in a new image window.</p> <p><b>Replace image</b> – if selected, indicates that the result image replaces the current active image.</p>		
<b>Process</b>		
<p><b>Whole image</b></p> <p>If selected, indicates that the erosion should be applied to the whole image.</p> <p><b>VOI region(s)</b></p> <p>If selected, indicates that the erosion should be applied only to selected VOI(s).</p>		
<b>OK</b>	Performs the dilation on the selected image based on your choices in this dialog box.	
<b>Cancel</b>	Disregards changes you made in this dialog box and closes the dialog box.	
<b>Help</b>	Displays online help for the algorithm.	

Figure 4. Close dialog box

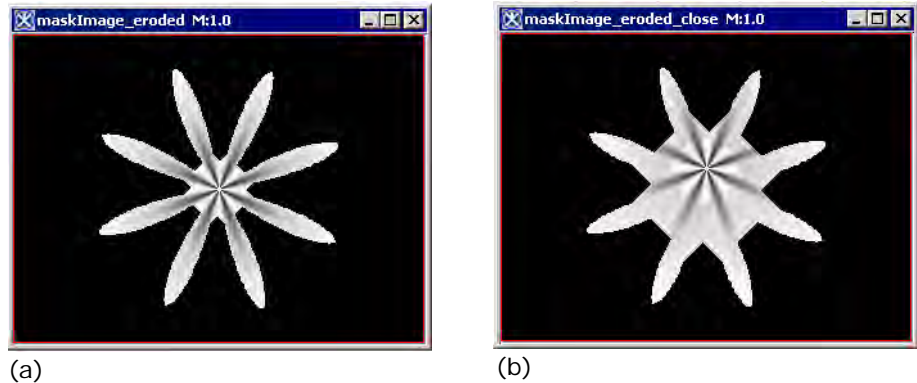


Figure 5. The original image (a) and the result image after the closing that includes 15 dilations followed by 15 erosions applied with the “3x3 -4 connected” kernel

## Delete Objects

The Delete operation deletes objects larger than a maximum size indicated in the Delete dialog box and also objects smaller than the indicated minimum size.

---

### APPLYING THE ALGORITHM

To apply the algorithm,

- 1** Open an image of interest.
- 2** Call Algorithms>Morphological> Delete Objects.
- 3** In the Delete Objects dialog box that appears,
  - Specify the maximum size of the object;
  - Specify the minimum size of the object;
  - Specify the destination of the result image;
  - And also specify whether you want apply the algorithm to the whole image or only for selected VOIs.
- 4** Press OK to run the algorithm.

When the algorithm finishes running, the result image appears in the specified frame. See also Figure 6.

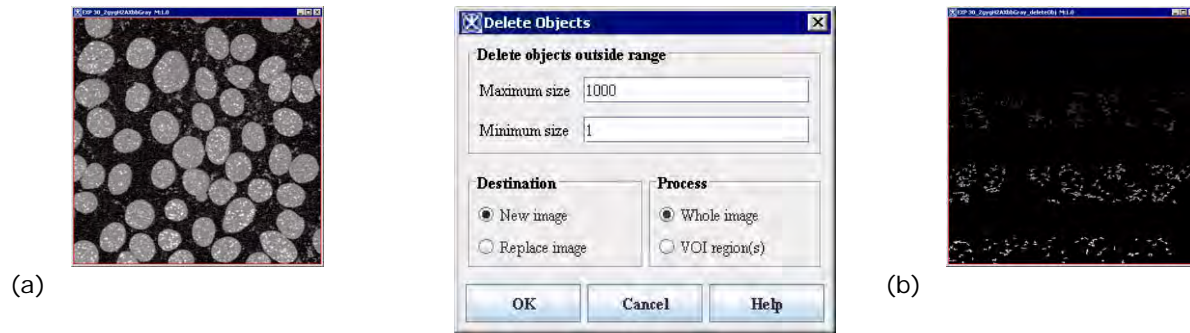


Figure 6. Applying the Delete Objects operation; (a) the original image, (b) the result image. The parameters are shown in the dialog box

## Dilate

We define the dilation as a process that consists of obtaining the reflection of B about its origin, and then shifting this reflection by x. The dilation of A by B is then the set of all x displacements such that  $\hat{B}$  and A overlap by *at least one nonzero element*. That can be expressed using the following equation:

EQUATION 8

$$A \oplus B = \{x \uparrow [(\hat{B})_x \cap A] \subseteq A\}$$

The object B is commonly referred to as *a structuring element* or *kernel* in dilation, as well as in the other morphological operations. Figure 7 (a) below shows a simple dilation by the symmetric kernel, so that  $\hat{B}=B$ . Figure 7 (b) shows a kernel designed to achieve more dilation vertically than horizontally.

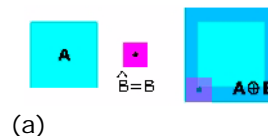


Figure 7. (a) An original image A, a square structural element B and its reflection, and the dilation of A by B; (b) an original image A, an elongated structural element B and its reflection, and the dilation of A by B

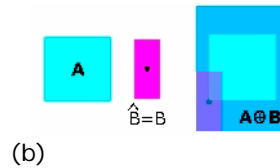


Figure 7. (a) An original image  $A$ , a square structural element  $B$  and its reflection, and the dilation of  $A$  by  $B$ ; (b) an original image  $A$ , an elongated structural element  $B$  and its reflection, and the dilation of  $A$  by  $B$

## APPLYING THE DILATION

To apply the algorithm,

- 1** Open an image of interest.
- 2** Call Algorithms>Morphological> Dilate.
- 3** In the Dilation dialog box that appears,
  - Specify the number of dilations (from 1 to 20);
  - Select the structural element or kernel;
  - Specify the destination of the dilated image;
  - And also specify whether you want apply the dilation to the whole image or only for selected VOIs. For the dialog box options refer to Figure 8.
- 4** Press OK to run the algorithm.

When the algorithm finishes running, the result image appears in the specified frame. See also Figure 9.

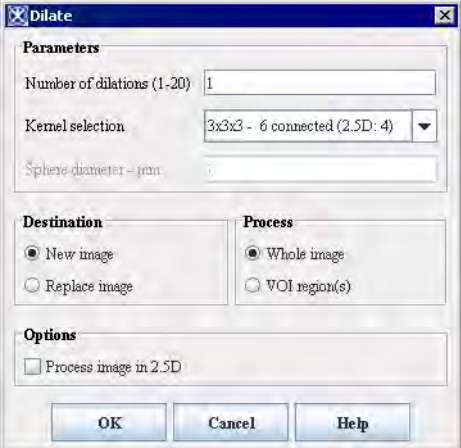
<b>Number of dilations</b>	Indicates the number of dilations to be performed.	
<b>Kernel selection</b>	Use this list box to select the structuring element that will be used for dilation.	
<b>Circle diameter (mm)</b>	This option is available only if you choose the User Sized Circle option in the Kernel Selection box.  Enter a value for the circle diameter, here.	
<b>Destination</b>		
<b>New image</b>		
If selected, indicates that the result image appears in a new image window.		
<b>Replace image</b>		
If selected, indicates that the result image replaces the current active image.		
<b>Process</b>		
<b>Whole image</b>		
If selected, indicates that the dilation should be applied to the whole image.		
<b>VOI region(s)</b>		
If selected, indicates that the dilation should be applied only to selected VOI(s).		
<b>OK</b>	Performs the dilation on the selected image based on your choices in this dialog box.	
<b>Cancel</b>	Disregards changes you made in this dialog box and closes the dialog box.	
<b>Help</b>	Displays online help for the algorithm.	

Figure 8. Dilation dialog box

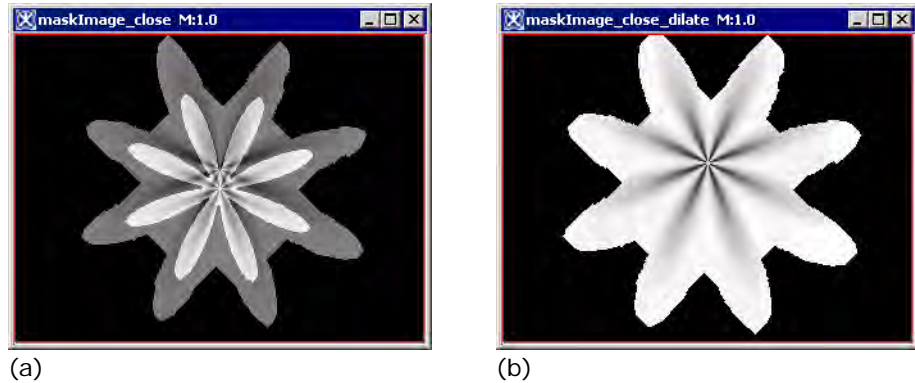


Figure 9. The original image (a) and the result image after 5 dilations applied with the “3x3–4 connected” kernel

## Distance Map

The algorithm uses the Euclidean distance metric to calculate a distance map for a selected image or image region. For 2D images,

- 1** The algorithm determines the image resolution in X and Y directions.
- 2** Then it identifies all edge pixels.
- 3** Then, it creates a distance metric, where each pixel of the metric is associated with a corresponding region of the source image. Each pixel is assigned a calculated distance value corresponding to the Euclidean distance between a center of that pixel and the nearest edge point.

For more information, please refer to “Background Distance map” on page 519.

---

## APPLYING DISTANCE MAP

To apply the algorithm,

- 1** Open an image of interest.
- 2** Call Algorithms>Morphological> Distance Map.
- 3** Specify whether you want apply the dilation to the whole image or only for selected VOIs.



**4** Press OK to run the algorithm.

When the algorithm finishes running, the result image appears in the specified frame. See also Figure 9.

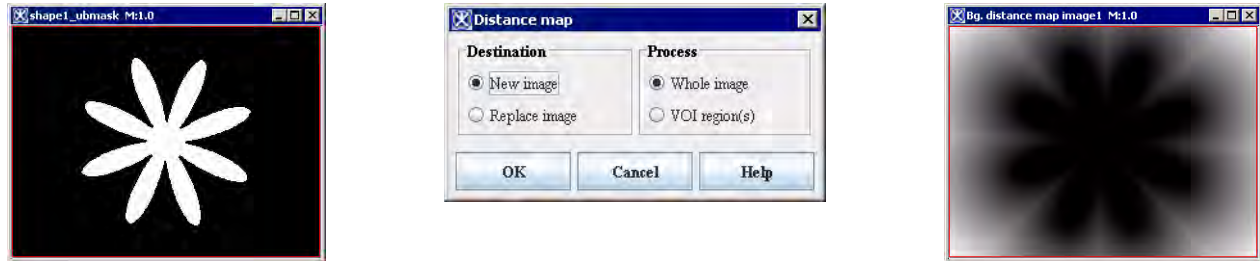


Figure 10. Applying the Background Distance map operation to the image. You can see 1) the original image (left), 2) the dialog box with the default settings (center) and 3) the background distance map (right)

## Erode

The erosion of A by B is the set of all points x such that B, translated by x, is contained in A. For 2D images it could be expressed using the following equation:

EQUATION 9

$$A \ominus B = \{x \uparrow (B)_x \subseteq A\}$$

## APPLYING THE EROSION

To apply the algorithm,

- 1** Open an image of interest.
- 2** Call Algorithms>Morphological> Erode.
- 3** In the Erode dialog box that appears,
  - Specify the number of erosions (from 1 to 20);
  - Select the structural element or kernel;
  - Specify the destination of the result image;

- And also specify whether you want apply the erosion to the whole image or only for selected VOIs. For the dialog box options refer to Figure 11.

4 Press OK to run the algorithm.

When the algorithm finishes running, the result image appears in the specified frame. See also Figure 12.

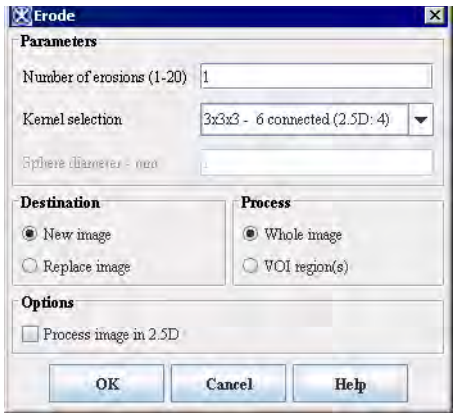
<b>Number of erosions</b>	Indicates the number of erosions to be performed.	
<b>Kernel selection</b>	Use this list box to select the structuring element that will be used for erosion.	
<b>Circle diameter (mm)</b>	This option is available only if you choose the User Sized Circle option in the Kernel Selection box.  Enter a value for the circle diameter, here.	
<b>Destination</b>		
<b>New image</b>	If selected, indicates that the result image appears in a new image window.	
<b>Replace image</b>	If selected, indicates that the result image replaces the current active image.	
<b>Process</b>		
<b>Whole image</b>	If selected, indicates that the erosion should be applied to the whole image.	
<b>VOI region(s)</b>	If selected, indicates that the erosion should be applied only to selected VOI(s).	
<b>Process image in 2.5D</b>	TBD.	
<b>OK</b>	Performs the dilation on the selected image based on your choices in this dialog box.	
<b>Cancel</b>	Disregards changes you made in this dialog box and closes the dialog box.	
<b>Help</b>	Displays online help for the algorithm.	

Figure 11. Erosion dialog box

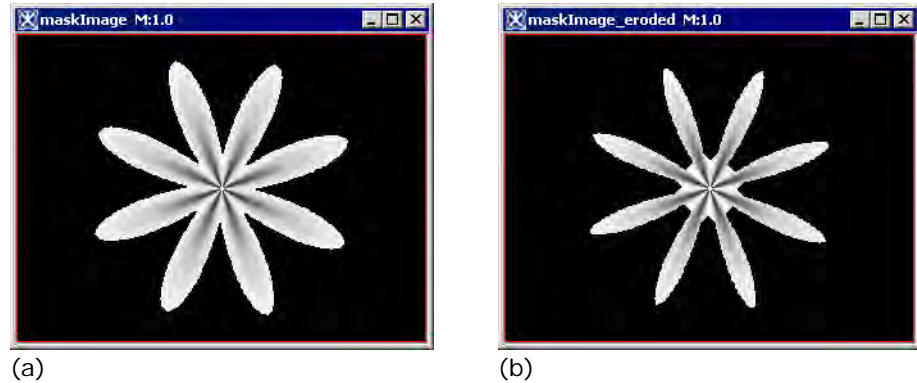


Figure 12. The original image (a) and the result image after 5 erosions applied with the “3x3 -4 connected” kernel

## Evaluate Segmentation

Evaluate Segmentation operation compares segmentation results of a test image to segmentation results of an ideal gold standard true image. For each evaluated segmentation pair, the false negative volume fraction, the false positive volume fraction, and the positive volume fraction are sent to the Output window. See Figure 13-e.

---

### APPLYING EVALUATE SEGMENTATION

To apply the algorithm,

- 1** Open a segmented image of interest and its gold standard segmented image.

---

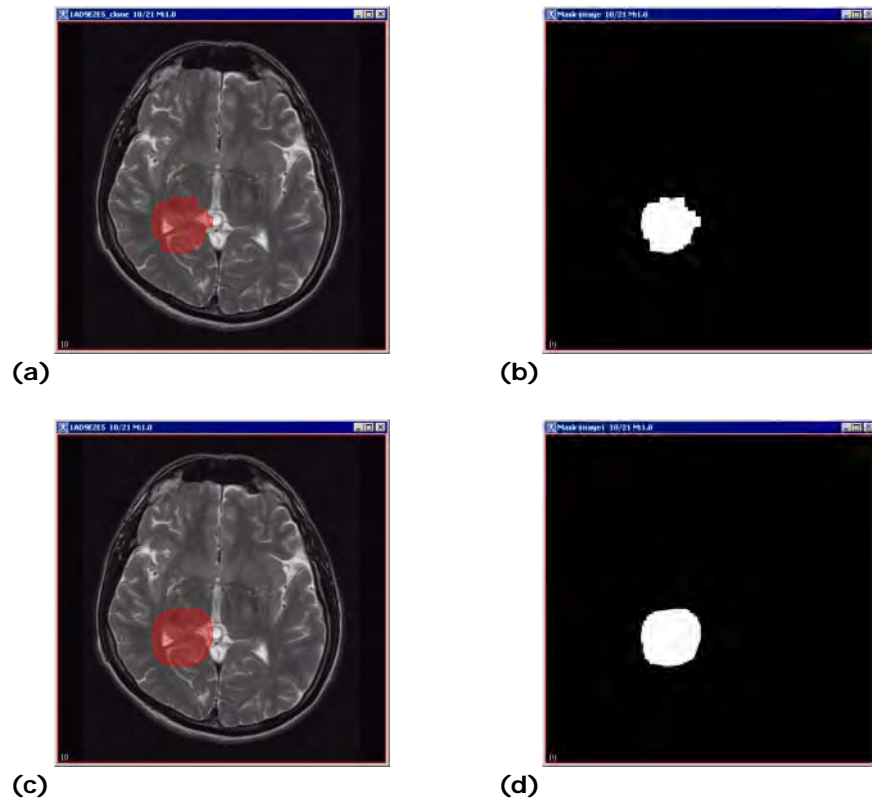
Note that the images should be of the following types: Boolean, Unsigned byte, or Short. Before running the algorithm, convert the images using the Utilities> Conversion Tools> Convert Type menu. See also “Convert Type” on page 365. You can also use the Paint Conversion tools menu options to convert the paint to the Unsigned Byte mask image. See Figure 13.

---

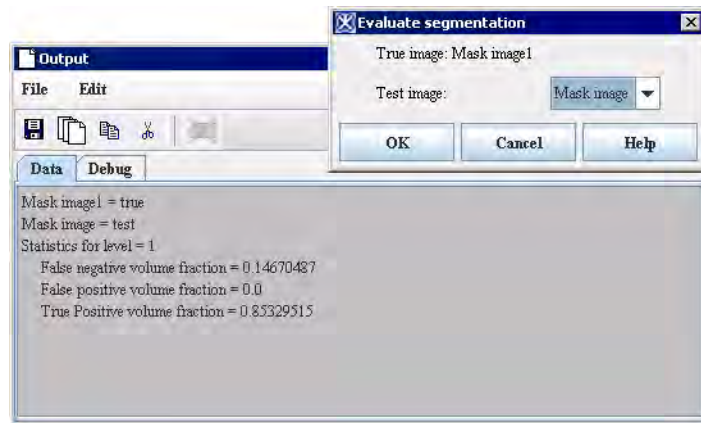
- 2** Select the gold standard segmented image.
- 3** Call Algorithms>Morphological> Evaluate Segmentation.
- 4** The Evaluate Segmentation dialog box appears.

**5** In the dialog box, select the test image, and then press OK to run the algorithm.

When the algorithm finishes running, the result statistics appears in the Output window. See also Figure 13.



**Figure 13. The Evaluate Segmentation algorithm : (a) – a gold standard segmented image, (b) – its unsigned byte mask, (c) – a segmented image for evaluation, (d) – its unsigned byte mask, and (d) – the Evaluate Segmentation dialog box and the Output window**



(e)

Figure 13. The Evaluate Segmentation algorithm (continued): (a) – a gold standard segmented image, (b) – its unsigned byte mask, (c) – a segmented image for evaluation, (d) – its unsigned byte mask, and (e) – the Evaluate Segmentation dialog box and the Output window

## Fill holes

To fill holes in objects MIPAV uses the following procedure:

- It segments an image to produce binary representation of objects;
- Then, it computes complement of a binary image as a mask image;
- It generates a seed image as the border of the image;
- Then, it propagates the seed into the mask;
- And finally, it complements the result of propagation to produce the final result.

## APPLYING FILL HOLES

To apply the algorithm,

- 1 Open an image of interest.
- 2 Call Algorithms>Morphological> Fill Holes.
- 3 The Fill Objects dialog box appears. In the dialog box:
  - Specify if you want to process the image in 2.5 D (this option is available for 3D images only);

- Specify the destination of the result image.

4 Press OK to run the algorithm.

When the algorithm finishes running, the result image appears in the new image frame. See also Figure 14.



Figure 14. The Fill Holes algorithm: (a) the original image; (b) the Fill Objects dialog box; (c) the result image

## Find Edges

The technique finds the edges of the objects in an image using combinations of the following morphological operations: dilation, erosion, and XOR. The algorithm has two options – it could find *outer edges* or *inner edges* depending on the user's choice. The algorithm, first, determines the image resolution. And then it uses the resolution to calculate the kernel or structural element that will be used for the dilation or erosion.

**If the Outer Edging option was selected**, the algorithm dilates the original image, and then performs XOR operation of the dilated image with the original image.

**If the Inner Edging option was selected**, the algorithm erodes the original image, and then performs XOR operation of the eroded image with the original image.

## APPLYING FIND EDGES

To apply the algorithm,

- 1 Open an image of interest.
- 2 Call Algorithms>Morphological> Find Edges.
- 3 The Find Edges dialog box appears. In the dialog box:
  - Specify what option you would like to use – Outer Edging or Inner Edging;
  - Specify the destination of the result image;
  - And also specify whether you want apply the algorithm to the whole image or only for selected VOI(s).
- 4 Press OK to run the algorithm.

When the algorithm finishes running, the result image appears in the new image frame. See also Figure 15.

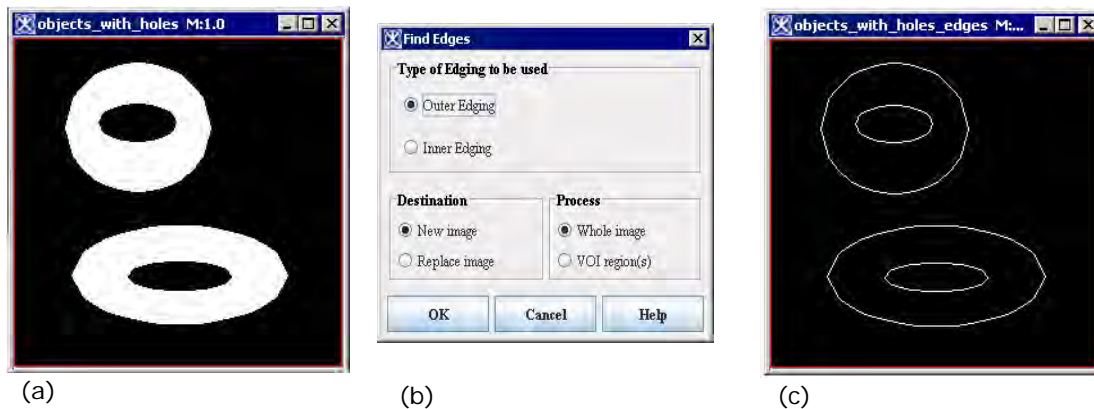


Figure 15. The Find Edges algorithm: (a) the original image; (b) the Find Edges dialog box; (c) the result image

## ID objects

The algorithm labels each object in an image with a different integer value and also deletes objects which are outside of the user defined threshold. It converts the image to black and white in order to prepare it for boundary tracing; and then it uses morphology functions, such as open, close and fill holes to remove pixels which do not belong to the objects of interest.

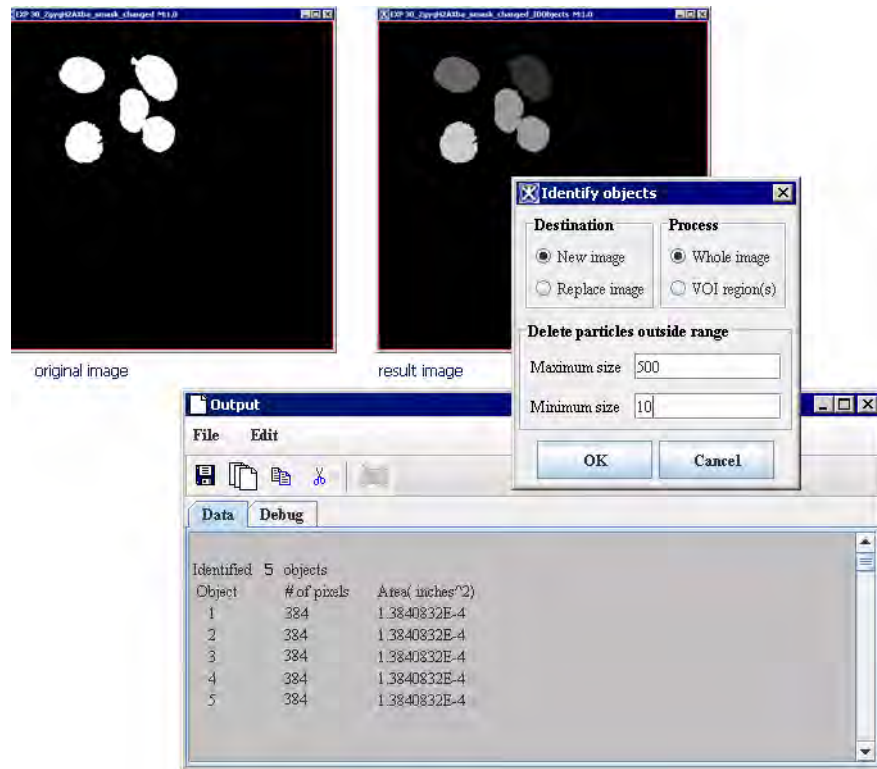


Figure 16. ID Objects algorithm

## APPLYING ID OBJECTS

To apply the algorithm,

- 1 Open an image of interest.
- 2 Call Algorithms>Morphological> ID objects.
- 3 The Identify Objects dialog box appears. In the dialog box:
  - Specify the destination of the result image;
  - Specify whether you want apply the algorithm to the whole image or only to selected VOI(s);
  - And also enter the max and min size for the particles you would like to exclude from the calculation.
- 4 Press OK to run the algorithm.

When the algorithm finishes running, the result image appears in the



designated image frame and the statistics appears in the Output window. See also Figure 16.

---

Note: you might consider smoothing the image and removing excess noise, first, and then apply ID Objects.

---

## Morphological filter

Morphological filter can be used to correct an image for the shading that was introduced in the image forming procedure. In other words, it corrects for non-uniform illumination and non-uniform camera sensitivity.

The illumination over the microscope field-of-view  $I_{ill}(x,y)$  usually interacts in a multiplicative (non-uniform) way with the biological object  $a(x,y)$  to produce the image  $b(x,y)$ :

EQUATION 10

$$b(x,y) = I_{ill}(x,y) \cdot a(x,y),$$

where the object  $a(x,y)$  representing various microscope imaging modalities, such as reflectable model  $r(x,y)$ , optical density model  $10^{-OD(x,y)}$  or fluorescence model  $c(x,y)$ . The Morphological filter works with the fluorescence model  $c(x,y)$ , assuming that this model only holds for low concentrations. The camera and microscope are also contributing *gain* and *offset* to the image, thus the equation for  $c(x,y)$  can be rewritten as:

EQUATION 11

$$c[m,n] = gain[m,n] \cdot b[m,n] + offset[m,n] = gain[m,n] \cdot I_{ill}[m,n] \cdot a[m,n] + offset[m,n]$$

where, a) the camera *gain* and *offset* can vary as a function of position and, therefore, contributing to the shading; and b)  $I_{ill}[m,n]$ ,  $gain[m,n]$  and  $offset[m,n]$  are slowly varying compared to  $a[m,n]$ .

The algorithm, first, compute (using morphological smoothing) a smoothed version of  $c[m,n]$ , where the smoothing is large compared to the size of the objects in the image. This smoothed version is the estimate of the background of the image. Second, it subtracts the smoothed version from  $c[m,n]$ . And then, it restores the desired average brightness value. This can

be written as a following formula:

EQUATION 12

$$\hat{a}[m,n] = c[m,n] - \text{MorphSmooth}\{c[m,n]\} + \text{constant},$$

where the morphological smoothing filter  $\text{MorphSmooth}\{c[m,n]\}$  involves two basic operations – the *maximum* filter and the *minimum* filter.

- In the *maximum* filter, defined over a window  $W$  of  $J \times K$  pixels where both  $J$  and  $K$  are considered to be of odd size (e.g.,  $5 \times 3$ ), the value in the output image, corresponding to the center pixel  $A$  in the input window, is the *maximum* brightness value found in the input window.
- In the *minimum* filter, defined over a similar  $J \times K$  window ( $W$ ), the value in the output image, corresponding to the center pixel  $A$  in the input window, is the *minimum* brightness value found in the input window.

Therefore, the *maximum* and *minimum* filters can be described as examples of the morphological *dilation* and *erosion* respectively.

Dilation:

EQUATION 13

$$D(A) = \max_{[j,k] \in W} \{a[m-j, n-k]\} = \max_W(A)$$

Erosion:

EQUATION 14

$$E(A) = \min_{[j,k] \in W} \{a[m-j, n-k]\} = \min_W(A)$$

Finally, the algorithm defines the morphological smoothing filter described as follows:

EQUATION 15

$$\text{MorphSmooth}(A) = \min(\max(\max(\min(A))))$$

where all of the filter operations are applied over the same  $J \times K$  filter window  $W$ . To better understand Equation 15, assume that you have two “subroutines” available, one for the *minimum* filter and the other one for the *maximum* filter, and that you apply them in a sequential fashion. See

Figure 17.

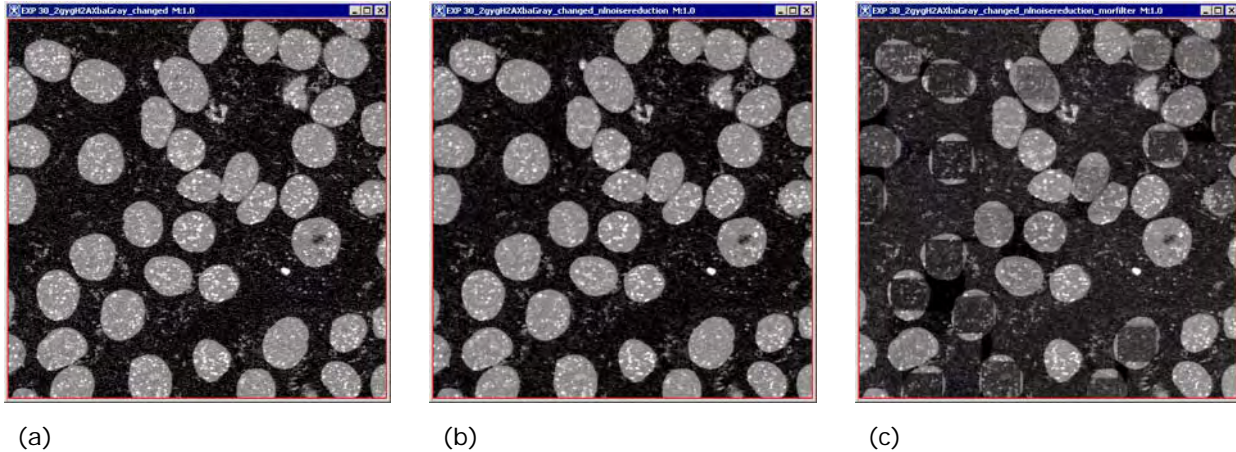


Figure 17. a: the original image; b: the same image after applying the noise reduction algorithm; c: the image from (b) after applying Morphological filter with parameters shown in Figure 18

## APPLYING MORPHOLOGICAL FILTER

Before applying Morphological filter you might consider smoothing the image using Gaussian Blur and (or) reducing the image noise using one of the noise reduction algorithms available in MIPAV.

To apply the algorithm,

- 1** Open an image of interest.
- 2** Call Algorithms>Morphological >Morphological Filter.
- 3** In the Morphological Filter dialog box that appears,
  - Specify the window size in the X direction;
  - Specify the window size in the Y direction;
  - Specify the window size in the Z direction (for 3D images only);
  - For 2.5D images, specify whether you want to process slices independently or not;
  - Specify the destination of the result image;

- And also specify whether you want apply the opening to the whole image or only for selected VOIs. For the dialog box options refer to Figure 18 on page 540.

4 Press OK to run the algorithm.

When the algorithm finishes running, the result image appears in the specified frame. See also Figure 17.

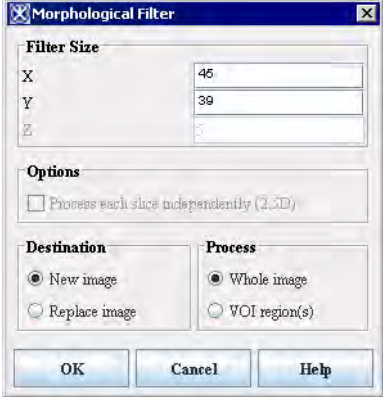
<b>Filter size</b>	Specify the size of the filter window W in X, Y and Z (for 3D images) direction. See also Equation 13 and Equation 14.	
<b>Options</b>	<i>Process each slice independently</i> – check this check box if you want to process slices independently. this option only works for 2.5D images.	
<b>Destination</b>		
<p><b>New image</b> – if selected, indicates that the result image appears in a new image window.</p> <p><b>Replace image</b> – if selected, indicates that the result image replaces the current active image.</p>		
<b>Process</b>		
<p><b>Whole image</b> If selected, indicates that the algorithm should be applied to the whole image.</p> <p><b>VOI region(s)</b> If selected, indicates that the algorithm should be applied only to selected VOI(s).</p>		
<b>OK</b>	Performs the dilation on the selected image based on your choices in this dialog box.	
<b>Cancel</b>	Disregards changes you made in this dialog box and closes the dialog box.	
<b>Help</b>	Displays online help for the algorithm.	

Figure 18. Morphological Filter dialog box

## Open

The opening of image A by a structuring element (a kernel) B is simply a combination of two operations: an erosion of A by B, followed by a dilation

of the result by B. See also Figure 19.

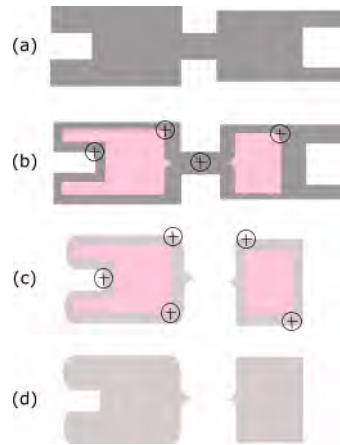


Figure 19. Illustration of opening (a) the original image A, (b) the erosion of image A by kernel B, (c) the dilatation of result of (b) by the same kernel B, (d) image A after opening

Opening generally smooths the contour of an image, breaks narrow strips and eliminates thin protrusions. The equation for the opening is as follows:

EQUATION 16

$$A \circ B = (A \ominus B) \oplus B$$

## APPLYING THE OPENING

To apply the algorithm,

- 1** Open an image of interest.
- 2** Call Algorithms>Morphological >Open.
- 3** In the Open dialog box that appears,
  - Specify the number of erosions (from 1 to 20);
  - Specify the number of dilations (from 1 to 20);
  - Select the structural element or kernel;
  - Specify the destination of the result image;

- And also specify whether you want apply the opening to the whole image or only for selected VOIs. For the dialog box options refer to Figure 20.

4 Press OK to run the algorithm.

When the algorithm finishes running, the result image appears in the specified frame. See also Figure 21.

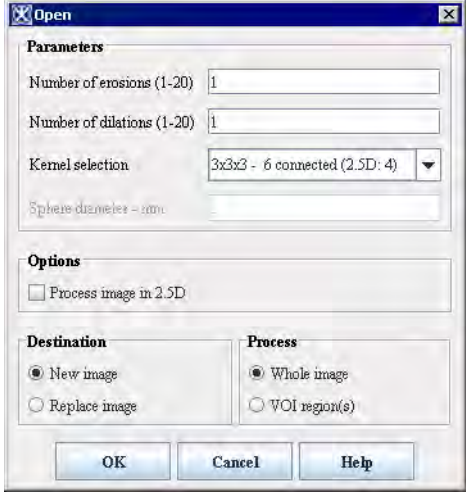
<b>Number of erosions</b>	Indicates the number of erosions to be performed.	
<b>Number of dilations</b>	Indicates the number of dilations to be performed.	
<b>Kernel selection</b>	Use this list box to select the structuring element that will be used for erosion.	
<b>Circle diameter (mm)</b>	This option is available only if you choose the User Sized Circle option in the Kernel Selection box.  Enter a value for the circle diameter, here.	
<b>Process image in 2.5D</b>	TBD.	
<b>Destination</b>		
<p><b>New image</b> – if selected, indicates that the result image appears in a new image window.</p> <p><b>Replace image</b> – if selected, indicates that the result image replaces the current active image.</p>		
<b>Process</b>		
<b>Whole image</b>		
If selected, indicates that the algorithm should be applied to the whole image.		
<b>VOI region(s)</b>		
If selected, indicates that the algorithm should be applied only to selected VOI(s).		
<b>OK</b>	Performs the dilation on the selected image based on your choices in this dialog box.	
<b>Cancel</b>	Disregards changes you made in this dialog box and closes the dialog box.	
<b>Help</b>	Displays online help for the algorithm.	

Figure 20. Open dialog box

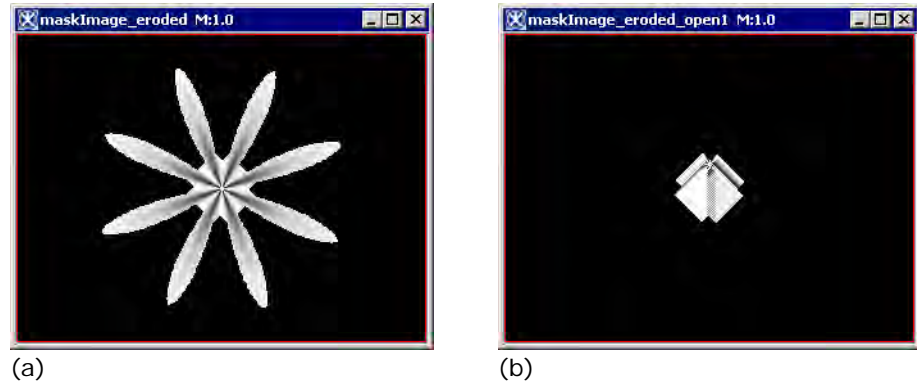
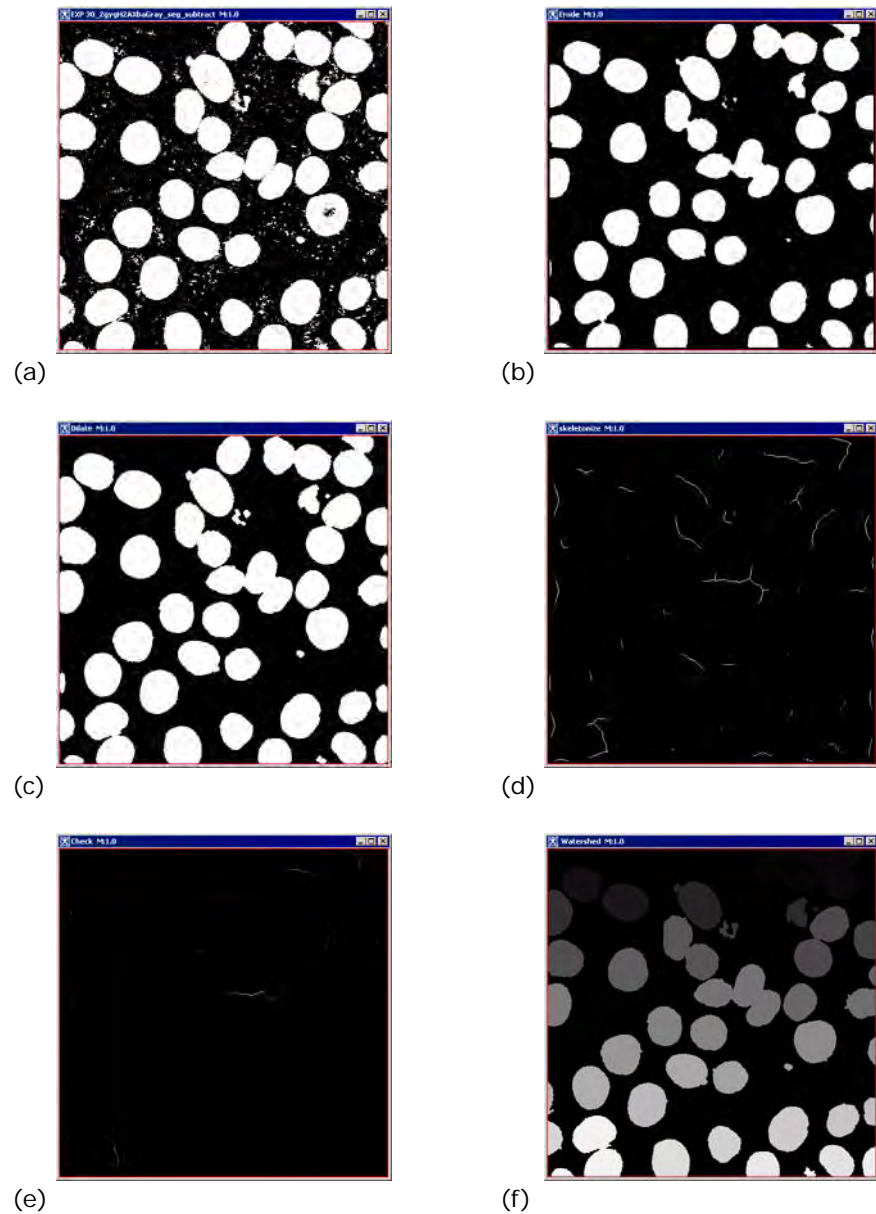


Figure 21. The original image (a) and the result image after the opening that includes 19 erosions followed by 19 dilations applied with the “3x3 -4 connected” kernel

## Particle analysis

This method combines several morphological operations to produce the information on the particle composition for the binary images.

- 1** It performs a number of openings and closings with the user defined structural element. This allows to separate the foreground elements from the background.
- 2** Then, it applies to the result skeletonize following by pruning.
- 3** To separate slightly overlapping particles, the method then uses watershed segmentation, which is as follows:
  - In the binary image, the black pixels are replaced with grey pixels of an intensity proportional to their distance from a white pixel (i.e. black pixels close to the edge are light grey, those closer to the middle are nearer black);
  - This forms the Euclidian distance map (EDM) for the image;
  - From this, it calculates the centres of the objects or *the ultimate eroded points* (UEPs), i.e. points that are equidistant from the edges. These points are then dilated until they meet another black pixel, then a watershed line is drawn. See also “Ultimate erode” on page 556.
- 4** The result statistics appears in the Output MIPAV window. See also Figure 22.



**Figure 22. Particle Analysis: (a) – an original binary image. The following steps were performed: (b) erosion; (c) dilation; (d) skeletonize; (e) prune; (f) watershed; (g) – the result image; (h) – particle statistics as it appears in the Output window**





(g)

Object	# of pixels	Area( inches^2)
1	2446	0.33854672
2	4758	0.65854875
3	5728	0.7928028
4	2853	0.3948789
5	3847	0.53245676
6	3327	0.46048445
7	2253	0.31183392
8	1125	0.15570934
n	SUM	0.74878895

(h)

**Figure 22. Particle Analysis: (a) – an original binary image. The following steps were performed: (b) erosion; (c) dilation; (d) skeletonize; (e) prune; (f) watershed; (g) – the result image; (h) – particle statistics as it appears in the Output window (continued)**

## IMAGE TYPES

Particle analysis can be applied only to binary images.

## APPLYING PARTICLE ANALYSIS

To apply the algorithm,

- 1** Open an image of interest.
- 2** Call Algorithms>Morphological >Particle analysis.
- 3** In the Particle Analysis New dialog box that appears,

- Specify the number of openings (from 1 to 20);
- For opening(s), also specify the structural element or kernel;
- Specify the number of closings (from 1 to 20);
- For closing(s), also specify the structural element or kernel;
- Specify the destination of the result image;
- And also specify whether you want apply the opening to the whole image or only for selected VOIs. For the dialog box options refer to Figure 23.

**4** Check the Show Intermediate Result Frames box, to view intermediate images.

**5** Press OK to run the algorithm.

When the algorithm finishes running, the particle statistics appears in the Output window and the result image appears in the specified frame. See also Figure 22.

<b>Number of open</b>	Indicates the number of openings to be performed.
<b>Kernel selection</b>	Use this list box to select the structuring element that will be used for opening.
<b>Number of close</b>	Indicates the number of closings to be performed.
<b>Kernel selection</b>	Use this list box to select the structuring element that will be used for closing.
<b>Circle diameter (mm) for openings and closings</b>	This option is available only if you choose the User Sized Circle option in the Kernel Selection box.  Enter a value for the kernel diameter, here.
<b>Destination</b>	
<b>New image</b>	– if selected, indicates that the result image appears in a new image window.
<b>Replace image</b>	– if selected, indicates that the result image replaces the current active image.
<b>Process</b>	

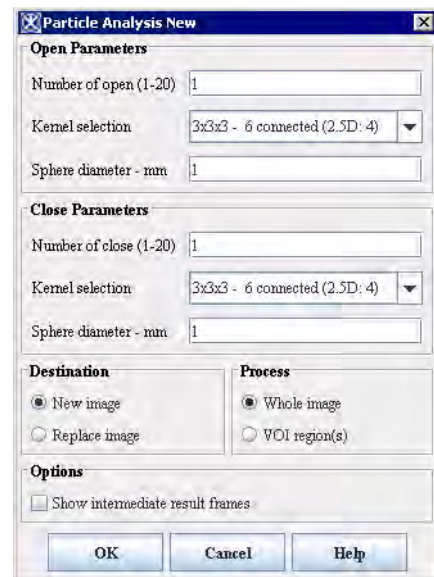


Figure 23. Particle Analysis New dialog box

**Whole image**

If selected, indicates that the erosion should be applied to the whole image.

**VOI region(s)**

If selected, indicates that the erosion should be applied only to selected VOI(s).

<b>Show Intermediate Result Frames</b>	Check to view intermediate images.
<b>OK</b>	Performs the dilation on the selected image based on your choices in this dialog box.
<b>Cancel</b>	Disregards changes you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for the algorithm.

Figure 23. Particle Analysis New dialog box (continued)

## Skeletonize

Skeletonize operation skeletonizes the image by means of a lookup table, which is used to repeatedly remove pixels from the edges of objects in a binary image, reducing them to single pixel wide skeletons. The method is based on a thinning algorithm by Zhang and Suen.

The lookup table shown below has an entry for each of the 256 possible 3x3 neighborhood configurations. An entry of '1' signifies to delete the indicated pixel on the first pass, '2' means to delete the indicated pixel on the second pass, and '3' indicates to delete the pixel on either pass.

The lookup table
0, 0, 0, 1, 0, 0, 1, 3, 0, 0, 3, 1, 1, 0, 1, 3, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 3, 0, 3, 3, 0, 0, 0, 0,
0, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 3, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0,
0, 2, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0,
2, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 3, 0, 2, 0, 0, 1, 3, 1, 0, 0, 1, 3, 0, 0, 0, 0, 0, 0, 1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 3, 1, 3, 0, 0, 1, 3, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 2, 3, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 0, 1, 0, 0, 0, 0, 2, 2, 0, 0,
2, 0, 0, 0

## APPLYING THE ALGORITHM

To apply the algorithm,

- 1 Open an image of interest.
- 2 Call Algorithms>Morphological >Skeletonize.
- 3 In the Skeletonize dialog box that appears,
  - Specify the number of pixels to remove (prune);
  - Specify the destination frame for the result image;
  - And also specify whether you want apply the skeletonizing to the whole image or only for selected VOIs.
- 4 Press OK to run the algorithm.

When the algorithm finishes running, the result image appears in the specified frame. See also Figure 24.



Figure 24. Skeletonize operation: (a) the original image, (b) the result after applying the operation with the parameters shown in the dialog box

## Skeletonize 3D pot field

The Skeletonize 3D Pot Field algorithm uses an iterative approach that simultaneously performs a hierarchical shape decomposition and a corresponding set of multi-resolution skeletons. A skeleton of a model is extracted from the components of its decomposition, therefore it causes both processes and the qualities of their results being interdependent. In particular, if the quality of the extracted skeleton does not meet some user specified criteria, then the model is decomposed into finer components and

a new skeleton is extracted from these components. The process of simultaneous shape decomposition and skeletonizing iterates until the quality of the skeleton becomes satisfactory.

For an input image, the algorithm computes a family of increasingly detailed curve-skeletons. The algorithm is based upon computing a repulsive force field over a discretization of the 3D object (voxelized representation) and using topological characteristics of the resulting vector field, such as *critical points* and *critical curves*, to extract the curve-skeleton.

The algorithm performs the following steps:

- 1** It identifies the boundary voxels of the 3D object as the source of the repulsive force field.
- 2** It computes the repulsive force function at each object voxel and produces a 3D vector field results.
- 3** The algorithm detects the critical points of the 3D vector field and connects them using path-lines by integrating over the vector-field. This step produces the core skeleton.
- 4** Then, it computes the divergence of the vector field at each voxel. Points with low divergence values are selected as new seeds for new skeleton segments. Varying the divergence threshold (given as a percentage, i.e., the top 20%) creates the Level 1 hierarchy after the core skeleton.
- 5** Finally, the algorithm computes the curvature at every boundary voxel and selects new seed points based on a user-supplied curvature threshold, given again as a percentage of the highest curvature value in the dataset (i.e. top 30%). This adds another level of hierarchy to the core and divergence skeletons, the Level 2 skeleton hierarchy.

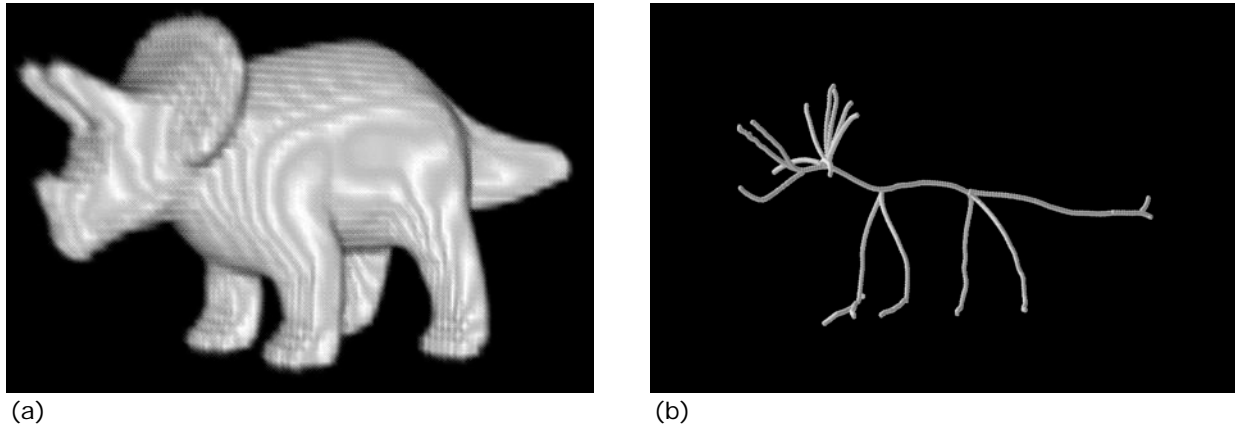


Figure 25. (a) - the original image, (b) - the skeleton<sup>1</sup>

1. These images are borrowed from the Nicu D. Cornea web site <<<http://www.caip.rutgers.edu/~cornea/Skeletonization>>>/

## ALGORITHM PARAMETERS

The key idea behind the potential field approach used in the algorithm is to generate a vector field inside the image by charging the image boundary. In order to generate that field, the algorithm uses a number of user defined parameters, such as *electrical point charge* or *field strength*, that should be specified in the Skeletonize 3 D dialog box, see Figure 27 on page 554.

### Maximum value for outer pixel to be object

The algorithm places electrical point charges on the image boundary, considering each boundary voxel to be a point charge (a boundary voxel is determined as an image voxel which has a background neighbor). In order to differentiate the background voxels from image voxels, the **Maximum value for outer pixel to be object** parameter is added to the Skeletonize 3D dialog box. This parameter allows a user to enter the *max* intensity value for image voxels.

### Field strength and Distance of electrical charge from object boundary

The repulsive force at an interior object voxel due to a nearby point charge is defined as a force pushing the voxel away from the charge with a strength that is inverse proportional to the **distance between the voxel and the**

**charge**, raised at a certain power  $M$  which is called (in this algorithm) the *field strength*. The **Field strength** and **Distance of electrical charge from object boundary** parameters are defined by the user. The final repulsive force at each interior voxel is computed by summing the influences of all the point charges. The resulting vector field is also called a *force field*.

Note that a high field strength ( $M$ ) will cause for a given interior voxel the charges from the local or closer points to have a higher influence than from the more distant charges, therefore creating a vector field with sharper path-lines because it follows the local boundary topology more closely. A low value for the field strength parameter will produce a smoother vector field, with more rounded corners, since the vector direction at a particular point is now influenced by more point charges.

Setting the **Distance of electrical charge from object boundary** parameter to a very low value is not a good idea. E.g., imagine the example of a very long cylinder. Setting the threshold smaller than half the length of the cylinder will cause the field not to flow towards the one attracting point in the middle of the cylinder. Instead, it will go towards the center, creating a critical point at each slice along the cylinder.

### **Fractions of divergence points to use**

The divergence of a vector field in a given region of space is a scalar quantity that characterizes the rate of flow leaving that region. A negative divergence value at a point indicates that the flow is moving mainly *towards* the given point. The algorithm takes the points with low divergence values, which indicate a low spot or “sink,” from these points the new *seed points* are chosen using the *threshold on the divergence value* parameter. The threshold is given as a fraction of the lowest divergence value in the entire vector field. From each of these new seed points, a new field-line is generated which will connect to the core skeleton.

By varying the **Fractions of divergence points to use** parameter the user can vary the number of seed points selected, and therefore the number of new skeleton segments, generating an entire hierarchy of skeletons that is called the Level 1 skeleton. Different values can be chosen based upon the application of the curve-skeleton.

## APPLYING SKELETONIZE 3D

**Preliminary steps.** In order to apply the Skeletonize 3D Pot Field algorithm to an image, the image must have a sufficient number of planes composed solely of background pixels at the x, y, and z boundaries of the image. For the default value of zero for the distance of the electrical charges from the object boundary, there must be background planes at  $x = 0$ ,  $x = x_{Dim} - 1$ ,  $y = 0$ ,  $y = y_{Dim} - 1$ ,  $z = 0$ , and  $z = z_{Dim} - 1$ . Call the Utilities > Add Image Margins, Utilities > Pad, or Utilities > Insert Slice menus to run a tool that will help you to create a version of the image with the sufficient number of padding pixels.

**To apply the algorithm,**

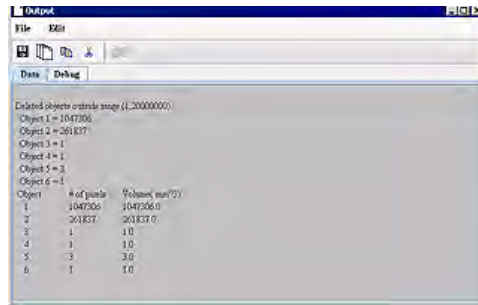
- 1 Open an image of interest, and then call Algorithms > Morphological > Skeletonize 3D Pot. The Skeletonize 3D dialog box appears.
- 2 Fill out the dialog box. For the dialog box options, refer to Figure 27 on page 554.
- 3 Press OK to run the algorithm.

Be patient, because the first algorithm run may take a considerable time. When the algorithm finishes running, the result skeleton image appears. See Figure 26 on page 552.

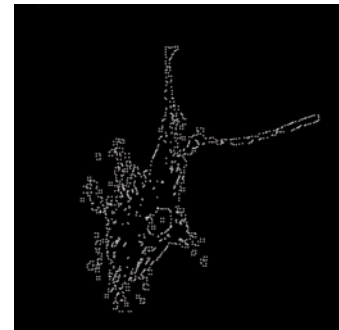


Figure 26. (a): the original image; (b): the Pad dialog box; (c): the padded image; (d): the Output window shows the skeletonize 3D statistics; (d): the Level 1 skeleton obtained using the default parameter values, see Figure 27 on page 554.





(d)



(e)

Figure 26. (a): the original image; (b): the Pad dialog box; (c): the padded image; (d): the Output window shows the skeletonize 3D statistics; (e): the Level 1 skeleton obtained using the default parameter values, see Figure 27 on page 554.

## ALGORITHM NOTES

- Most of the image calculation time (e.g. 98%) will spend in the potential field calculation. Therefore, the first time you run the algorithm, select the Save the Vector Fields radio button to save the x, y, and z vector fields and the Level 1 skeleton. On the following runs, select the Load the Vector Field from Files radio button to load the x, y, and z vector fields and the Level 1 skeleton. See also Figure 27 on page 554.
- Then, you can also vary the **Fractions of divergence points to use** parameter to change the extensiveness of the skeleton generated. As more divergence points are used in the Level 2 skeleton calculation, the skeleton will become more extensive.
- The algorithm must be run on a *single solid object*. Use the following options to make sure that your image is a single solid object:
  - A user selected threshold (defined using the **Maximum value for outer pixel to be object** parameter) helps separate the background voxels from the object voxels.
  - The default selected checkbox for slice by slice hole filling is used to convert the image into a single solid object.
  - After identifying all image objects in 3D, all but the largest object are deleted.

Parameters	
<b>Maximum value for outer pixel to be object</b>	For the image voxels, it determines the <b>max</b> intensity value that a voxel can have in order to be the image voxel (not the boundary voxel). The default value is set to 1.0
<b>Slice by slice hole filling</b>	If checked, performs the slice by slice hole filling for the image. Refer to "Algorithm notes" on page 553.
<b>Distance of electrical charge from object boundary</b>	This parameter determines the distance between the inferior voxel and the boundary point charge. If a boundary point is at a distance greater than the entered value, then it is ignored, i.e., it does not influence the field at this point. See also "Algorithm parameters" on page 550.
<b>Field strength</b>	<p>A vector field inside the image is determined by the strength of the repulsive force from the nearby charge that affects each interior voxel. The repulsive field strength is inverse proportional to the distance between the voxel and the charge (determined by the <b>Distance of electrical charge from object boundary</b> parameter) raised at a certain power M which is determined by the Field Strength parameter. The final repulsive force at each interior voxel is computed by summing the influences of all the point charges.</p> <p>If the Field Strength parameter value is set high, it will create a vector field with sharper path-lines, because it follows the local boundary topology more closely. A low value for the Field Strength parameter will produce a smoother vector field, with more rounded corners. See also "Algorithm parameters" on page 550.</p>
<b>Fractions of divergence points to use</b>	By varying this parameter you can vary a number of seed points and correspondingly the number of new skeleton segments, generating an entire hierarchy of skeletons for the Level 1 skeleton. See also "Algorithm parameters" on page 550.
<b>Save the vector fields to files</b>	The first time this program is run on an image select this option to save the x, y, and z vector fields and the Level 1 skeleton.
<b>Load the vector field from files</b>	On the following runs select the Load the vector field from files option to load the previously saved x, y, and z vector fields and the Level 1 skeleton.

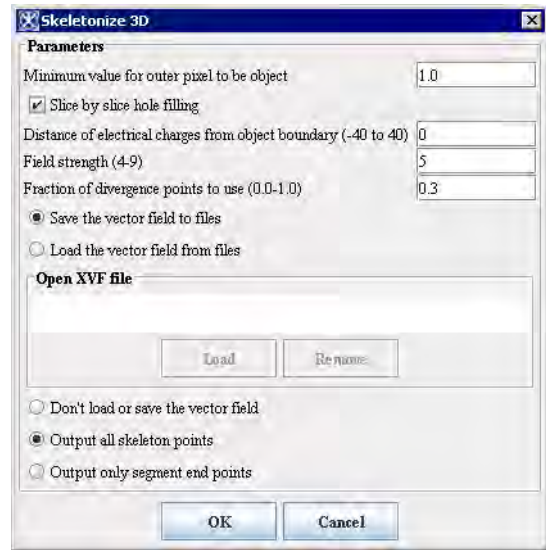


Figure 27. Skeletonize 3D dialog box

On following runs vary the <b>Fractions of divergence points to use</b> parameter to change the extensiveness of the skeleton generated. As more divergence points are used in the Level 2 skeleton calculation, the skeleton will become more extensive.	
<b>Open XVF file</b>	
<b>Load</b>	Loads the file.
<b>Remove</b>	Removes the file from loading.
<b>Don't save the vector field</b>	If checked, doesn't allow to save the vector field.
<b>Output all skeleton points</b>	If checked, sends to the output all skeleton points.
<b>Output only segmented end points</b>	If checked, sends to the output only segmented end points.
<b>OK</b>	Performs the skeletonizing on the selected image based on your choices in this dialog box.
<b>Cancel</b>	Disregards changes you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for the algorithm.

Figure 27. Skeletonize 3D dialog box (continued)

## SKELETONIZE 3D POT REFERENCES

This is a port of the C++ code for pfSkel: Potential Field Based 3D Skeleton Extraction written by Nicu D. Cornea, Electrical and Computer Engineering Department, Rutgers, The State University of New Jersey, Piscataway, New Jersey, 08854, cornea@caip.rutgers.edu. The code was downloaded from <http://www.caip.rutgers.edu/~cornea/Skeletonization/>

Chuang J-H, Tsai C, Ko M-C (2000) Skeletonization of Three-Dimensional Object Using Generalized Potential Field. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(11):1241-1251

Nicu D. Cornea, Deborah Silver, and Patrick Min, Curve-Skeleton Applications, *Proceedings IEEE Visualization*, 2005, pp. 95-102.

Nicu D. Cornea, Deborah Silver, Xiaosong Yuan, and Raman Balasubramanian, Computing Hierarchical Curve-Skeletons of 3D Objects, *Springer-Verlag, The Visual Computer*, Vol. 21, No. 11, October, 2005, pp. 945-955.

---

## Ultimate erode

The algorithm generates the ultimate eroded points (UEPs) of an Euclidian distance map for an image. These UEPs represent the centers of particles that would be separated by segmentation. The UEP's gray value is equal to the radius of the virtual inscribed circle of the corresponding particle.

---

### IMAGE TYPES

It requires a binary image (boolean type) as an input.

---

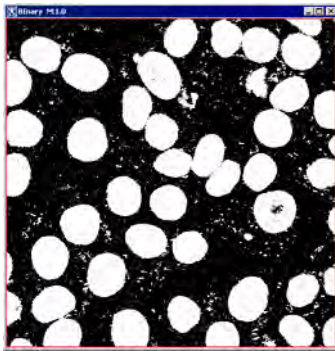
### APPLYING ULTIMATE ERODE

To apply the algorithm,

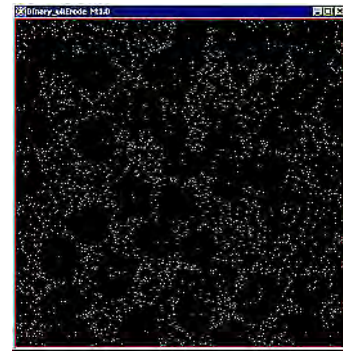
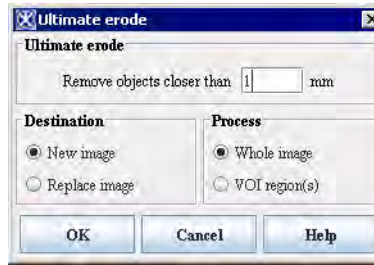
- 1** Call Algorithms>Morphological >Ultimate Erode.
- 2** In the Ultimate Erode dialog box that appears,
  - Use the **Remove objects closer than** text box to specify the min distance between the objects;
  - Specify the image frame for the result image;
  - And also specify whether you want apply Ultimate Erode to the whole image or only for selected VOIs.
- 3** Press OK to run the algorithm.

When the algorithm finishes running, the result image appears in the specified frame.

See also Figure 28.



(a)



(b)

Figure 28. Ultimate Erode – (a) the original binary image, (b) the result after applying the operation with the parameters shown in the dialog box

## Muscle Segmentation

*The Muscle Segmentation plug-in from MIPAV is a semi automatic tool for segmenting different muscles and muscles and fat in Computed Tomography (CT) images of the thigh and abdomen. Of particular interest is the lean to fat tissue ratio in the quadriceps, hamstrings, and Sartorius, while the remaining muscles are grouped together into the abductors. The Muscle Segmentation plug-in uses CT images of the thigh as this imaging modality provides the best muscle discrimination compared to other imaging options. The method incorporates several algorithms that interactively identify the Fasceal border and borders of the muscles of interest it also uses additional methodologies for preventing the same tissue being identified as belonging to two different muscles.*

## Background

The method uses the LiveWire tool for delineating boundaries in images. Specifically, the LiveWire tool allows users to identify a starting location on the boundary and then it automatically determines the minimum cost path between that starting point and current location of the cursor. The user accepts the LiveWire path by identifying an ending point. LiveWire then inserts additional points along the minimum cost path so that a piece wise linear approximation closely matches the minimum cost path. Once the boundary of a muscle is identified, the enclosed region is set to a uniform background color. This re-coloring prevents the LiveWire tool from including pixels already identified as belonging to one muscle while the user identifies additional adjacent muscles.

---

### OUTLINE OF THE METHOD

The method is composed of a sequential set of different processing steps. A block diagram is shown in Figure 1. The input image is a CT image containing both the left and right thighs and also a phantom containing water and some other material with known density values. The output is a pseudo colored image, where different muscles and tissues correspond to different colors or shades of gray. The final image is referred to the segmented or label image. See Figure 9.

The steps are as follows:

- 1** “Water calibration” using a phantom
- 2** “Skin identification”
- 3** Applying “LiveWire” and “Level set tool” to outline the VOIs on the image
- 4** “Fasceal detection and subcutaneous fat detection”
- 5** “Threshold classification and quantification”
- 6** “Tissue Classification”
- 7** “Bone and bone marrow identification”
- 8** “Quadriceps identification, classification, and quantification”
- 9** “Hamstring and Sartorius identification, classification, and quantification”
- 10** Showing results, refer to “Results” on page 567

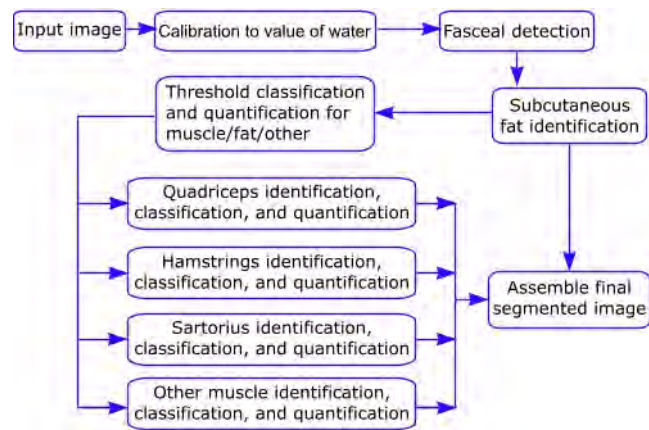
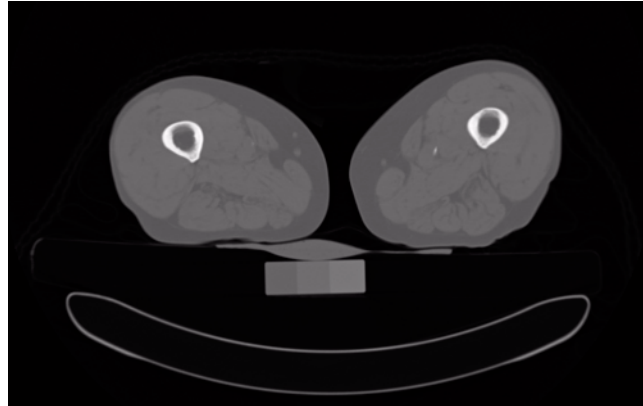


Figure 1. A block diagram of the thigh muscle segmentation method

## WATER PHANTOM

The water phantom is included in the image so that an offset can be computed for each image in the study set, and it is used to correctly adjust the Hounsfield Units (HU) within the image. Figure 2 shows a typical image processed by the plug-in.



**Figure 2.** A typical image processed by the Muscle Segmentation plugin. Visible in the figure are the right and left thigh as well as the water phantom, which appears as the gray rectangular block at the bottom center of the image

## WATER CALIBRATION

The processing begins with the user identifying a rectangular region-of-interest (ROI) completely inside the water phantom and free of any partial volume pixels. The median value of this region is then determined and used as the Hounsfield value of water. Since this value should be zero in CT, the median value is subtracted from all image pixels.

## LIVEWIRE

The LiveWire tool implemented in the plug-in is used as an interactive boundary finding optimization. A local cost function involving a weighted combination of gradient magnitude, gradient direction, and zero-crossings of the Laplacian operator is computed for each pixel in the image. As the user moves from a selected starting point the result is a pixel-to-pixel path with a total minimum cost from the anchor point to the current cursor location as shown in Figure 3.

LiveWire produces a weighted directed graph across the image of the costs is generated via a minimum-spanning tree of the graph computed using Dijkstra's graph searching algorithm. Figure 3-b shows a graph of intensity values across the green line drawn in Figure 3-a to evaluate the accuracy of



LiveWire. Figure 3-c and Figure 3-d provide an analysis of the location of the LiveWire VOI.

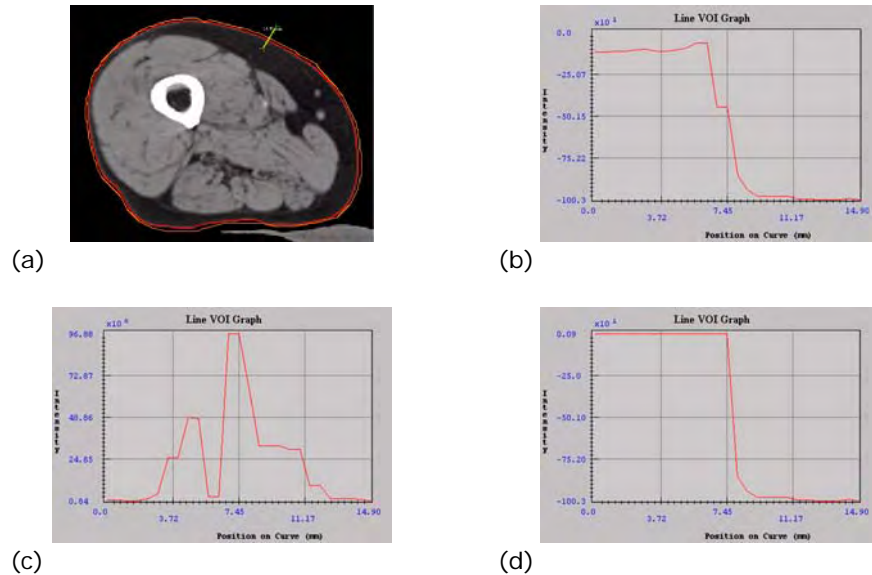


Figure 3. LiveWire produced VOI (Orange), LevelSet produced VOI (Red); Plot of intensity values along green line; Magnitude of gradient along the green line perpendicular to thigh edge; Intensity values when all the pixels inside the livewire detected VOI have been set to 1

## HOW LIVEWIRE WORKS

First, the grayscale image is modeled as a rectangular matrix whose pixel values are integers ranging from 0 to 255 (if the image is a color one, it might be converted to a grayscale one and use the algorithm the same way, although it's better to maximize the costs on each color channel). Each pixel of the matrix is a vertex of the graph and has edges going to the 8 pixels around it, as up, down, left, right, upper-right, upper-left, down-right, down-left. The edge costs are defined based on a cost function.

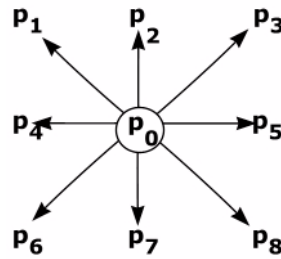


Figure 4. Each pixel of the matrix is a vertex of the graph and has edges going to the 8 pixels around it, as up, down, left, right, upper-right, upper-left, down-right, down-left

Since a minimum cost path should correspond to an image component boundary, pixels (or more accurately, links between neighboring pixels) that exhibit strong edge features should have low local costs and vice-versa. Thus, local component costs are created from the various edge features:

- Laplacian Zero-Crossing  $f_Z$
- Gradient Magnitude  $f_G$
- Gradient Direction  $f_D$

The local costs are computed as a weighted sum of these components. Letting  $l(p, q)$  represent the local cost on the directed link from pixel  $p$  to a neighboring pixel  $q$ , the local cost function is where each  $w$  is the weight of the corresponding feature function<sup>1</sup>.

EQUATION 1

$$l(p, q) = w_Z \times f_Z(q) + w_G \times f_G(q) + w_D \times f_D(p, q)$$

The Laplacian zero-crossing is a binary edge feature used for edge localization. Convolution of an image with a Laplacian kernel approximates the 2-nd partial derivative of the image. The Laplacian image zero-crossing corresponds to points of maximal (or minimal) gradient magnitude. Thus, Laplacian zero-crossings represent “good” edge properties and should,

1. Empirically, weights of  $w_Z = 0.43$ ,  $w_G = 0.43$ , and  $w_D = 0.14$  seem to work well in a wide range of images.

therefore, have a low local cost. If  $I_L(q)$  is the Laplacian of an image  $I$  at pixel  $q$ , then

EQUATION 2

$$f_Z(q)=0, \text{ if } I_L(q)=0 \text{ and } f_Z(q)=1, \text{ if } I_L(q) \neq 0$$

However, application of a discrete Laplacian kernel to a digital image produces very few zero-valued pixels. Rather, a zero-crossing is represented by two neighboring pixels that change from positive to negative. Of the two pixels, the one closest to zero is used to represent the zero-crossing. The resulting feature cost contains single-pixel wide cost where “canyons” used for boundary localization.

Since the Laplacian zero-crossing creates a binary feature,  $f_Z(q)$  does not distinguish between strong, high gradient edges and weak, low gradient edges. However, GRADIENT MAGNITUDE provides a direct correlation between edge strength and local cost. If  $I_X$  and  $I_Y$  represent the partials of an image  $I$  in the  $X$  and  $Y$  directions respectively, then the gradient magnitude  $G$  is approximated with

EQUATION 3

$$G=(I_x^2+I_y^2)$$

The gradient is scaled and inverted, so high gradients produce low costs and vice-versa. Thus, the gradient component function is

EQUATION 4

$$f_G=(\max(G)-g)/\max(G)=1-(G/\max(G))$$

giving an inverse linear ramp function.

Finally, gradient magnitude costs are scaled by Euclidean distance. To keep the resulting maximum gradient at unity,  $f_G(q)$  is scaled by 1 if  $q$  is a diagonal neighbor to  $p$  and by  $1/(\sqrt{2})$  if  $q$  is a horizontal or vertical neighbor. The gradient direction adds a smoothness constraint to the boundary by associating a high cost for sharp changes in boundary direction.

The gradient direction is the unit vector defined by  $I_x$  and  $I_y$ . Letting  $D(p)$  be the unit vector perpendicular (rotated 90 degrees clockwise) to the gradient

direction at point  $p$  (i.e., for  $D(p) = (I_Y(p), -I_X(p))$ ), the formulation of the gradient direction feature cost is

EQUATION 5

$$f_D(p, q) = \frac{2}{3\pi} \cdot (\text{acos}(d_p(p, q)) + \text{acos}(d_q(p, q)))$$

where

EQUATION 6

$$d_p(p, q) = D(p) \cdot L(p, q)$$

and

EQUATION 7

$$d_q(p, q) = L(p, q) \cdot D(q)$$

are vector dot products and

EQUATION 8

$$L(p, q) = q - p, \text{ if } D(p) \cdot (q - p) \geq 0$$

$$L(p, q) = p - q, \text{ if } D(p) \cdot (q - p) < 0$$

is the bidirectional link or edge vector between pixels  $p$  and  $q$ . Links are either horizontal, vertical, or diagonal (relative to the position of  $q$  in  $p$ 's neighborhood) and point such that the *dot product of  $D(p)$  and  $L(p, q)$*  is positive. The neighborhood link direction associates a high cost to an edge or link between two pixels that have similar gradient directions but are perpendicular, or near perpendicular, to the link between them. Therefore, the direction feature cost is low when the gradient directions of the two pixels are similar to each other and to the link between them.

## LEVEL SET TOOL

The level set tool is a less interactive tool that can be used for automatic segmentation of thigh components. Distance from the given contour to the nearest boundary is computed for all pixels from the user-specified point to all points along a path of pixels with close intensity. The level set extraction described for lattices of 3D data produces a collection of unique vertices,

edges, and triangles. These components can be stored as a triangle mesh that is represented by a vertex-edge-triangle data structure. The data structure supports various topological and geometrical queries about the mesh. Refer to the LevelSet tool in the MIPAV User Guide Volume 2.

## Muscle Segmentation steps

---

### SKIN IDENTIFICATION

The skin in the thigh image is identified by interactively isolating the outer boundary using the LiveWire algorithm. Typically, the user needs to identify only three or four points spaced around each thigh and the LiveWire algorithm inserts additional points on the boundary. Once a closed region is identified, a mask image is computed where all pixels inside and on the boundary are assigned a value of one and all other pixels are assigned a value of zero. This mask image is then eroded four times using a binary mathematical morphology operator creating another mask image. All pixels outside the eroded mask image are set to background value equals to  $-1024$  HU.

---

### TISSUE CLASSIFICATION

Threshold classification of CT images allows identification of muscle and fat tissue within the thigh. Potential partial voluming effects are decreased by adopting a schema that is more exclusive than the Hounsfield scale assigns to normalized CT intensities:

$$-190 < \text{fat pixel} < -30$$

$$0 < \text{muscle pixel} < 100$$

$$-30 < \text{volume pixel} < 0$$

---

## FASCEAL DETECTION AND SUBCUTANEOUS FAT DETECTION

The fasceal is a very thin, broken boundary that appears very faint and surrounds the muscles of the thigh. Because of these characteristics, we chose to interactively enter points on this border, which are then joined as a closed polyline. The fasceal boundary separates the subcutaneous fat from the muscles, interstitial fat, and other tissues within the muscles. Therefore, all the non-background pixels outside the fasceal boundary are by definition classified as subcutaneous fat pixels.

---

## BONE AND BONE MARROW IDENTIFICATION

Bone is easily identified in CT images because corresponding pixels are significantly more intense (dense) than those found in the soft tissue. Therefore, we used a threshold value of 176 HU and above to classify the bone pixels. Once these pixels are identified, they are set to background value. The bone marrow pixels are then identified as those pixels above the background value that are inside the bone pixels. All bone marrow pixels are also set to the background value.

---

## THRESHOLD CLASSIFICATION AND QUANTIFICATION

The result of all the processing described above is an image containing only muscle, interstitial fat, partial volume pixels, and perhaps blood vessels. Furthermore, the quantification results include the total thigh area and average Hounsfield value for both muscle and fat. These values are determined by applying a threshold operation, where

- fat pixels are between -190 and -30 HU inclusive,
- muscle pixels are between 0 and 100 HU inclusive,
- and the partial volume muscle and fat pixel correspond to all the values between these ranges.

We refer to the resulting image as the total thigh label image.

---

## **QUADRICEPS IDENTIFICATION, CLASSIFICATION, AND QUANTIFICATION**

The quadriceps is interactively identified with the LiveWire tool using an image that results by blending the total thigh label image with the gray scale image. Once a closed VOI for each quadriceps has been identified, they are classified using the same threshold regions for fat, muscle, and partial volume pixels as described above. The left and right quadriceps are quantified separately and the results of each are saved in an output file. An image with unique values for muscle, fat, and partial volume is also saved and is called the quadriceps label image.

---

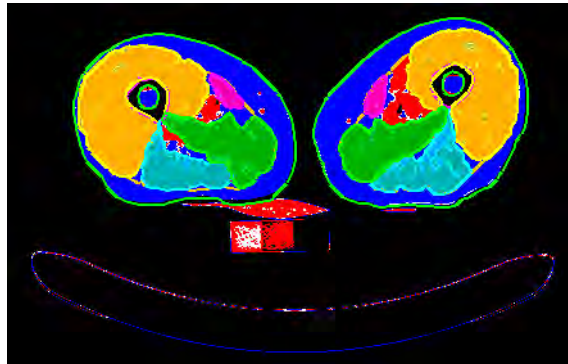
## **HAMSTRING AND SARTORIUS IDENTIFICATION, CLASSIFICATION, AND QUANTIFICATION**

Identifying the hamstrings follows the processing steps for identifying the quadriceps, except that the pixels corresponding to the quadriceps are first set to the background value. This prevents the LiveWire tool from including pixels from a region that has already been identified as a different muscle. This insures there will be no overlapping pixels belonging to two different muscles. The Sartorius is similarly determined in an image with both the quadriceps and hamstrings set to background. Finally, the pixels corresponding to the other muscles (the abductors) are identified as those pixels not already classified.

---

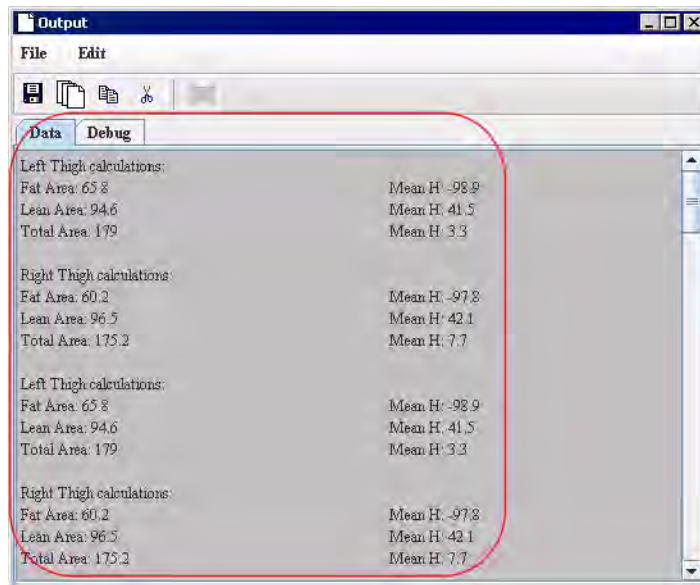
## **RESULTS**

Figure 5 shows the segmented resulting from the Muscle Segmentation plug-in for the image shown in Figure 2. The different muscles are shown in different colors, as well as the subcutaneous and interstitial fat.



**Figure 5. Segmented image that results from the Muscle Segmentation plug-in. The different muscles and fat regions are assigned different values**

The fat/ lean statistics for chosen thigh (or abdomen) objects (e.g. the left and right thighs including all bones and muscles) appears in the Output window and also can be saved as a PDF file.



**Figure 6. Plug-in output**

## IMAGE TYPES

Any CT image of the thighs that can currently be opened by MIPAV.



---

## REFERENCES

Eric N. Mortensen, William A. Barrett, *Intelligent Scissors for Image Composition*, Brigham Young University, Graphical Models and Image Processing, v 60, n 5.

“Level Set Methods and Fast Marching Methods” by J.A. Sethian, Cambridge University Press, 1999.

## Applying the Muscle Segmentation method

To run the plug-in do the following:

- 1** Start MIPAV.
- 2** Install the plug-in (in order to learn how to do that, refer to the MIPAV User’s Guide Volume 1, “Developing Plug-in programs”).
- 3** Open an image of interest (a thigh or abdomen image).
- 4** Call Plugin > General > Muscle Segmentation to run the plug-in.
- 5** The plug-in window appears with the image open in the window. See Figure 7.

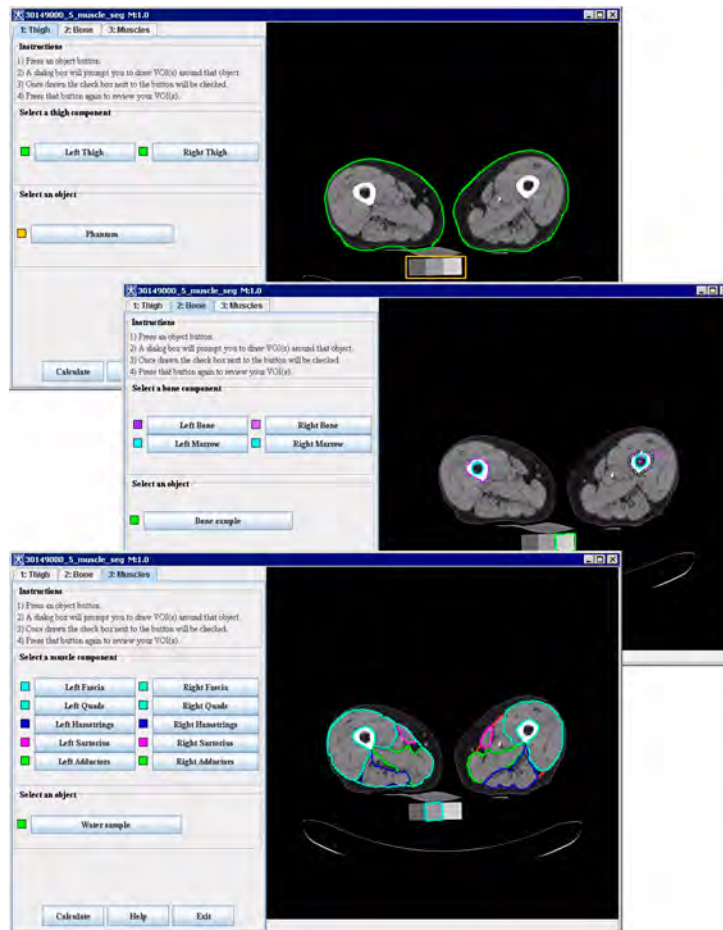


Figure 7. The plug-in user interface with, first, the Thigh tab selected, then the Bone tab selected, and finally, the Muscle tab selected. All VOIs are available for editing

## VOI SELECTION

The left pane in Figure 7 contains three tabs that represent different processing steps. When a tab is pressed the display shows its particular VOI set. Each VOI shown in the right pane has an associated checked button in the left pane. When any given button is pressed, that VOI is available for editing. In Figure 7, all muscle VOIs are filled. Based on these VOIs the plug-in generates a zero intensity image mask, which is then used by the LiveWire tool to guarantee non-overlapping segments.

---

## THIGH

To draw a VOI around a chosen thigh,

- 1** Open the Thigh tab. See Figure 7.
- 2** To select a thigh, press an object button (Left Thigh or Right Thigh).
- 3** The VOI tab appears prompting you to draw a VOI around the chosen thigh, e.g. the left thigh.
- 4** Draw a VOI and then press OK
- 5** The VOI appears on the thigh.

To calculate VOI statistics and fat/lean ratio for the whole thigh, press Calculate. Refer to Analysis section to learn how to show results in the Output window.

---

## BONE

To highlight a bone sample, press the Bone Sample button. The VOI appears selecting the bone sample on the phantom.

To delineate a VOI around a chosen thigh bone,

- 1** Open the Bone tab. See Figure 7.
- 2** To select a bone, press an object button (Left Bone or Right Bone),
- 3** The VOI tab appears prompting you to draw a VOI around the chosen bone, e.g. the left bone.
- 4** Draw a VOI and then press OK
- 5** The VOI appears around the bone.

Repeat the same steps for a bone marrow. Refer to Analysis section to learn how to show results in the Output window.

---

## MUSCLES

To highlight a muscle sample, press the Water Sample button. The VOI appears selecting the water sample on the phantom.

To delineate VOIs around chosen muscles,

- 1** Open the Muscle tab. See Figure 7.
- 2** To select a muscle, press a corresponding object button e.g. Left Fascia or Right Fascia.
- 3** The VOI tab appears prompting you to draw a VOI around the chosen muscle, e.g. the let bone.
- 4** Draw a VOI and then press OK
- 5** The VOI appears around the chosen muscle.
- 6** Repeat the same steps for another muscle.

After you've done with VOIs, click Calculate. This opens the Analysis tab. Refer to Analysis section to learn how to show results in the Output window.

---

## SAVING SEGMENTATION VOIS

Each VOI is saved automatically in a separate XML file with publicly available schema. These files are saved in the image directory and can be viewed and modified outside of this tool and MIPAV. The file name for a VOI file looks like Right Object Name.xml or. Left Object Name.xml. See Figure 8.

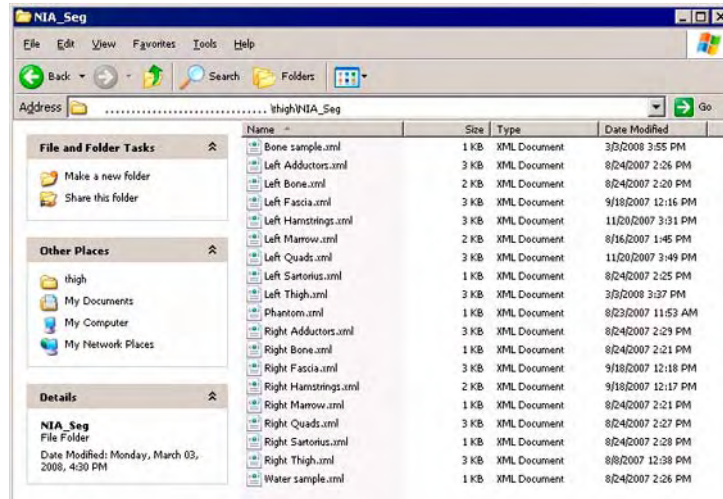


Figure 8. The catalogue where segmentation VOIs are stored

## ANALYSIS

Use the Analysis tab options to select the thigh muscles using VOIs and output VOI statistics for selected VOIs. The statistics include the calculated Fat Area, Lean Area and Total Area for a chosen muscle. The Analysis tab is hidden when you run the plug-in. It appears only when you press the Calculate button in the Thigh, Bone or Muscle tab. See Figure 9.

To run the fat/lean ratio calculation for the thigh

- 1** Select the thigh object(s) (a thigh, a bone, a bone marrow, a muscle) by pressing the corresponding button. Note that you can select as many objects as you wish.
- 2** If you want to calculate a fat/ lean statistics for a single object (e.g. the left hamstring), select that object and then press Output. The fat/ lean statistics for the left hamstring appears in the output window.
- 3** If you want to calculate a fat/ lean statistics for a set of objects (e.g. the left thigh, left hamstring, and left quads), select these objects by pressing the corresponding buttons, and then press Output. The fat/ lean statistics appears in the output window.

- 4 If you want to calculate a fat/ lean statistics for all objects (e.g. the left and right thighs including all bones and muscles) press Output All. This will output a whole fat/lean statistics for both thighs in the Output window. See Figure 9.
- 5 To save the fat/ lean statistic as a PDF file, press Save. The PDF file appears in the same catalogue where VOIs for a selected image are stored.

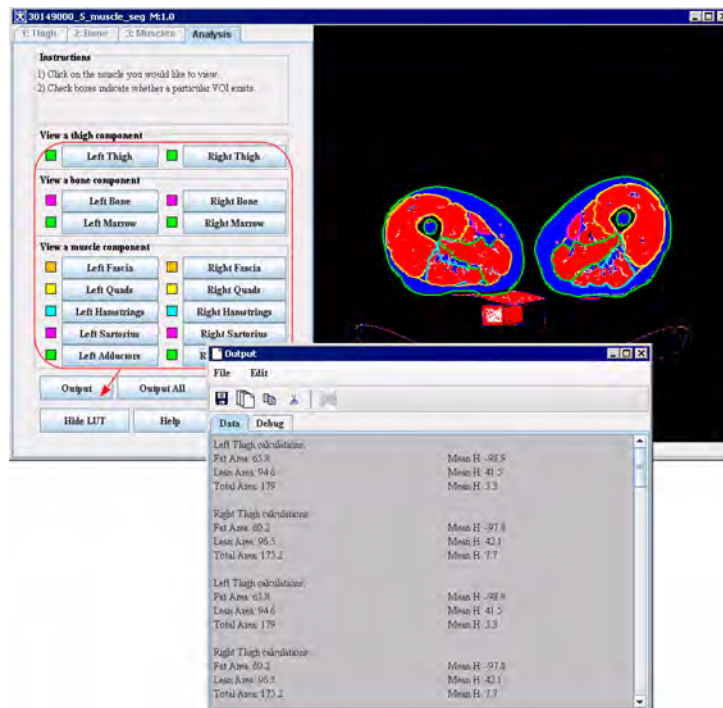


Figure 9. Fat/Lean ratio for the thigh and thigh muscles in the Output window. Use the scroll bar to view the whole statistics

## TO SHOW/HIDE LUT:

To show the LUT, press Show LUT button, the image appears highlighted by LUT. To hide the LUT, press Hide LUT. See Figure 10.

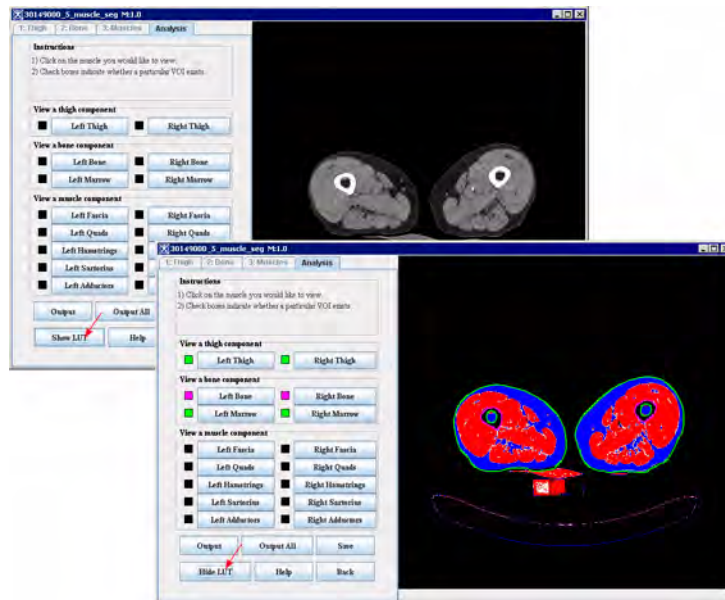


Figure 10. Show and Hide LUT options

## Abdomen Segmentation

*Given abdomen image slice or multiple slices, this algorithm automatically delineates contours that define the abdomen exterior border and subcutaneous fat area. The visceral cavity and abdomen muscles are then semi-automatically determined.*

### Background

The Abdomen Segmentation algorithm starts with the extraction of the abdomen perimeter. Then, it calculates the geometric center of the extracted abdomen area.

The algorithm draws radial lines from the detected abdomen perimeter to the geometric center at fixed angular increments (1–3 degree). Along each line, the it analyzes the intensity profile to determine the most inner point of the subcutaneous fat layer.

The algorithm repeats this calculation for each radial line, and then, forms an inner contour of the subcutaneous fat area by connecting adjacent points. Inner points are identified as the points whose intensity is greater than an upper threshold value (16 HU which is corresponding to the muscle density) and less than lower threshold value of -300 HU.

Further segmentation of the visceral and subcutaneous fat pixels is done using the threshold values with the:

EQUATION 1

$$\begin{aligned}
 & -190 < \text{fat pixel} < -30 \\
 & 0 < \text{muscle pixel} < 100 \\
 & -30 < \text{partial volume pixel} < 0
 \end{aligned}$$

The visceral fat is, then, defined as the intraperitoneal area with pixel intensities that match fat intensities.

The subcutaneous area is defined as the fat layer between the inner body skin coat and the outer margin of the body wall musculature. The subcutaneous fat length (SFL) is defined as the distance between two points – one on the inner subcutaneous fat contour and another one on the outer abdominal contour – and also intersecting the radial line.



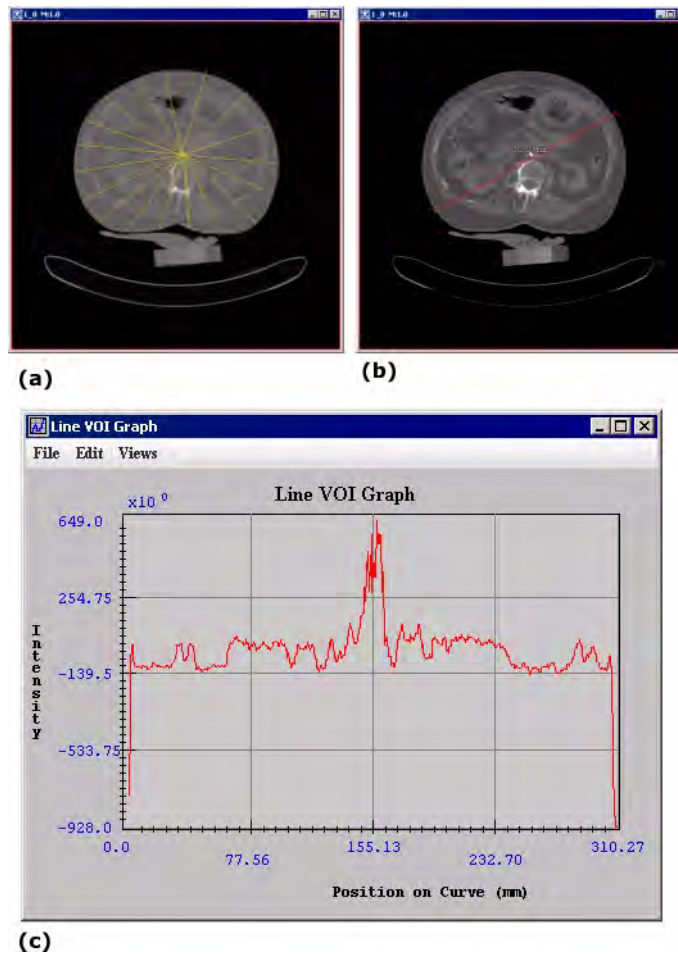


Figure 1. Determination of subcutaneous fat layer: (a) - radii drawn from the outer contour towards the body center at a fixed angle, (b) - a line VOI covering the radius, (c) - an intensity profile along the line VOI

Once the outer and the inner contours have been extracted and smoothed (see Smoothing contours), the algorithm creates the following three masks:

- Body fat mask, which is the area inside the outer contour;
- Visceral cavity mask, which is the area inside the inner contour;
- Subcutaneous fat mask, which is the area between the inner and outer contours.

For a given image slice, the total body fat, visceral fat and subcutaneous fat are calculated as the numbers of pixels that meet the following criteria:

- Enclosed by these masks;

- Have intensities within the fat range (see Equation 1).

For 3D images, fat volumes are calculated using the slice thickness and the values obtained for each slice.

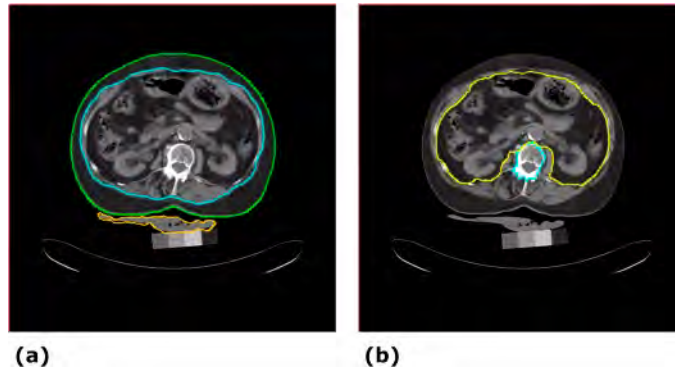


Figure 2. Segmentation: (a) - an abdomen (green) and subcutaneous area (blue) and (b) - a visceral cavity (yellow)

## SMOOTHING CONTOURS

The smoothing method is based on the assumption that the subcutaneous fat length values (SFL) should change smoothly between the adjacent radii. However, larger discontinuities are permitted depending on the separation of the radii that defines (SFL) values. The method calculates the discontinuous contour using the mathematical morphological closing operation.

For more information about the mathematical morphological operations used in MIPAV refer to the User Guide Volume 2, Algorithms available on the MIPAV web site {<http://mipav.cit.nih.gov/documentation.php>}.

## REFERENCES

Automated quantification of body fat distribution on volumetric computed tomography, BINSHEG ZHAO, COLVILLE Jane, KALAIGIAN John, CURRAN Sean, LI JIANG, KIJEWski Peter, SCHWARTZ Lawrence H., *Journal of Computer Assist. Tomogr.*, Volume 30, # 5, Sept/Oct 2006, pp. 777–783.

## Applying the method for the abdomen segmentation

When the plug-in is running, the left dialog box pane contains three tabs – Abdomen, Tissue, and Muscles that represent the subsequent segmentation steps. By default, the dialog opens the Abdomen tab.

When any plug-in tab is open, it shows its particular VOI set. VOIs are color coded. Each VOI shown in the right pane has an associated checked button in the left pane, see Figure 3.



Figure 3. The Abdomen tab shows abdomen VOIs

Pressing any given button opens the VOI tab and makes the associated VOI available for editing. Based on delineated VOIs the plug-in generates a zero intensity image mask, which is then used by the LiveWire tool to guarantee non-overlapping segments.

### ABDOMEN TAB

By default, the Abdomen tab can display three VOIs delineated on the abdomen, subcutaneous area, and water phantom, see Figure 3. There are three options offered by the plug-in:

- If the predefined VOIs are stored in the default directory, the plug-in loads them automatically;
- You can delineate and/or modify VOIs manually using the VOI drawing tools provided by the MIPAV VOI toolbar;
- Or you can run the MIPAV algorithm to delineate abdomen, subcutaneous area VOIs on the image. See also Section Abdomen automatic segmentation algorithm.

For example, to draw the Abdomen VOI,

- 1** Open the image of interest.
- 2** Use the MIPAV Image slice slider to select the slice of interest. Or, you can select the slice of interest later from inside the plug-in. Note that you can use the MIPAV main menu and toolbars from inside the plug-in.
- 3** Run the plug-in.
- 4** In the Abdomen tab, click on the Abdomen button. The VOI tab appears.
- 5** In the VOI tab, use the MIPAV VOI toolbar icons to delineate a single closed VOI around the abdomen. Make sure that MIPAV VOI toolbar is available.
- 6** To propagate the VOI to other slices, use the Propagate icons.
- 7** Click OK to save the VOI.
- 8** Repeat these steps for other tabs, tissues, and muscles. See Figure 9.

---

## MODIFYING VOIS

If you load predefined VOIs, you can, then, modify them. For example, to modify the Abdomen VOI:

- 1** Open the Abdomen tab. See Figure 3.
- 2** Press the Abdomen button.
- 3** The VOI tab appears prompting you to modify the VOI.
- 4** Modify the VOI, and then press OK.

- 5 The message window appears “Abdomen.xml exists. Overwrite?” Press OK if you want to overwrite the existing VOI. Press No if you want to save the VOI under a different name.
- 6 The updated abdomen VOI appears on the image. Repeat these steps for subcutaneous area.

To delete the VOI, select it and then click the Cut icon on the MIPAV toolbar. This will delete the VOI.

To run the fat/lean calculation for the Abdomen and Subcutaneous area, press Calculate. In the Analysis tab that appears, select the Abdomen and Subcutaneous areas by pressing the corresponding buttons. Press Output. The results appear in the Output window.

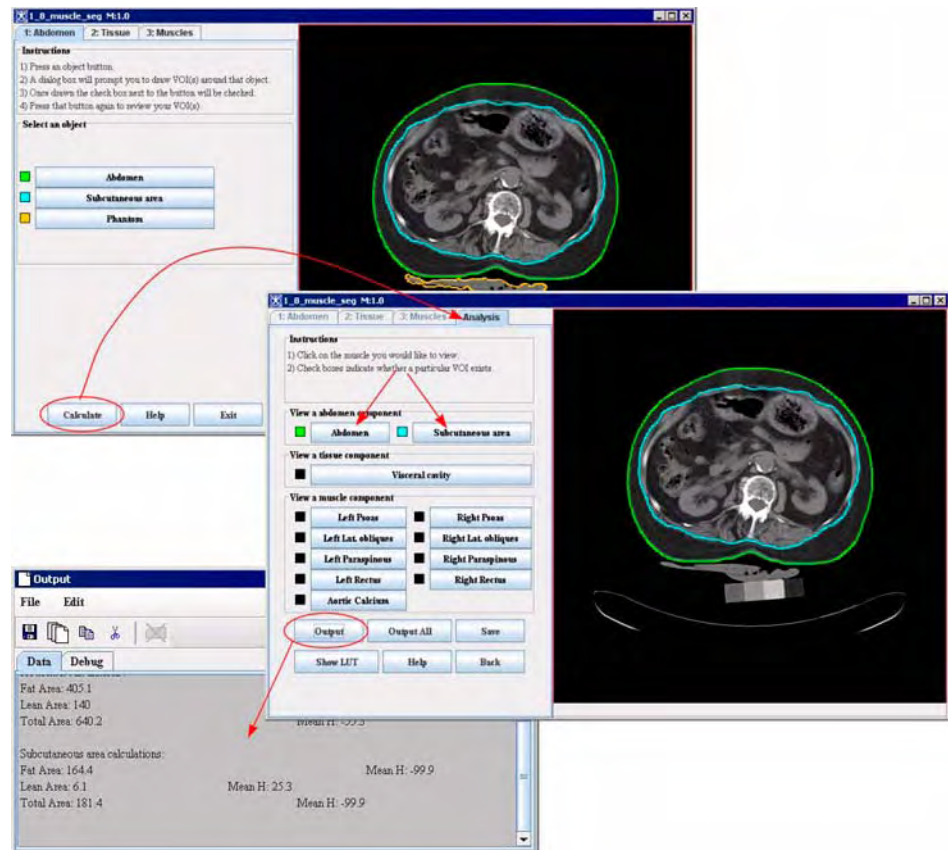


Figure 4. Calculating the fat/lean ratio for the abdomen and subcutaneous area

## TISSUE TAB

In the Tissue tab, you can select and/or edit a VOI drawn around the visceral cavity, bone and water phantom. See Figure 6.

To select/edit a VOI around the visceral cavity,

- 1** Open the Tissue tab.
- 2** If VOIs do not appear automatically, press the Visceral cavity button; this will visualize the corresponding VOI on the image.
- 3** The VOI tab appears prompting you to modify the VOI.
- 4** Modify the VOI if needed, and then press OK.
- 5** The updated VOI appears on the image.
- 6** Repeat these steps for the bone and water phantoms if needed.

If you choose to modify the visceral cavity VOI (or any other VOI), the message window appears “[VOI name].xml exists. Overwrite?” Press OK if you want to overwrite the existing VOI. Press No if you want to save the modified VOI under a different name.

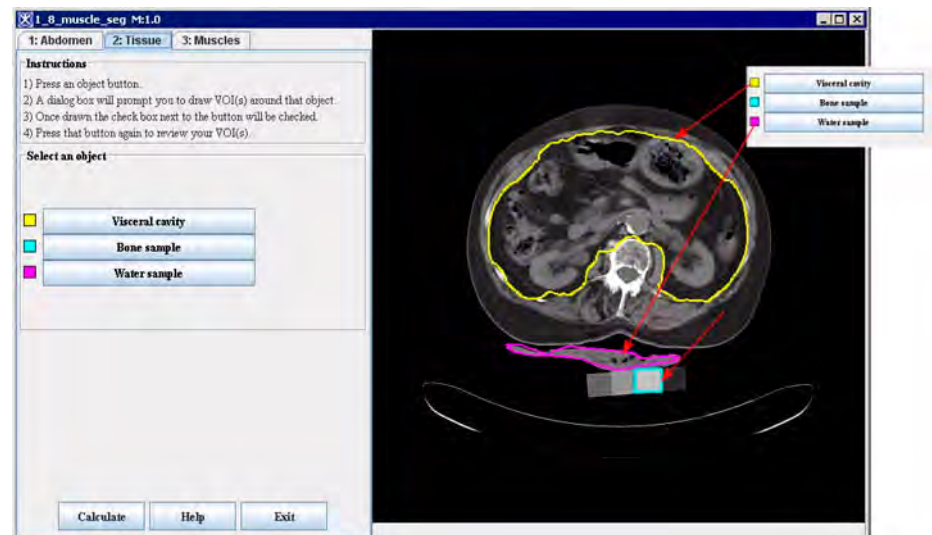


Figure 5. The Tissue tab

## MUSCLES TAB

To highlight an abdomen muscle, press the corresponding button. The VOI appears highlighting the selected muscle on the image.

To modify VOIs around chosen muscles,

- 1 Open the Muscles tab.
- 2 Press the corresponding muscle button, for example, Left Psoas or Right Lat. Obliques.
- 3 The VOI tab appears prompting you to modify the VOI around the chosen muscle.
- 4 Modify the VOI and then press OK.
- 5 Repeat the same steps for other muscles.

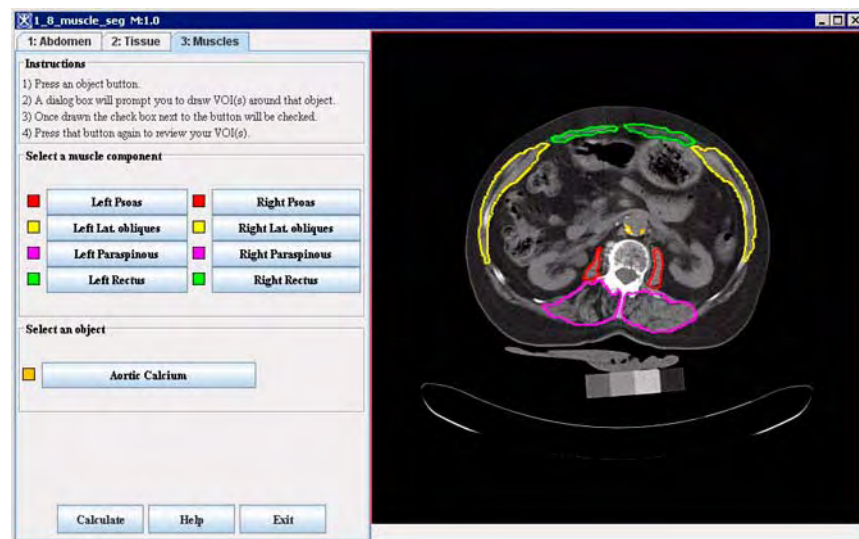


Figure 6. The Muscles tab

If you choose to modify the muscle VOI and pressed OK, the message window appears “[muscle name].xml exists. Overwrite?” Press OK if you want to overwrite the existing VOI. Press No if you want to save the modified VOI under a different name.

If you press the Aortic Calcium button, this will visualize the VOI contours drawn around the aortic calcinations.

After you've done with VOIs, click Calculate. This opens the Analysis tab. In the Analysis tab select the areas of interest by pressing the corresponding buttons. Press Output. The results appear in the Output window.

---

## VOI TAB

The VOI tab appears when you click on the object button, for example, Abdomen button, in the Abdomen, Tissue and Muscles tab. In the VOI tab, you can select, highlight, modify and save the VOI. See also VOI tab warnings.

The following options are available:

**VOI Selection box** – this is the information box that tells you which VOI is about to being modified as well as lets you know the maximum number of curves allowed to be drawn for a particular VOI.

**OK** – if you choose to modify the VOI and pressed OK, the message window appears “[VOI name].xml exists. Overwrite?” Press OK if you want to overwrite the existing VOI. Press No if you want to save the modified VOI under a different name. Note that if the result VOI is not correct the plug-in will issue a warning message, refer to Section “VOI tab warnings” .

**Hide all** – hides all VOIs.

**Show all** – displays all VOIs.

**Cancel** – closes the VOI tab and returns you to the previous visited tab.

**Reset** – resets the image.

**Hide except [current VOI name]** – hides all VOIs except the current VOI.

To delete a chosen VOI use the Cut icon from the MIPAV toolbar.

---

## VOI TAB WARNINGS

*“Too many curves” warning.* If you mistakenly draw too many curves for a chosen VOI, the plug-in will detect it and notify you when you press OK in order to save the VOI. It will not allow you to save the VOI until you delete the unwanted VOI(s).



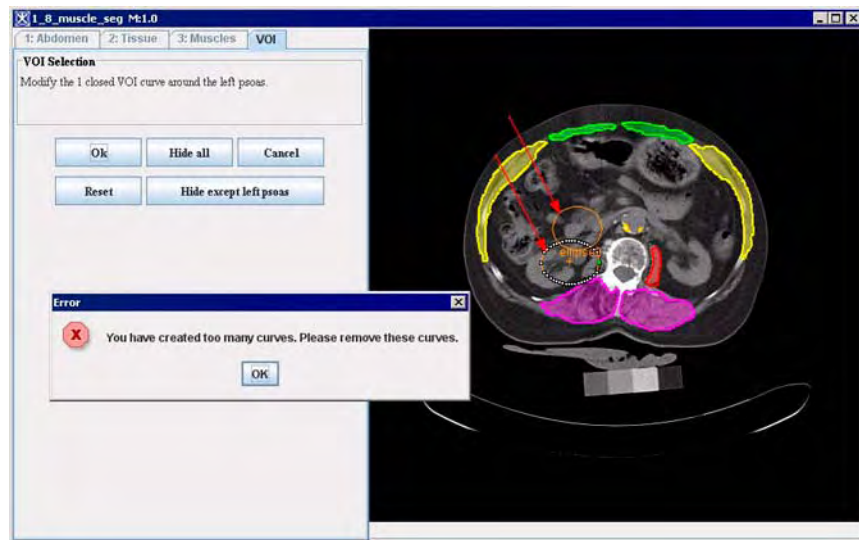


Figure 7. The “Too many curves” warning

*“Any curves made must be closed” warning.* Any VOI drawn on the abdomen must be a closed curve. You are not allowed to draw an open line or point VOI, while using the plug-in. If the open VOI was drawn, the warning appears, informing you that the VOI is not closed.

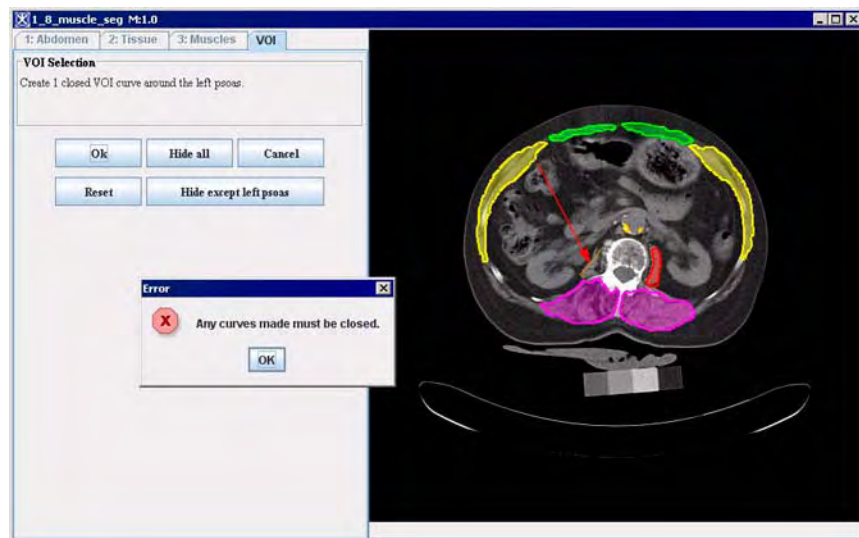


Figure 8. The “Any curves made must be closed” warning

*“MIPAV created VOI” warning.* When you open the VOI created by MIPAV,

the VOI tab displays the warning notifying you that this VOI was created automatically and needs revision. See also Section MIPAV automatic segmentation and Figure 9.

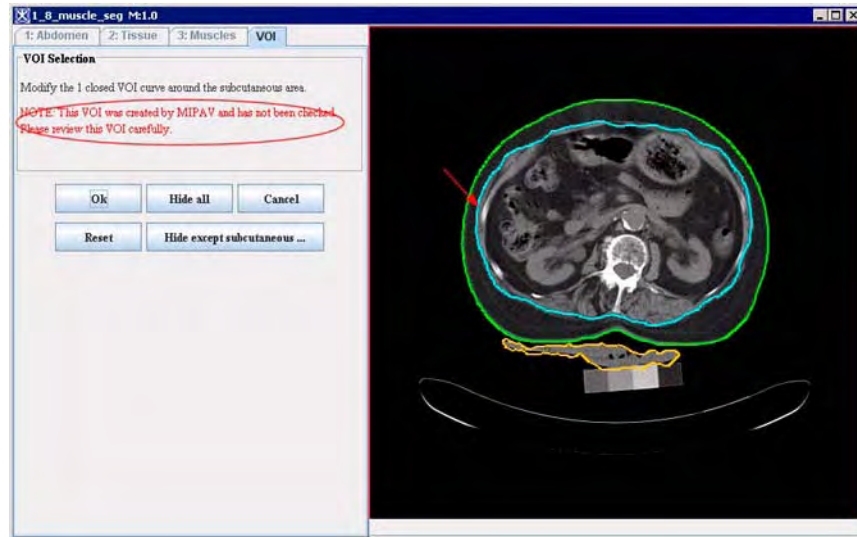


Figure 9. The “MIPAV created VOI” warning

## WHERE VOIS ARE STORED

By default, the plug-in looks for VOIs in the `NIA_Seg` subdirectory of the directory where the image dataset is stored. For example, if your abdomen image of interest is located in the `C:\MIPAV\images\AbdomenImages` catalogue, you should copy the image segmentation VOIs to the `C:\MIPAV\images\AbdomenImages\NIA_Seg` catalogue. In that case the plug-in will load those VOIs automatically. See Figure 10.

When created and/or modified, each VOI is saved in a separate XML file with the publicly available schema. The VOI files can be viewed and modified outside of this tool and MIPAV. The file name for a VOI file looks like `Object Name.xml` or `Right/Left Object Name.xml`.

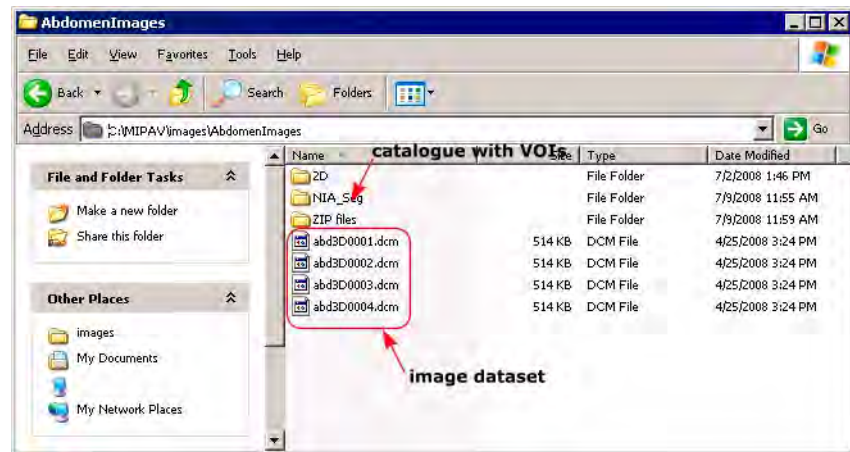


Figure 10. The plug-in looks for VOIs in the subdirectory of the directory where the image dataset is stored

## CHANGING COLOR FOR VOIS

The default colors for VOIs are shown in Figure 15. You can change the color for a selected VOI(s) using the options offered by the Pick VOI Color dialog box, see Figure 11. Note that these changes will be in effect only for a current plug-in session. When you restart the plug-in all VOIs will return to the default colors.

To change the color for a chosen VOI, click on the colored square next to the VOI's button. This will open the Pick VOI Color dialog box. In the dialog box, select the color from the color palette and press OK. The new color will apply to the chosen VOI. See Figure 11.

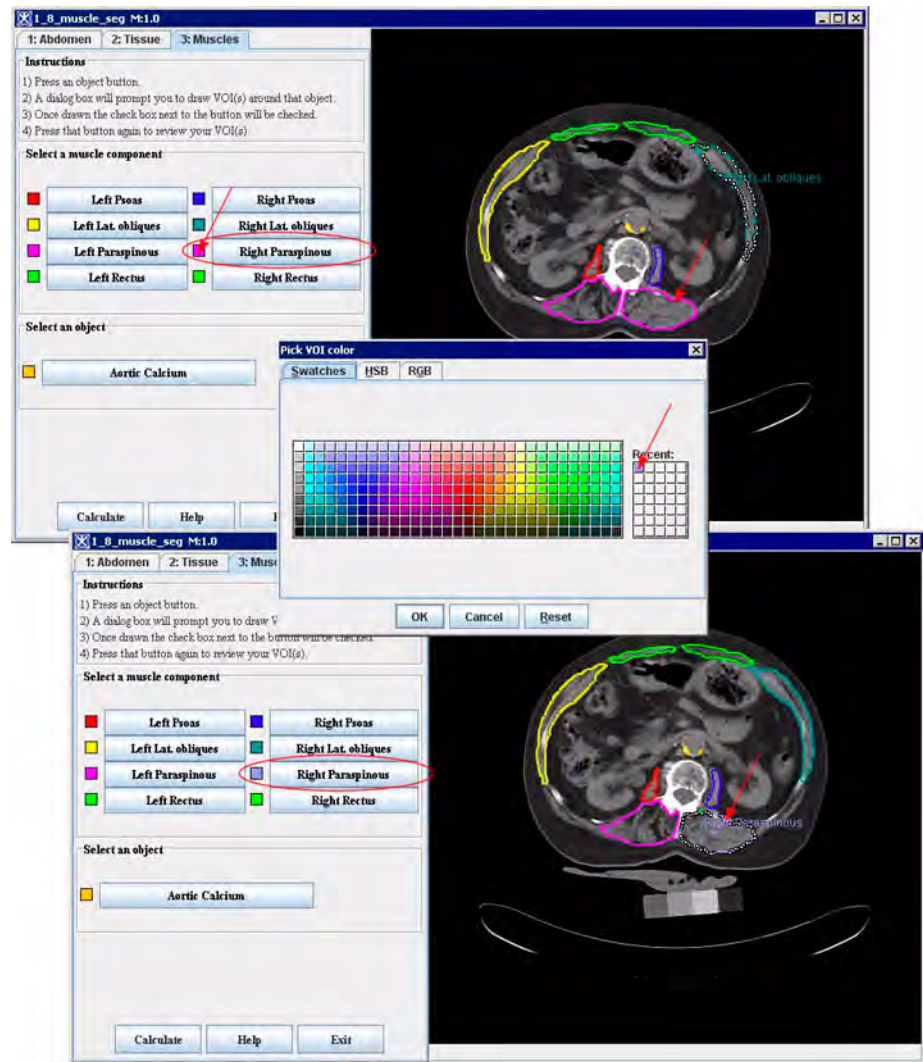


Figure 11. VOI color selection

## MANUAL SEGMENTATION

You can draw VOIs from inside the plug-in using the MIPAV VOI toolbar options. For example, to draw the Abdomen VOI,

- 1** Open an abdomen image of interest.
- 2** Run the plug-in. See Figure 12.
- 3** Select the Abdomen tab.

- 4 Use the MIPAV Image slice slider to select the slice of interest.
- 5 In the Abdomen tab, click on the Abdomen button. The VOI tab appears.
- 6 In the VOI tab, use the MIPAV VOI toolbar icons to delineate a single closed VOI around the abdomen. Make sure that MIPAV VOI toolbar is available.
- 7 To propagate the VOI to adjacent slices, use the Propagate icons.
- 8 Click OK to save the VOI.
- 9 Repeat these steps for other VOIs. See also Section “VOI tab warnings”.

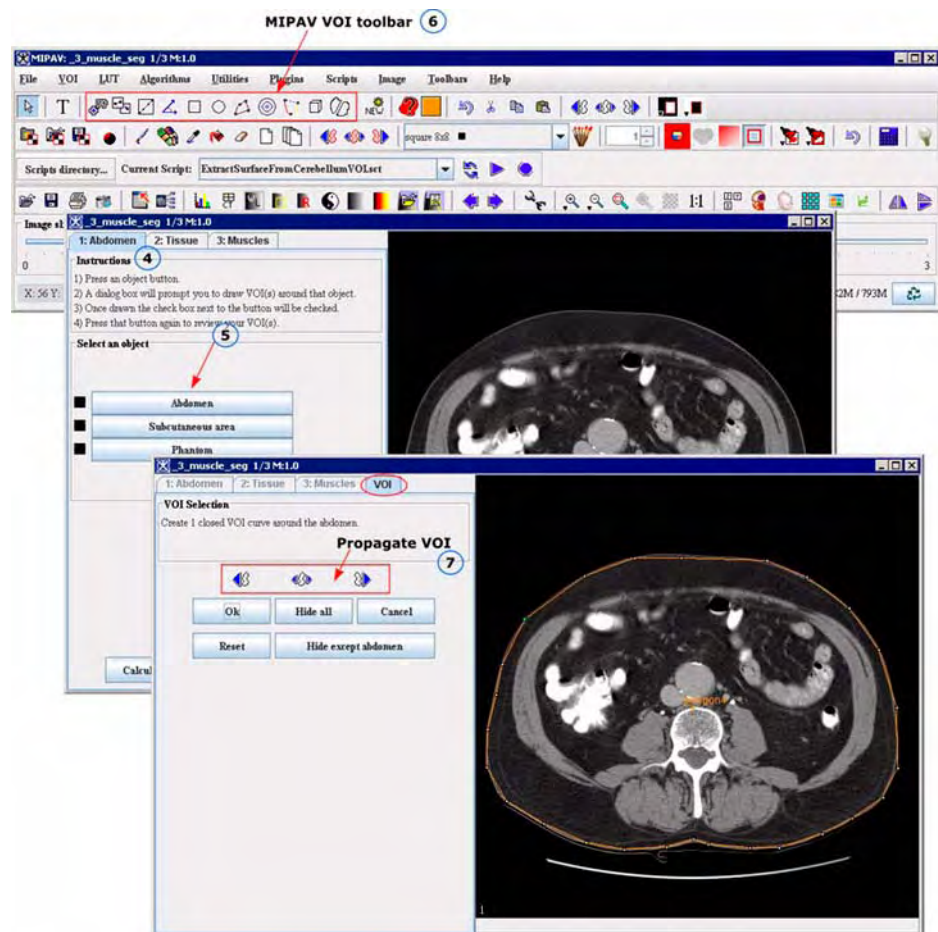


Figure 12. Manual segmentation steps

## MIPAV AUTOMATIC SEGMENTATION

By default, the plug-in calculates the fat/lean ratio for the abdomen based on the predefined VOIs. The VOIs for a chosen image dataset should be stored as XML files in the NIA\_Seg subdirectory of the directory where the image file is stored. In that case the plug-in finds VOIs and loads them automatically. Refer to Section “Where VOIs are stored” on page 586.

In case if some VOIs are missed, the plug-in will display the warning window and ask you if you want MIPAV to create these VOIs automatically. If you hit Yes, MIPAV will run the automatic segmentation procedure and delineate missed VOIs on the image. For more information about the abdomen segmentation algorithm, refer to Section “Background” on page 576.

To distinguish VOIs created by MIPAV from the VOIs that come with the image, the plug-in color codes the VOI buttons as they appear in the tabs. For VOI created by MIPAV, the corresponding buttons appear in red color.

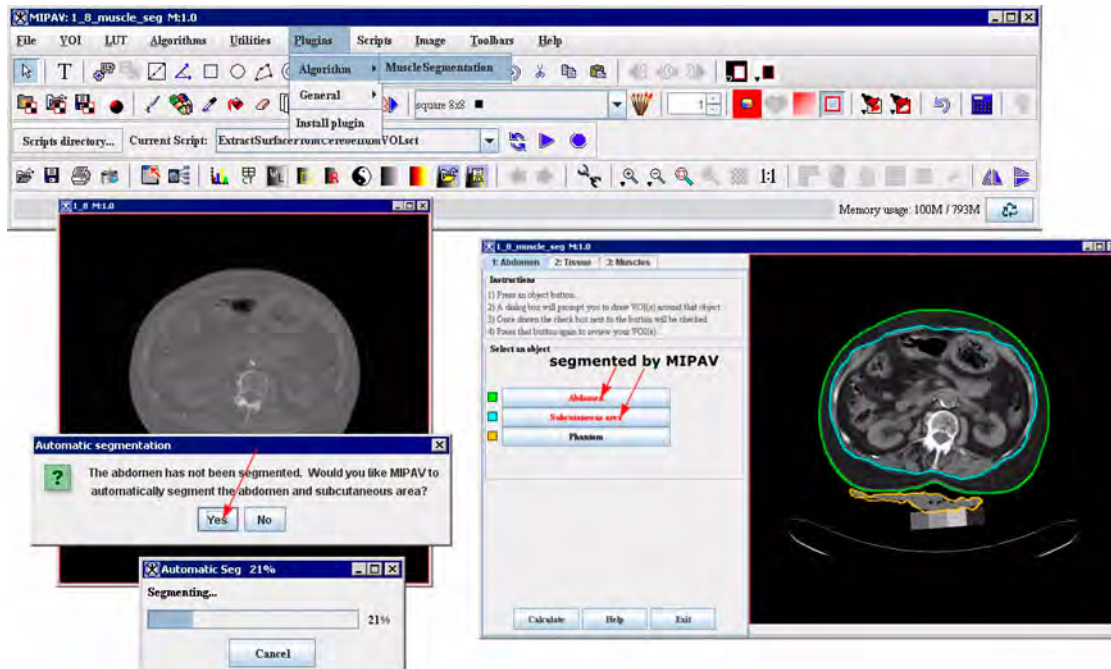


Figure 13. The MIPAV automatic segmentation of the abdomen and subcutaneous area

For example, Figure 13 shows the Abdomen tab with the Abdomen and

Subcutaneous area buttons appear in red, which means that the VOIs for these areas were created by MIPAV. See also Figure 9.

## USING THE MIPAV VOI STATISTICS GENERATOR

To view the VOI statistics you can use the Analysis tab options (refer to Section Analysis tab) or run the VOI Statistics Generator from the main MIPAV menu.

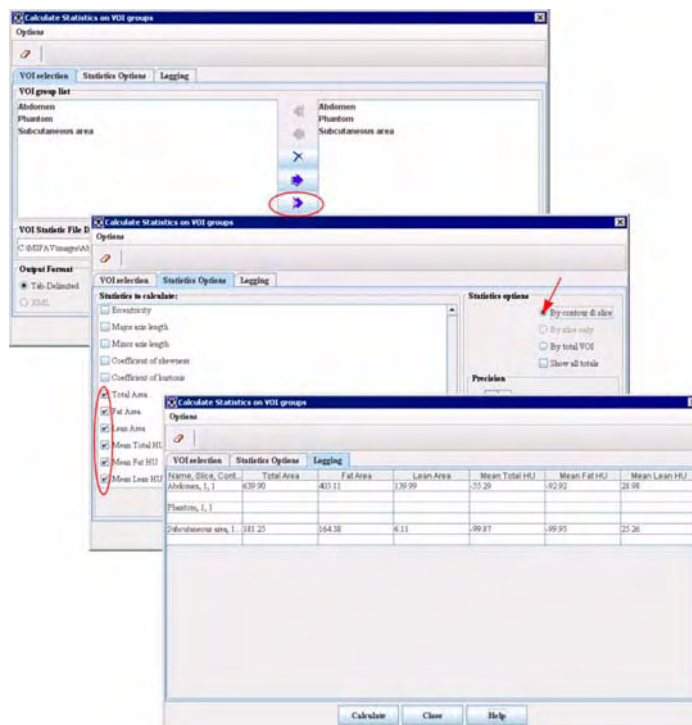


Figure 14. Using the MIPAV Statistics Generator

To run the VOI Statistics Generator, call VOI > Statistics Generator menu. The Calculate Statistics on VOI groups dialog box appears.

In the dialog box,

- 1** In the VOI Selection tab, select all VOI groups;
- 2** Then, navigate to the Statistic Options tab. Scroll down to the end of list, and then check the following statistics calculation options: Total Area, Fat Area, Lean Area, Mean Total HU, Mean Fat HU, Mean Lean HU. Note that these options only appear if you run the Muscle Segmentation plug-in;
- 3** Select the By Contour and Slice option;
- 4** Press Calculate;
- 5** The fat/ lean statistics for selected abdomen objects appears in the Logging tab. See Figure 14.

---

## ANALYSIS TAB

The Analysis tab appears when you click the Calculate button. It has 12 buttons; each button possesses the name corresponding to the particular VOI.

To run the fat/lean ratio calculation

- 1** Select the abdomen object(s) (the abdomen, subcutaneous area, visceral cavity, etc.) by pressing the corresponding button. Note that you can select as many objects as you wish.
- 2** If you want to run the calculation for a single object, for example for the visceral cavity, select that object, and then press Output. The statistics appears in the output window.
- 3** If you want to calculate a fat/ lean statistics for a set of abdomen objects, select these objects by pressing the corresponding buttons, and then press Output. The fat/ lean statistics appears in the output window.
- 4** If you want to calculate a fat/ lean statistics for all objects, press Output All. This will output a whole fat/lean statistics in the Output window. See Figure 15.



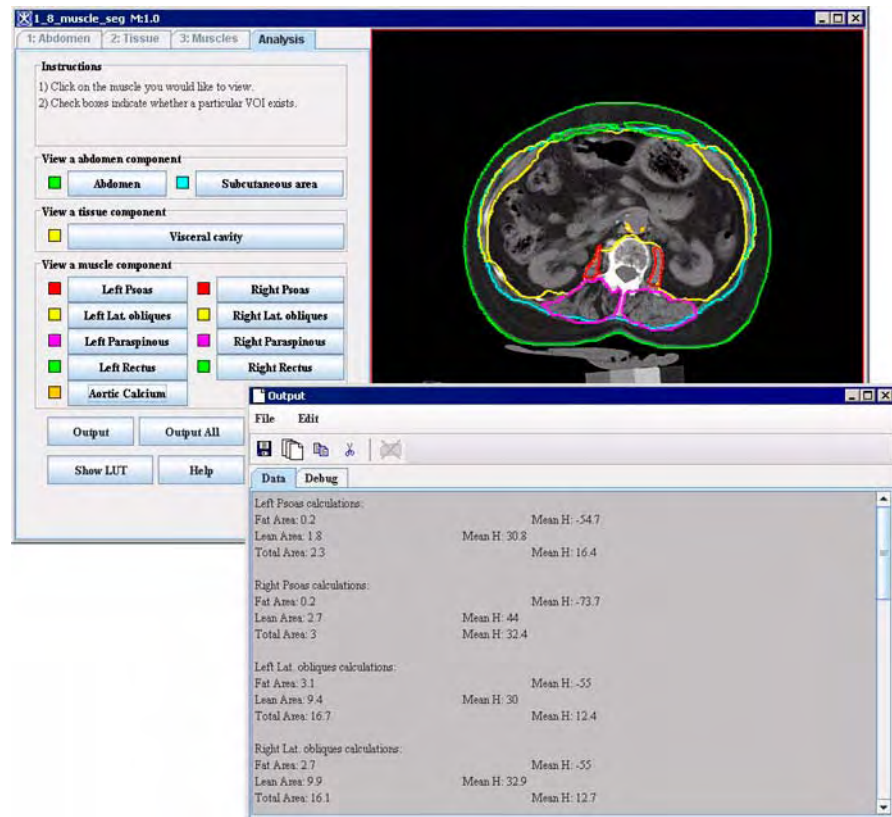


Figure 15. The Analysis tab

## SAVING THE STATISTICS AS A PDF FILE

To save the output statistics as a PDF file, press the Save button. The data from the Output window will be saved as a PDF file in the folder where VOI files are stored. The information window will appear displaying the path to the PDF file and the file name. See also Figure 16 and Figure 17.

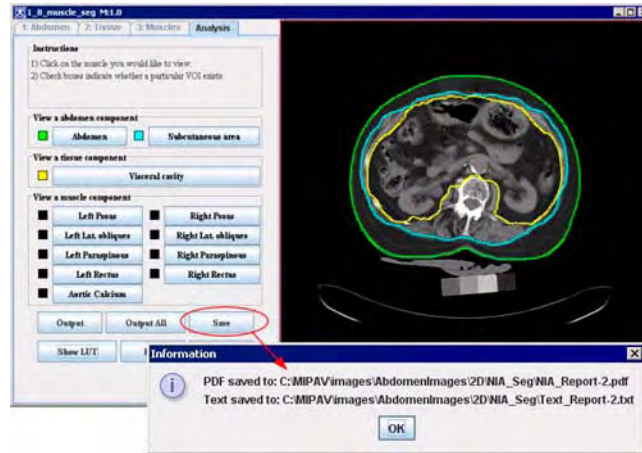


Figure 16. Saving the statistics as a PDF file

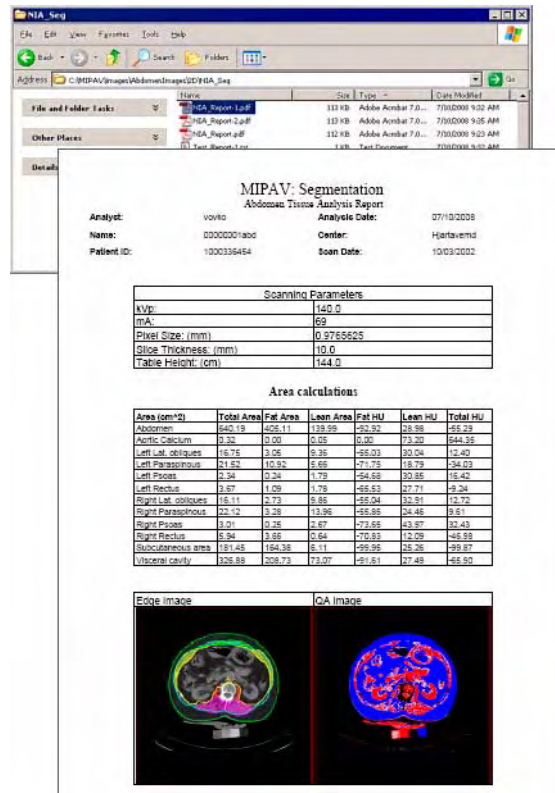


Figure 17. The MIPAV Segmentation PDF file

## TO SHOW AND HIDE LUT

To show the LUT, press Show LUT button, the image appears highlighted by LUT. To hide the LUT, press Hide LUT. See Figure 18.

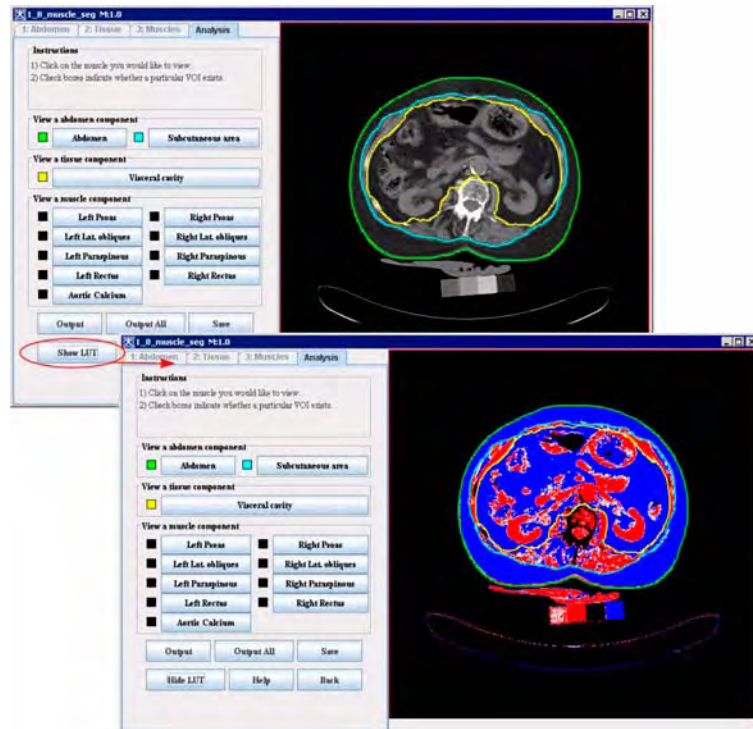


Figure 18. Showing the LUT for selected VOIs

---

## Optimized Automatic Registration 3D

*The Optimized Automatic Image Registration method in MIPAV determines a transformation that minimizes a cost function, which represents the quality of alignment between two images. The standard registration problem of computing a transformation that best aligns a reference image to a target image is formulated as a mathematical optimization problem of determining the global minimum of a cost function. The method evaluates the cost function at several different image resolutions, starting with the lowest resolution. Each step of increasing resolution utilizes the previously determined optimal transformation as a starting point and further refines its values. This method usually works well with the images of the same modality.*

### Background

The main approach for determining an optimal transformation is to calculate a cost function, determine how it changes as individual transformation parameters are varied, and find the parameters that minimize the value of the cost function. For instance, Figure 1 shows a plot of a sample cost function for a selection of transformation parameters; in each plot, a single parameter is varied while all other are kept constant.

Another fundamental step in this registration method is to resample and interpolate the reference and target images at several different resolutions (starting with a coarse voxel size and gradually increasing the resolution), while searching for the *min* cost function.

---

An advantage of this multi-resolution approach is that the initial optimization considerably reduces computational cost, since the number of sample points is substantially less. In addition, the overall alignment is easier to find at a coarse resolution. Improved accuracy of the transformation parameters is achieved using the highest resolution image that has not been subsampled.

---

The resampling process may influence the computed value of the cost function; therefore, several different interpolation methods are incorporated with this technique. These methods are discussed in more detail later in “Resampling Interpolation” on page 605.

Taking into account the resampling and interpolation criteria, the Optimized Automatic Registration method begins searching for an optimum transformation using the Powell method at a coarse resolution. Once a transformation has been determined that minimizes the cost function, the transformation is further refined by progressively increasing the resolution.

The algorithm offers various resampling interpolation methods combined with the Correlation Ratio, Least Squares, Normalized Cross Correlation, and Normalized Mutual Information as a cost function.

See also “Cost functions used in MIPAV algorithms” .

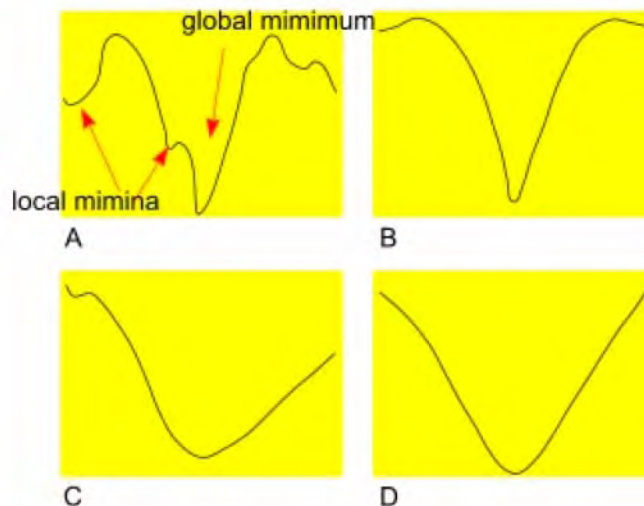


Figure 1. The plots of the Correlation Ratio cost function versus some of the individual parameter values. In each plot (A, B, C, and D), a single parameter is varied while the other are kept constant

---

## IMAGE TYPES

You can apply Optimized Automatic Registration method to color, grayscale and black-and-white 2D and 3D images.

---

## OUTLINE OF THE METHOD

### Pre-optimization steps

- 1** The minimum resolution of the reference and target images is determined. For more information, refer to “Blurring” on page 599.
- 2** Both images are resampled and interpolated to create high resolution isotropic voxels. The following interpolation methods are available: trilinear (a default), B-spline 3-rd and 4-th order, 4, 5, 6 Lagrangian, and Windowed sinc. Refer to “Interpolation methods used in MIPAV” on page 74.
- 3** The center of mass (COM) for both images is then computed and one translation step is performed to align the COM. Refer to “Calculating the center of mass” on page 600.

### Optimization steps

As a part of the transformation optimization process the images are subsampled by 8, 4, and 2 times.

- 4 levelEight optimization.** Both images are subsampled and interpolated, so that each image is 8 times smaller. The parameters of the transformation are then systematically varied, where the cost function is evaluated for each setting. The parameters corresponding to the smallest value(s) of the cost function are then used as the initial transformation for the next level in the optimization.
- 5 levelFour optimization.** The images are subsampled and interpolated, so that each image is 4 times smaller. And the transformation corresponding to the smallest value(s) of the cost function are determined and used as the initial transformation for the next level in the optimization.
- 6 levelTwo optimization.** Similar processing as the previous two levels except the images are, first, subsampled and interpolated, so that each image is 2 times smaller.
- 7 levelOne optimization** – 1mm resolution images are used in this step and the transformation is generalized to include 12 degrees of freedom.

## Post optimization

- 8 Using the same interpolation methods as described in Step 2 and the optimal transformation determined above, the method transforms the reference image into the same coordinate system of the target image.

Following Sections 1.2-1.5 describe in detail the algorithms and methods used to perform Steps 1-8 of the optimization routine.

---

## PRE OPTIMIZATION

### Blurring

The registration algorithm first, determines the minimum resolution for each dimension of the reference and target images. The reference image is blurred to the resolution of the target image if one of the resolutions in a dimension of the target image is 50% or larger than the corresponding resolution in the reference image. Likewise, the target image is blurred to the resolution of the reference image if one of the resolutions in a dimension of the reference image is 50% or larger than the corresponding resolution in the target image. It is possible (but not likely) that both the reference and target images will be blurred.

---

See also “Resampling Interpolation” on page 605.

---

### Resampling

Once this isotropic 1mm resolution images have been obtained, the subsampled versions are also created. The sub-sampling algorithm then simply keeps every n-th point (that is, 2, 4 or 8) on the lattice in each direction. Therefore, the new volume contains  $1/n^3$  as many points as the original.

---

Resampling can be performed for x, y, and z axes for 3D images, or for only x and y for 2D images.

---

## CALCULATING THE CENTER OF MASS

To calculate the center of mass (COM), the method uses the right hand convention in 3D coordinate systems (X, Y, Z). Thus, in the image space, the left hand corner of the image is set to (0,0,0). The x axis goes left to right, y axis goes top to bottom and z axis goes into the screen.

- 1 To calculate the COM, we, first, define the characteristics function of an object in an image to be as described in Equation 1:

EQUATION 1

$$b(x,y) = \begin{cases} 1, & \text{for points in the image} \\ 0, & \text{for background points} \end{cases}$$

- 2 Then, an area of an image can be calculated as

EQUATION 2

$$A = \int \int b(x,y) dx dy$$

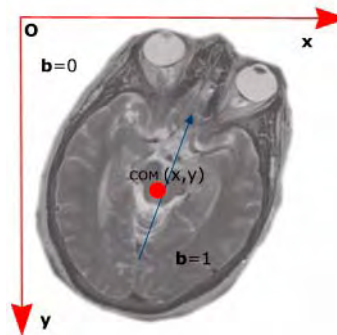


Figure 2. Calculating the center of mass (COM)

- 3 And the center of mass, denoted by  $(x_{COM}, y_{COM})^1$  is given by the 1-st moments of the object:

1. For 2D images. For 3D images, add the z coordinate also.



$$x_{COM} = \frac{\int \int xb(x,y)dxdy}{\int \int b(x,y)dxdy}$$

$$y_{COM} = \frac{\int \int yb(x,y)dxdy}{\int \int b(x,y)dxdy}$$

---

## PROCESS

The process used in this method performs the following:

- Calculates the COM for both images as described above in “Calculating the center of mass” on page 600;
- Aligns the COM of both images, which is an initial transformation that involves only translation;
- For each resampled image (which is 8, 4, 2, and 1 times), determines the transform that minimizes the cost function.

### levelEight optimization

levelEight optimization is primarily concerned with determining the rotational transformation that will lead to the globally optimal transformation, which corresponds to the global minimum of the cost function. levelEight optimization uses the lowest resolution images and coarse rotation angles evaluated at relatively large step sizes.

### Optimization

This Optimized Automatic Registration method is based on the empirical observation that finding the correct orientation, or rotation, is the most difficult task in image registration, since the three rotation parameters are highly coupled and most of the erroneous registrations that have been examined have happened primarily due to an incorrect orientation. Therefore, the search concentrates on the rotational part of the transformation.

Specifically, the reference image is oriented, interpolated, and the cost func-

tion evaluated for coarse rotations angles  $(-30, -30, -30)$ ,  $(-30, -30, -15)$ , ...,  $(-30, -30, 30)$ , where the values represent the amount of rotation in degrees about the x, y, and z axes respectively. Since there are three rotation angles and each angle can contain five different values, there are 125 possible angle configurations.

For each angle configuration, a 4-DOF local optimization is also performed to find the optimal translation and (global) scale. By default, the initial values are set to

- 0 for translation
- 1 for scale.

The best 20% of the cost values and/or the corresponding angle configurations (candidate local minima) are stored in a vector of minima that is used as a starting point for a further optimization, which uses a smaller step size over a narrower range of angles. Refer to Figure 3.

For each parameter setting corresponding to the top 20% of the cost function minima, the algorithm performs rotations over the fine grid (which is by default 15 degrees with 6 degrees step) and evaluates the cost function.

The method now finds all angle configurations that have the corresponding cost function values lower than their neighbors from the vector of candidate local minima.

For each of these sets of parameters a 7-DOF optimization is then performed, storing the results of the transformation and costs *before* and *after optimization* in a vector of minima. See also “Degrees of freedom” on page 606.

Since the relative costs of each candidate solution may change at higher resolutions, where structures become more sharply defined, the top 20% of the cost function minima points are considered for the next higher resolution (4) stage, rather than just a single best solution.

By default, the algorithm uses Trilinear interpolation to transform images to this new orientation.

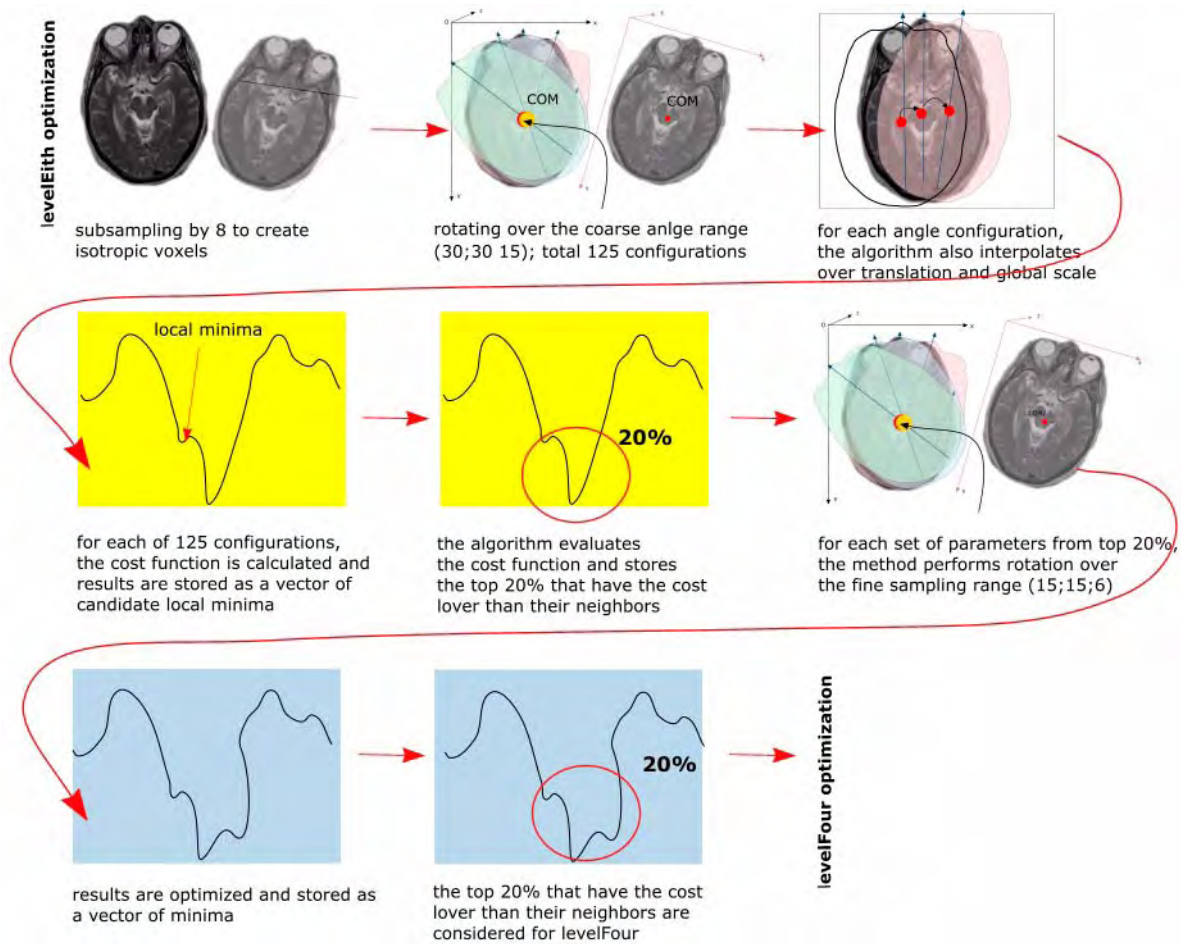


Figure 3. levelEight optimization

### levelFour optimization

The levelFour optimization step uses the interpolated images subsampled by 4 to determine the transformation that minimizes the cost function starting with the transformations determined in levelEight.

The optimization determines a 7-DOF transformation that corresponds to the minimum value of the cost function. This transformation is then perturbed and the cost function is computed for these new settings. The perturbations correspond to six-degrees for each rotation parameter and a global scaling factor of 0.8, 0.9, 1.1, and 1.2. A type of transformation (translation and/or global scale) is specified by a user. It includes the following transformations: Affine-12 (a default), Rigid-6, Global Rescale-7, and Specific Rescale-9. Here, numbers indicate degree of freedom (DOF).

---

A vector of parameters and top 20% of the *min* cost function values are considered for the next step, which involves images subsampled by 2.

---

See also “Cost functions used in MIPAV algorithms” on page 63 and “Degrees of freedom” on page 606

---

### levelTwo

In levelTwo, the method uses the images subsampled and interpolated by 2. The cost function is evaluated using the transformation parameters corresponding to the top 20% minima obtained from the levelFour. It finds the best minimum, and then optimizes it with the 7-DOF and then 9-DOF transformation. The method then returns the best minimum after optimization.

---

**Note:** if a user has limited the degrees of freedom to six, as for the rigid transformation, there will only be one optimization run, with six degrees of freedom.

---

### levelOne

This step uses the unsubsampled interpolated images and computes the value of the cost function for each parameter setting determined in the previous levelTwo optimization. The parameters corresponding to the minimum value of the cost function with this resolution are then further optimized for transformations with 7-, 9-, and 12- DOF, unless a rigid (6-DOF) transformation is desired.

### What is saved

The method stores

- The set of parameters or vector at which the minimum was reached;
- The value of the cost function at that minimum;
- And the matrix, which was the true input into the cost function and represents the transformation that gives the minimum cost of differences between the images.

## Additional registration parameters

To better align the images, one can also include into calculation additional registration parameters, such as weighting coefficients and advanced parameters used for calculating the min cost function. For more information, refer to “Applying the AlgorithmOAR 3D” on page 607.

---

## RESAMPLING INTERPOLATION

The Optimized Automatic Registration method resamples images using an interpolation scheme, where anisotropic voxels in the Z direction are resampled into isotropic 1mm cubic voxels. New voxel values are computed using a weighted combination of existing voxel values within a defined neighborhood of the new voxel location. The interpolation methods are available in this implementation of the registration technique, include Trilinear, B-spline 3-rd order, B-spline 4-th order, Cubic Lagrangian, Quintic Lagrangian, Heptic Lagrangian, and Windowed sinc.

For mor information about the interpolation methods used in Optimized Automatic Registration, refer to “Interpolation methods used in MIPAV” .

## Cost Functions

A *similarity or cost function* measures the similarity between two images. During the Optimized Automatic Registration the adjusted image V is transformed using the transformation functions described above. And the similarity  $S(U;V^t)$  between the reference image U and transformed image  $V^t$  is then calculated. The Optimized Automatic Registration method searches for the transformation that gives **the smallest value of the cost function**, which we assume is the transformation that also gives the best alignment.

The cost functions which are implemented in this method include:

- Correlation ratio, refer to “Cost functions used in MIPAV algorithms” , “Correlation ratio” on page 64
- Least Squares, refer to “Cost functions used in MIPAV algorithms” , “Least Squares” on page 66

- Normalized Cross Correlation, see “Cost functions used in MIPAV algorithms” , “Normalized cross correlation” on page 67
- Normalized Mutual Information, see “Cost functions used in MIPAV algorithms” , “Normalized mutual information” on page 70.

---

## POWELL ALGORITHM

The Optimized Automatic Registration method uses the Powell algorithm to find the global minimum of the chosen cost function. For more information about the Powell algorithm, refer to the following link <http://math.fullerton.edu/mathews/n2003/PowellMethodMod.html>

---

## DEGREES OF FREEDOM

The number of independent pieces of information that go into the estimate of a parameter is called the degrees of freedom (DOF).

For more information refer to “Cost functions used in MIPAV algorithms” , “Degrees of freedom” on page 71.

---

## REFERENCES

Björn Hamre “Three-dimensional image registration of magnetic resonance (MRI) head volumes” Section for Medical Image Analysis and Informatics Department of Physiology & Department of Informatics University of Bergen, Norway.

Bourke Paul “Interpolation methods” <http://local.wasp.uwa.edu.au/~pbourke/other/interpolation/index.html>

B-splines: <http://www.cl.cam.ac.uk/teaching/1999/AGraphHCI/SMAG/node4.html>

Chapter 3: "Registration Methodology: Concepts and Algorithms" by Derek L.G.Hill and Philippe Batchelor in *Medical Image Registration* edited by Joseph V. Hajnal, Derek L.G.Hill, and David J. Hawkes, CRC Press, 2001, pp. 40-70.

Chapter 33 "Within-Modality Registration Using Intensity-Based Cost Functions" by Roger P. Woods in *Handbook of Medical Image Processing and Analysis*, Editor, Isaac N. Bankman, Academic Press, 2000, pp. 529-553.

FLIRT, visit their homepage at <http://www.fmrib.ox.ac.uk/fsl/flirt/>

Jenkinson, M. and Smith, S. (2001a) "A global optimisation method for robust affine registration of brain images". *Medical Image Analysis*, 5(2):143-156.

Jenkinson Mark and Stephen Smith "Optimisation in Robust Linear Registration of Brain Images" FMRIB Technical Report TROOMJ2. <http://www.fmrib.ox.ac.uk/fsl/flirt>

Josien P. W. Pluim, J. B. Antoine Maintz and Max A. Viergever. *Mutual information based registration of medical images: a survey*. IEEE TRANSACTIONS ON MEDICAL IMAGING, VOL. XX, NO. Y, MONTH 2003.

Powell method to find the global minimum: <http://math.fullerton.edu/mathews/n2003/PowellMethodMod.html>

Stephen M. Smith, BET: "Brain Extraction Tool", FMRIB technical Report TR00SMS2b, FMRIB (Oxford Centre for Functional Magnetic Resonance Imaging of the Brain), Department of Clinical Neurology, Oxford University, John Radcliffe Hospital, Headington, Oxford OX3 9DU, UK.

Thomas M. Lehmann,\* Member, IEEE, Claudia Gonner, and Klaus Spitzer. *Survey: Interpolation Methods in Medical Image Processing*. IEEE TRANSACTIONS ON MEDICAL IMAGING, VOL. 18, NO. 11, NOVEMBER 1999 1049.

## Applying the AlgorithmOAR 3D

**To run the algorithm, complete the following steps:**

- 1** Open an input and reference image.
- 2** Call Algorithms > Registration > Optimized Automatic Registration.
- 3** The Optimized Automatic Registration dialog box appears.

In the dialog box,

- Select the image in the Register to box;
- Complete the rest of the dialog box options. Refer to “” on page 608.

- 4** Press OK.

The algorithm begins to run and the Registering Images window appears with the status. When the algorithm stops running the registered image appears in a new image frame. Refer to Figure 4.

---

**Recommendation:** First, use the default algorithm settings. And then, after completing the first registration run, modify them depending on the result you receive.

---



Figure 4. The input image (A), reference image (B) and result image (C). Registration was performed using the default algorithm settings.

## Applying the Constrained AlgorithmOAR 3D

To run the algorithm, complete the following steps:

- 1** Open an input and reference image.
- 2** Call Algorithms > Registration > Constrained Optimized Automatic Registration.
- 3** The Optimized Automatic Registration dialog box appears.

In the dialog box,

- Select the image in the Register to box;
  - Complete the rest of the dialog box options. See Figure 5;
  - Press the Advanced Settings button, and then complete the Advanced OAR settings dialog box. In this dialog box, you can limit the number of iterations and also the translation's range. Refer to Figure 6 on page 612 for the dialog box options.
- 4** Press OK.

The algorithm begins to run and the Registering Images window appears with the status. When the algorithm stops running the registered image appears in a new image frame.



**Recommendation:** For the Constrained Optimized Automatic Registration algorithm, first, use the default algorithm settings. And then, after completing the first registration run, modify them (including Advanced settings) depending on the result you receive.

## OPTIMIZED AUTOMATIC REGISTRATION DIALOG BOX OPTIONS

Input options	
<b>Register</b>	Select the reference image from the list. The list alphabetically displays all opened images.
<b>Degrees of freedom</b>	Specifies the spatial model used to restrict the type of linear transformation being applied during registration. Select one of the following: rigid-6, global rescale-7, specific rescale-9, and affine-12.
<b>Interpolation</b>	Specifies the type of interpolation the algorithm will use when resampling. The list includes: <ul style="list-style-type: none"> <li>• Trilinear</li> <li>• B-spline 3-rd order</li> <li>• B-spline 4-th order</li> <li>• Cubic Lagrangian</li> <li>• Quintic Lagrangian</li> <li>• Heptic Lagrangian</li> <li>• Windowed sinc</li> </ul> See also "Interpolation methods used in MIPAV" .

Figure 5. The Optimized Automatic Registration dialog box options

<b>Cost function</b>	Specify a cost function which will be used to evaluate registration. The list includes: Least squares, Correlation ratio, Normalized cross correlation, Normalized mutual information. For more information, refer to "Cost functions used in MIPAV algorithms" on page 63.
<b>Use the max of the min resolutions of the two datasets when resampling</b>	If this option is chosen in the dialog box, the method uses the maximum resolution of the two datasets. It throws away some image information, but works faster. If this option is not chosen the algorithms uses the minimum of the resolutions when resampling the images. This can be slower, but does not lose the information.
<b>Initiate registration process by applying Least Squares</b>	This option provides a way to register images using the corresponding landmark points or VOIs placed in both images (you must put at least 4 landmarks). Note that this option works only for images that require rigid transformation e.g., rotation and/or translation, but it does not work on images that require scaling.
<b>Rotations</b>	
Use this option if there is a need to make manual adjustments to registration.	
<b>Apply same rotations to all dimensions</b>	If checked, applies rotation that you specified to all dimensions. Otherwise, you should select the axis – X, Y, or Z – to which rotation should apply.
<b>Coarse angle increment and Fine angle increment</b>	
options can be used for two pass registration. The first pass would apply a coarse-level rotation to quickly arrive at an approximate registration, and then the second pass would apply a finer-level rotation to continue from the point where the first pass registration has finished. By default, the coarse angle increment is set to 15 degrees and fine angle increment is set to 6 degrees.	
<b>Weighted Images</b>	This gives weights for certain areas within the reference or transformed image. These areas can be specified using VOIs or reference files. Here, <b>higher weights mean a greater impact in that area on the registration.</b>
<b>No weight</b>	– the weighting factor is not used for registration.
<b>Register area delineated by VOIs only</b>	uses VOI to register a selected area.
<b>Weight registration</b>	If this option is checked, the following parameters become available: <b>Choose ref. weight</b> opens the dialog where you can select the reference file which contains the information about the weighting factor. <b>Choose input weight</b> opens the dialog where you can select the input file which contains the information about the weighting factor.
<b>Output options</b>	
<b>Figure 5. The Optimized Automatic Registration dialog box options (continued)</b>	

<b>Display transformed image</b>	allows to view the transformed image in a separate window. By default, this option is active.
<b>Interpolation:</b>	– select the calculation method, which is used to produce the output image from the array of transformation matrices and input image. You can choose among Trilinear, B-spline 3-rd order, B-spline 4-th order, Cubic lagrangian, Quintic lagrangian, Heptic lagrangian, or Windowed sinc. The default choice is trilinear.
<b>Advanced options</b>	
Pressing the Advanced button opens the Advanced OAR Settings dialog box where you can specify the additional parameters which will be used to calculate the min cost function. Parameters are as follows:	
<b>Multiple of tolerance to bracket the minimum</b>	Recommended values are 10–60.
<b>Number of iterations</b>	Recommended are 1–5 iterations.
<b>Number of minima from Level 8 to test at Level 4</b>	By default this is the best 20%, but you can enter your own number here.
<b>Subsample image for speed</b>	subsamples the image.
<b>Skip multilevel search. Assume images are close to alignment.</b>	
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes it.
<b>Help</b>	Displays online help for this dialog box.

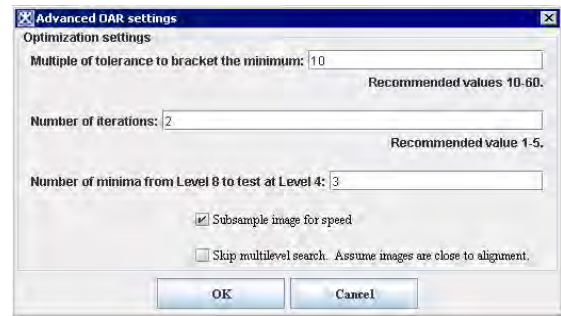


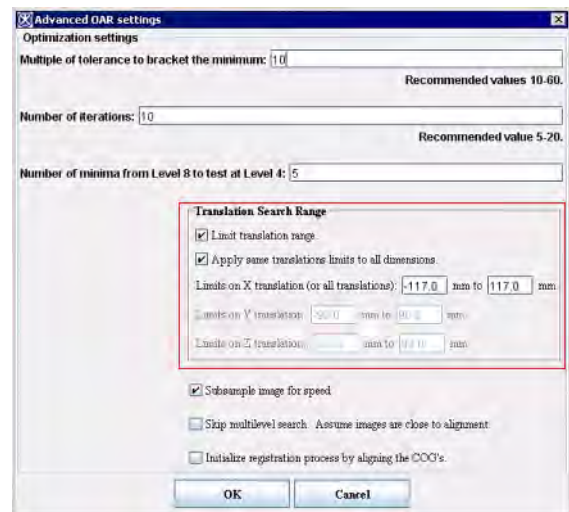
Figure 5. The Optimized Automatic Registration dialog box options (continued)

## ADVANCED OAR SETTINGS FOR CONSTRAINED OPTIMIZED AUTOMATIC REGISTRATION 3D

Pressing the Advanced button in the main Optimized Automatic Image Registration 3D dialog box (see Figure 5) opens the Advanced OAR Settings dialog box where you can specify the additional parameters and (or) constrains (if Constrained AlgorithmOAR 3D was called) that will be used to calculate the min cost function. Parameters are as follows:

<b>Multiple of tolerance to bracket the minimum</b>	Recommended values are 10–60.
<b>Number of iterations</b>	Recommended are 1–5 iterations.
<b>Number of minima from Level 8 to test at Level 4</b>	By default, this is set to 5, but you can enter your own number here.
<b>Translation search range</b>	These options allow the user to specify the limits in translations range (in the X, Y, and Z dimensions), and therefore, constrain the optimized registration. The options appear active only if the Constrained Optimized Registration algorithm is called.
<b>Limit translation range</b>	If checked, this option limits the range of translations by the values (in mm) specified by the user.
<b>Apply same translation to all dimensions</b>	If checked, this option applies same limits to all dimensions.
<b>Limits on X translation (or all translations)</b>	Specify the translation limit in X direction, or in all directions if the <b>Apply same translation to all dimensions</b> option is checked.
<b>Limits on Y translation</b>	Specify the translation limit in Y direction.
<b>Limits on Z translation</b>	Specify the translation limit in Z direction.
<b>Subsample image for speed</b>	subsamples the image.
<b>Skip multilevel search. Assume images are close to alignment.</b>	

**Figure 6. The Advanced Optimized Automatic Registration (constrained) dialog box options**



<b>Initialize registration process by aligning COG's</b>	If checked, the algorithm, first, aligns centers of gravity (COG) or centers of mass (COM) of both images, and then proceed with registration. See also "Calculating the center of mass" on page 600.
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes it.
<b>Help</b>	Displays online help for this dialog box.

**Figure 6. The Advanced Optimized Automatic Registration (constrained) dialog box options (continued)**

---

## Registration: Patient Position (DICOM)

The method uses the image origin coordinates and image orientations to align DICOM images based on a patient position.

### Background

The algorithm extracts the images orientation and origins coordinates from the DICOM files headers, and then uses them to register the images. In the DICOM protocol, image orientation specifies the direction cosines of the first row and the first column with respect to the patient and is defined as two unit vectors in the patient coordinate system. These were used as an initial estimate of the orientation parameters, with the initial translation parameters set to zero.

The DICOM standard uses a patient-centric LPS, coordinate system, where  $+x = \text{left}$ ,  $+y = \text{posterior}$ , and  $+z = \text{superior}$ . The direction cosines between the rows and columns of the image and these axes are stored in the Image Orientation Patient field. For axial images with no rotation, this field will be 1/0/0/0/1/0. This may be read as "image rows increase from right to left, image columns increase from anterior to posterior". Most DICOM viewers follow the convention of plotting the data starting in the upper left corner of the view.

For axial brain images, this will place the left side of the brain on the right side of the view, and the posterior of the brain toward the bottom of the view. This point of view is sometimes referred to as "radiological".

Sagittal will have an Image Orientation Patient field of 0/1/0/0/0/-1, indicating that rows increase from anterior to posterior, and columns increase from superior to inferior. In this case, the posterior will be toward to the right and the inferior toward to the bottom.

Coronal images will have an Image Orientation Patient field of 1/0/0/0/0/-1.

---

## OUTLINE OF THE METHOD

"The algorithm reads origins coordinates from file headers.

"Then, it extracts the images orientation from DICOM files.

"Based on the extracted information, the algorithm obtains the transformation matrix from orientation of ImageB to orientation of ImageA.

"Then, it creates an index arrays that stores the reordering of Image B data (i.e. origin, resolution, dimension) to the ordering of Image A.

"It creates a model image that is ImageB transformed to coordinates of ImageA

"For that image, the algorithm updated resolutions and dimensionality for ImageB so that they correspond to the new ordering.

"It pads or crops the model image so that its size corresponds to the size of ImageA.

"The algorithm finds the optimal transformation  $T:(x, y, z) \rightarrow (x', y', z')$  (same for all time frames), which will map any voxel in ImageB to its corresponding voxel in ImageA. This transformation consists of three translations ( $T_x, T_y, T_z$ ) and three rotations ( $R_x, R_y, R_z$ ).

TBD. BEFORE AND AFTER GRAPHIC (WHAT HAPPENS WHEN ALGORITHM IS RUN)?

Figure 1. Patient Position (DICOM) algorithm processing

---

## IMAGE TYPES

TBD.

---

## SPECIAL NOTES

TBD.





## Registration: Landmark—Least Squares

*The Registration: Landmark—Least Squares algorithm provides a way for registering an image to the reference image by using corresponding points placed in both images. This algorithm works on images requiring rotation and translation, but it does not work on images requiring scaling, because this is a rigid transformation based on two homologous landmark datasets.*

### Background

The algorithm relates the two corresponding point sets on the image to be registered and on the reference image using the following equation:

$$pA[i][j] = R(pB[i][j]) + T + No[i][j]$$

where

$pA[i][j]$  = the point set of the reference image

$pB[i][j]$  = the point set of the image to be registered

$i$  = a value from 0 to the number of dimensions - 1

$j$  = a value from 0 to the number of user-placed corresponding points - 1

$R$  = a rotation matrix

$T$  = a translation vector

$No[i][j]$  = a noise vector

This algorithm calculates a least-squares solution of  $R$  and  $T$ , which is based on a singular value decomposition of a  $2 \times 2$  matrix in 2D or a  $3 \times 3$  matrix in 3D. It, first, calculates the centroid  $p1[i]$  in the image to be registered:

$$p1[i] = \frac{1}{N} \cdot \sum_{j=0}^{N-1} pB[i][j]$$

It, then, calculates the centroid  $p2[i]$  in the reference image:

$$p2[i] = \frac{1}{N} \cdot \sum_{j=0}^{N-1} pA[i][j]$$

where  $N$  = the number of corresponding user points.

The algorithm, next, calculates the difference between the points and the centroid voxel in the image to be registered ( $q1[i][j]$ ):

$$q1[i][j] = pB[i][j] - p1[i]$$

and then the difference between the points and centroid voxel in the reference image ( $q2[i][j]$ ):

$$q2[i][j] = pA[i][j] - p2[i]$$

The algorithm then performs the following steps:

- Calculates the 2 x 2 or 3 x 3 matrix:

$$H = q1(q2 \text{ transpose})$$

- Performs the singular value decomposition of  $H$ . That is:

$$\text{Express } H = U(\Lambda)(V \text{ transpose})$$

- Calculates the following:

$$X = V(U \text{ transpose})$$

- Calculates the determinant of  $X$  (refer to the following table).

If the determinant is 1, the algorithm succeeded and the rotation matrix  $R = X$ .

If the determinant is -1 and the images are 3D, then the singular values of matrix  $H$  must be examined. If one and only one of the three singular values is 0 and the two singular values are unequal, then the case is coplanar, but not colinear. It is made to work by simply changing the sign of the third column of the  $V$  matrix to form  $Vp$  and obtain  $R = Vp(U \text{ transpose})$ .

- Calculates the translation vector:

$$T[i] = p2[i] - R(p1[i])$$

- Sets the first dim elements of the last column of the transformation matrix to  $T$ .
- Copies  $R$  into the upper left dim by dim elements of the transformation matrix.
- Sets the last row of the transformation matrix to 0,0,1 in the 2D case or to 0,0,0,1 in the 3D case.
- A bilinear or trilinear transformation is used with the given transformation matrix to transform the target image into the resulting image (Figure 1). Refer to “Interpolation methods used in MIPAV” .

**Table 64.1 A determinant of X depending on the image type**

Image type	If this is true . . .		Then . . .		Comments
	Determinant	Matrix H	Success	Failure	
2D images	1		X		Rotation matrix $R = X$
	-1	Has no zero singular values		X	Data too noisy
	-1	Has one singular value		X	Data is colinear
3D images	1		X		
	-1	Has no zero singular values		X	Data too noisy
	-1	Has one zero singular value and the other two singular values are unequal	X		This is made to work by changing the sign of the third column of the $V$ matrix to form $Vp$ and obtain $R = Vp(U^{\text{transpose}})$
	-1	Has two equal singular values		X	Colinear

## IMAGE TYPES

You can apply this algorithm to 2D and 3D grayscale and color images.

## SPECIAL NOTES

The following notes apply:

- Do not use this algorithm in cases where an image must be scaled.
- For 2D images, you must identify at least three noncollinear points.
- For 3D images, you must identify at least four noncollinear points.



**Figure 1. Landmark—Least Squares processing**

*Both the reference image (A) and the image to be registered (B) have at least three noncollinear landmarks identified. Applying the Landmark—Least Squares algorithm produces the image shown in C.*

## REFERENCE

Refer to the following reference for more information:

“Least-Squares Fitting of Two 3-D Point Sets” by K.S. Arun, T.S. Huang, and S.D. Blostein, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-9, No. 5, September, 1987, pp. 698-700.

## Applying the Landmark—Least Squares Registration algorithm

To run this algorithm, complete the following steps:

- 1** Open two images: the target image and the reference image. When you apply this algorithm, MIPAV registers the target image to the selected reference image.
- 2** Perform, as an option, any other image processing, except scaling, on both images.
- 3** Place a minimum of three points in 2D images or four points in 3D images in the reference image. The points must not be colinear.

---

**Note:** Three-dimensional (3D) images can handle cases that are coplanar but not colinear.

---

- 4** Place the corresponding points in the target image. Note that the dialog box shows the last image clicked on as the target image.

---


**Tip:** Make sure that you delineate the points on the target image in the same order as on the reference image. That is, point 1 on the reference image should correspond to point 1 on the target image, point 2 on the reference image should correspond to point 2 on the target image, etc.

---

- 5** Select Algorithms > Registration > Landmark - Least Squares. The Least Squares Registration dialog box (Figure 2) appears.
- 6** Select the name of the target image in the **Register to** box if it is not already present.
- 7** Click OK.

The algorithm begins to run. A pop-up window appears with the status.

When the algorithm finishes running, the pop-up window closes, and the registered image appears in the Transformed Image window.

<b>Register [name of source image] to [name of reference image]</b>	<p>Displays a set of possible reference images.</p> <p>This box registers the image to be registered to the selected reference image.</p>	
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.	
<b>Cancel</b>	Disregards any changes that you made and closes the dialog box.	
<b>Help</b>	Displays online help for this dialog box.	

**Figure 2. Least Squares Registration dialog box**

---

## Registration: Landmark—TPSpline

The Landmark—TPSpline (Thin Plate Spline) algorithm allows you to align two or more images along a corresponding VOI that you delineate in all of the images.

### Background

When applied to 2D images, the Landmark—TPSpline algorithm uses a 2D interpolation scheme for arbitrarily spaced tabulated data  $(x_i, y_i, f(x_i, y_i))$ .

These splines are the generalization of the natural cubic splines in 1D. The spline surface represents a thin metal sheet that is constrained not to move at the grid points. When applied to 3D images, the TPSpline algorithm operate on data of the form  $(x_i, y_i, z_i, f(x_i, y_i, z_i))$ .

MIPAV uses the spline matching points sets to obtain spline interpolation coefficients, which it uses to transform all the *xorg, yorg* grid positions in the base image space to *xnew, ynew* grid positions in the match space. Then, at every *xorg, yorg* grid position in the base space, MIPAV checks to see if the corresponding *xnew, ynew* grid position in the match space is within the image bounds of the match image.

If *xnew* and *ynew* is within the match space bounds, then the data value at the *xnew, ynew* grid position in the match space is assigned to be the registered value at the *xorg, yorg* position in the base space. Since *xnew, ynew* is a floating point number and the data in the match image is only contained at integer grid points, interpolation must be used.

For a 2D image, the data value at *xnew, ynew* in the match space is obtained by bilinear interpolation from its four nearest neighbors. For a 3D image, the data value at *xnew, ynew* in the match space is obtained by trilinear interpolation from its eight nearest neighbors. If the *xnew, ynew* is outside the match space bounds, then a zero is assigned to the *xorg, yorg* position in the base space.

When processing the image, the algorithm first computes the thin plate spline coefficients and then generates match space grid positions from base space grid positions.

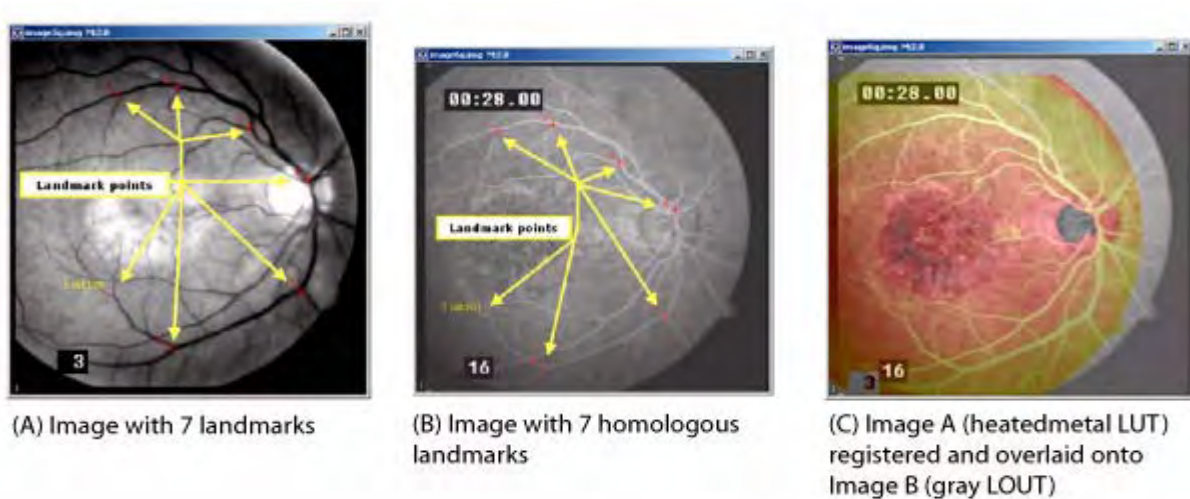


Figure 1. Example of Registration: Landmark—TPSpline algorithm

## IMAGE TYPES

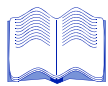
You can apply this algorithm to all 2D and 3D image types. The dimensions or image type of the match image need not be the same as the dimensions or image type of the base image. The registered image has the same image type as the match image and the same extents as the base image.

## SPECIAL NOTES

The following notes apply:

- For 2D images, three or more points are required. The algorithm may fail if nearly all of the points fall on the same line.
- For 3D images, four or more points are required and the algorithm may fail if the points nearly all fall on the same plane.

## REFERENCES



Refer to the following references for more information about this algorithm:




David Eberly, “Thin Plate Splines” of Magic Software at <http://www.magic-software.com>. Also at the same site are the files: MgcInterp2DThinPlateSpline.h, MgcInterp2DThinPlateSpline.cpp, MgcInterp3DThinPlateSpline.h, MgcInterp3DThinPlateSpline.cpp Paper that explains warping from one x,y source to another x',y target set.

Fred L. Bookstein, “Principal Warps: Thin-Plate Splines and the Decompositions of Deformations,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 6, June 1989, pp. 567–585.

## Applying the Registration: Landmark—TPSpline algorithm

To run this algorithm, complete the following steps:

Selected image is a match image, the image that gets transformed until it is registered to the base image.


- 1 Open two images: a base image and a match image.
- 2 Perform, as an option, any image processing, such as improving the contrast, eliminating noise, adding color, etc., on both images.
- 3 Delineate a point VOI on the base image using the point VOI icon  and mouse.
- 4 Select the match image.
- 5 Delineate a point VOI on the match image at corresponding positions on the image.



**Note:** Make sure that you delineate the points on the match image in the same order as on the base image. That is, point 1 on the base image should correspond to point 1 on the match image, point 2 on the base image correspond to point 2 on the match image, and so on.

- 6 Click Algorithm > Registration > Landmark - TPSpline. The Thin Plate Spline Registration dialog box opens.
- 7 Select the base image in the Register to list box.

**8** lick OK. The algorithm begins to run. A pop-up window appears with the status. The following messages appear in succession: “Performing base to match grid transformation” and “Performing interpolation into result buffer.” When the algorithm finishes running, the pop-up window closes, and the result appears in a new image window.

<b>Register to</b>	Displays a list of images including the base image and any match images.	
<b>OK</b>	Applies the mean algorithm to the base image that you chose in this dialog box.	
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes the dialog box.	
<b>Help</b>	Displays online help for this dialog box.	

**Figure 2. Thin Plate Spline Registration dialog box**

## Registration: Manual 2D Series

The Manual 2D Series algorithm allows you to manually or semimanually register two 2D grayscale or color images by placing corresponding points on both images and then applying either the Landmark—Thin Plate Spline or Landmark—Least Squares algorithm. After applying one of these algorithms, you can then, if needed, use the other algorithm. In addition, at any time, you can manually adjust the alignment of each image.

The algorithm also lets you manually register different slices of a 2.5 image dataset in the same way as the 2D images.

### Background

Through the Registration: Manual 2D Series window (Figure 1), this algorithm provides the tools for manually registering images and the ability to use two of the landmark registration algorithms. The Registration: Manual 2D Series window includes two tabs: Blended and Dual.

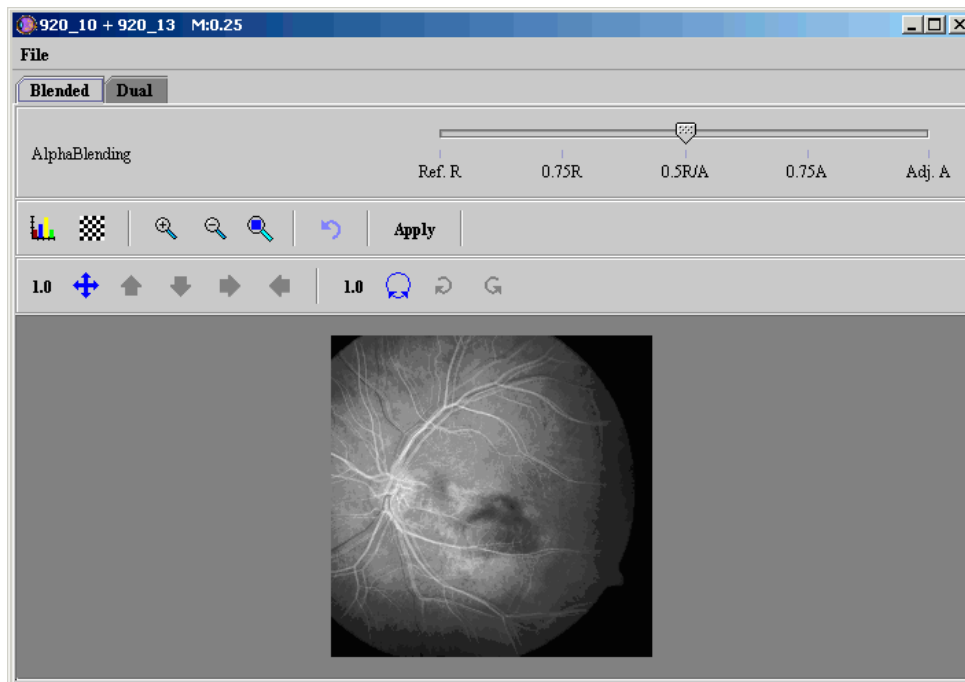





Figure 1. Registration: Manual 2D Series window


On the Blended page, the window superimposes the image to be registered on top of the reference image. This page includes icons for aligning, rotating, alphablending, and magnifying or minifying the images as well as icons for modifying the lookup table for each image and for applying a checkerboard pattern to assist in alignment.

Clicking the Dual tab opens the Dual page, which displays the images side by side. A number of icons appear in the toolbar. The red icons are color coded to the reference image, which appears on the left and is surrounded by a red border. The green icons apply to the image to be registered, which is shown on the right side of the page and is surrounded by a green border.

Before being able to use either the Thin Plate Spline or Least Squares algorithm, you must first delineate a minimum of three corresponding points on both images. These points serve as guides, or landmarks, for aligning the images. In order for the points to be corresponding, you should delineate each point in the same sequence; i.e., the first point that you delineate on the reference image should be the first point delineated on the image to be registered, the second point on the reference image should be the second point delineated on the image to be registered, and so on.

To delete a point, you must first select the point and then use the appropriate Delete icon to delete it. However, before you can select the point, you need to return the cursor to its default mode using the  icon.

On both the Blended and Dual pages, you can modify the lookup table (LUT) by clicking , the LUT icon. When you select , the Lookup Table window appears. This window includes two tabbed pages: one for Image A, the reference image, and one for Image B, the image to be registered.

A common icon—, the Reset icon—appears on both the Blended and Dual pages. Use this icon if you want to restore the original version of the images. This icon deletes all points on the reference image and on the image to be registered.

---

**Note:** If the Apply button has already been selected, the Reset icon undoes all changes that occurred since the last time the Apply button was selected.

---

---

## IMAGE TYPES

You can apply this algorithm to 2D grayscale and color images.

The dimensions or image type of the image to be registered need not be the same as the dimensions or image type of the reference image. The registered image has the same image type as the image to be registered and the same extents as the reference image.

---

## SPECIAL NOTES

The following notes apply:

- For 2D images, three or more points are required. The algorithm may fail if nearly all of the points fall on the same line.
- For 3D images, four or more points are required and the algorithm may fail if the points nearly all fall on the same plane.

---

## REFERENCES

Refer to the following references for more information:

“Least-Squares Fitting of Two 3-D Point Sets” by K.S. Arun, T.S. Huang, and S.D. Blostein, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 5, September, 1987, pp. 698-700.

David Eberly, “Thin Plate Splines” of Magic Software at <http://www.magic-software.com>. Also at the same site are the files: `MgcInterp2DThinPlateSpline.h`, `MgcInterp2DThinPlateSpline.cpp`, `MgcInterp3DThinPlateSpline.h`, `MgcInterp3DThinPlateSpline.cpp` Paper that explains warping from one  $x,y$  source to another  $x,y$  target set.

Fred L. Bookstein, “Principal Warps: Thin-Plate Splines and the Decompositions of Deformations,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 6, June 1989, pp. 567–585.

## Applying the Manual 2D Series algorithm on 2D images

To run this algorithm, complete the following steps:

- 1 Open two images: the reference image and the image that you want to register.
- 2 Select the reference image.
- 3 Select Algorithms > Registration > Manual 2D Series. The Load dialog box appears.



Figure 2. The Load dialog box

- 4 Select the image that you want to register to the reference image. If only two images are open, the Set as Image B box automatically selects the other open image—the one you did not select—as Image B, or the image to be registered.
- 5 To select a different image, click Browse to find and open another image file.
- 6 Click OK. The Manual 2D Series window opens.

---

**Note:** Opening this window automatically corrects image resolution and scaling.

---

Note that this window includes two tabs:

- Blended, which displays the Blended page
- Dual, which displays the Dual page


The Blended page shows both images: the image to be registered superimposed on top of the reference image. Note that the icons near the top of the page allow you to move the image to be registered in all directions or specifically up, down, left, and right a certain number of pixels. You can also rotate the image to be registered clockwise or counterclockwise.


**7** Click Dual to go to the Dual page.



The Dual page shows both the reference image and the image to be registered side by side in the image area at the bottom of the page.



Note that the border around the reference image, the image on the left side of the page, appears in red. The border around the image to be registered, which is on the right side of the page, appears in green. Several of the icons near the top of the page are color coded to the images. That is, the red icons apply to the reference image, and the green icons apply to the image to be registered.



**8** Place at a minimum three landmarks, or points, on each of the images.

On the reference image, click , the Reference Slice Markers icon, and click the mouse on the reference image where the landmark should be located. The landmark appears on the image in red type and is sequentially numbered. Create at least two more landmarks on the image.





On the image to be registered, click , the Adjusted Slice Markers icon, and creates a landmark on the image by clicking the mouse. The landmark, which is sequentially numbered, appears in green type. Create a minimum of three landmarks on the image.

To delete a point on the reference image, you need to return the cursor to its default mode by clicking on . You can then select the point by clicking on it with the mouse and then delete it by clicking , the Delete selected reference slice markers icon.



To delete a point on the image to be registered, or adjusted image, click , select the point, and then delete it by clicking , the Delete adjusted slice markers icon.


**9** Select, if desired, either , the Least Squares algorithm icon, or , the Thin Plate Spline algorithm, to register the images. The program registers the images and the landmarks appear in yellow type followed by the specific positions in parentheses.

**10** At this point, you should determine the following:

- Whether there is a need to make manual adjustments to the registration. If so, return to the Blended page and use the direction or the rotation icons. Use  to verify alignment (refer to “Making manual adjustments to the image to be registered” on page 632 and to the following tables for more information).
- Whether to run the other algorithm. If so, simply click either , the Least Squares algorithm, or , Thin Plate Spline algorithm, as appropriate.
- Whether to restore the image to its original form by clicking .
- Whether to select Apply to keep changes made by the algorithm.

## Making manual adjustments to the image to be registered

After you apply either , the Least Squares algorithm icon, or , the Thin Plate Spline algorithm on the images, you may want to manually ascertain whether key portions of the reference image and the image to be registered, or adjusted image, are in alignment. One way of checking

alignment, or registration, is using , the Window region of image B icon. A recommended method for checking registration and for correcting it follows.



To check registration, do the following:


- 1** Click Blended in the Manual 2D Series window to display the Blended page. Both the reference image and the image to be registered, or adjusted image, appear at the bottom of the window. The adjusted image is superimposed over the reference image.
- 2** Note that the arrow on the Alphablending slider appears in the middle of the slider at 0.5R/A, which means that 50 percent of each image are shown on the page.
- 3** Slide the arrow on the Alphablending slider all the way to the right until it reaches the label “Adj. A.” This label indicates that 100 percent of the adjusted image is displayed on the page and, therefore, the reference image does not appear.




---

**Note:** The label “0.75A” means that 75 percent of the adjusted image appears and only 25 percent of the reference image is displayed.

---


- 4** Click , the Lookup Table icon, to display the Lookup Table window. Note that the window includes two tabs: one for Image A, the reference image, and one for Image B, the image to be registered.
- 5** Drag the image intensity line for the red channel up in the middle to adjust the color of the adjusted image. In this case, it creates a lighter colored red. The adjusted image now appears in pale red.






---

**Note:** Actually, you can change the color of the image to any color. The purpose in doing this task is to easily distinguish the adjusted image from the reference image.











---

- 6** Move the slider arrow all the way to the left on the slider. The label “Ref. R” indicates that the page now displays 100 percent of the reference and does not display the adjusted image.
- 7** Click  and position the cursor over the reference image. Notice that a square box appears on the image. The box is filled with a red image, which is the adjusted image. You can use this box to look at the alignment of details in the images.






**Tip:** You may want to maximize the window and enlarge the images so that you can view details of the images while checking alignment. To maximize the window, simply click , the Restore button. To maximize the image, click , the Magnify image 2.0x icon, which magnifies the image to twice its current size, as many times as needed.

**8** Determine whether you want to correct the alignment of the images. If you do, there are several ways to move the adjusted image:

- Moving the image using , the Translate image icon (refer to “Moving the adjusted image in any direction” on page 635)
- Moving the image using the direction arrows (refer to “Moving the adjusted image with the direction arrows” on page 636):
  - , the Move image up icon
  - , the Move image down icon
  - , the Move image right icon
  - , the Move image left icon
- Setting the number of pixels to move the image with , the Set pixel increment icon
- Rotating the image with the following icons (refer to “Rotating the adjusted image” on page 636):
  - , the Rotate image icon
  - , the Rotate clockwise icon
  - , the Rotate counterclockwise icon
  - , the Set degree increment icon

## MOVING THE ADJUSTED IMAGE IN ANY DIRECTION

The Translate image icon, or , allows you to move the adjusted image in any direction. This icon only appears on the Blended page, which is where all manual adjustments to image registration occurs.

To move the adjusted image, click , and, holding down the left mouse button, drag the icon in the direction in which to move the image, and then release the mouse button. After a few moments, the image appears in the new position. Note that, depending on where you moved the adjusted image, its outer portions may disappear beyond the outer edges of the reference image (Figure 3). In this case, you can use  to move the adjusted image again.

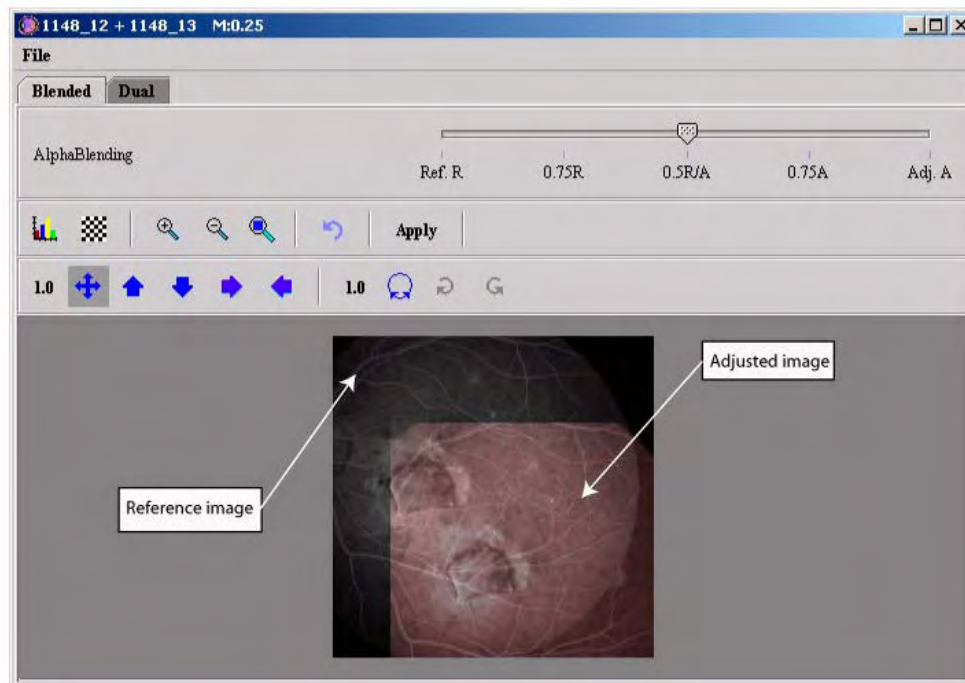









Figure 3. The Blended page of the Manual 2D Series window showing the adjusted image moved down and to the right




## MOVING THE ADJUSTED IMAGE WITH THE DIRECTION ARROWS

The icons that display direction arrows (, the Move image up icon; , the Move image down icon; , the Move image right icon; and , the Move image left icon) let you move the adjusted image in one direction only by a specified number of pixels. The default number of pixels is 1; however, the number of pixels may be from 0.01 to 2048.0. Therefore, the direction arrows provide the greatest specificity in moving the image, which is perfect for fine adjustments.

To change the number of pixels, click  <sup>1.0</sup>, the Set pixel increment icon; enter a number in the Pixel Increment box; and click Apply.

## ROTATING THE ADJUSTED IMAGE

If the rotation of the adjusted image is incorrect, you can change the rotation of the image with the rotation icons: , the Rotate image icon; , the Rotate clockwise icon; and , the Rotate counterclockwise icon.

To select the center of rotation for the image, click  and set the center by clicking with the mouse on the image. The letter *C* appears with a yellow crosshair cursor (Figure 4). You can then drag a line from the center of rotation to move the image above or below the center of rotation. To rotate the image clockwise, click ; and to rotate the image counterclockwise, click .

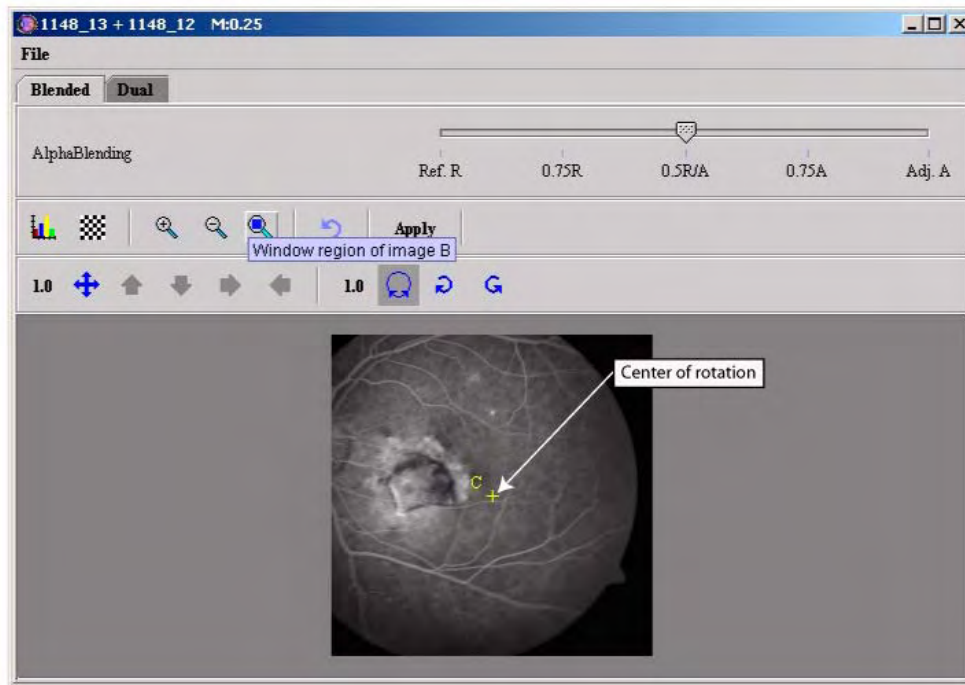
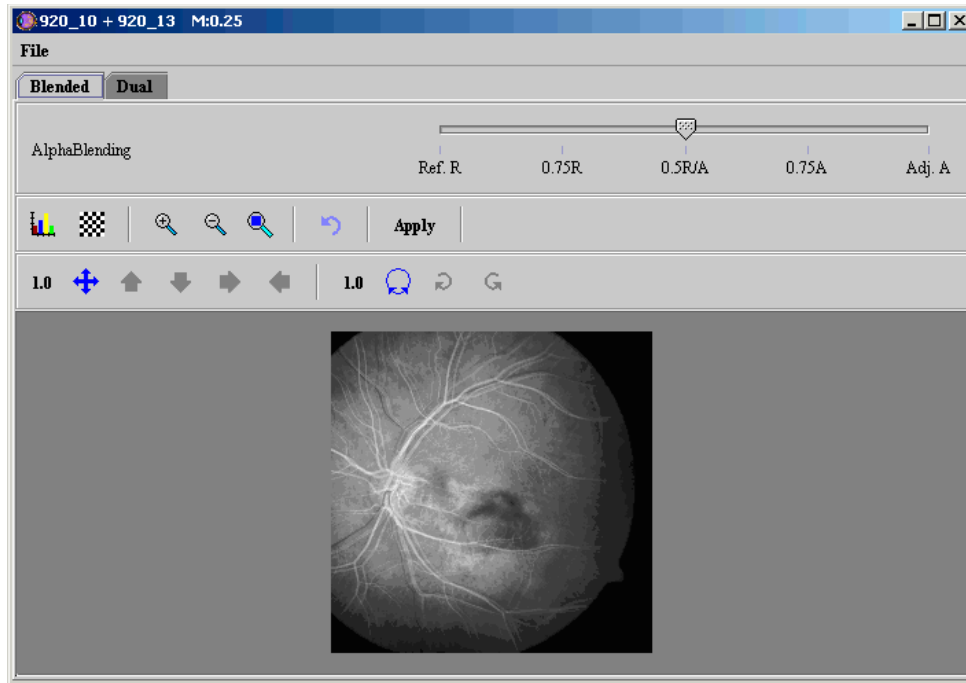


Figure 4. Manual 2D Series window showing center of rotation

As with the direction arrow icons, you change the degree increment to achieve exact registration. The default degree increment is 1.0, but you can specify from 0.01 to 360 degrees. To change the degree increment, click **1.0**, the Set degree increment icon. The Change Degree Increment dialog box appears. Enter the degree increment in the Degree Increment box, and click Apply.





<b>File</b>	<b>Close Registration</b>	Closes this window without saving the images.
	<b>Show VOIs in blended window</b>	Displays, on the Blended page, the VOIs that you delineated on the Dual page.
<b>Blended tab</b>	Displays the Blended page, which shows the image to be registered superimposed on the reference image. This page allows you to move and align the image to be registered with the reference image.	
<b>Dual tab</b>	Displays the Dual page, which shows each image side by side, and allows you to delineate three or more landmarks on each image before you apply the Landmark—Least Squares algorithm or the Landmark—Thin Plate Spline algorithm to the images.	
<b>Alphablending slider</b>	Adjusts the translucency of the alpha channels in each image using the alphablending technique. When two images share a window, you can adjust the alphablending settings so that you can see a blend of both images and can compare overlapping regions in two datasets.	
 <b>Lookup table</b>	Displays the Lookup Table window, which includes two tabs: Image A, the reference image, and Image B, the image to be registered.	
 <b>Checkerboard A &amp; B</b>	Displays a checkerboard pattern on the images. You can use this pattern for assistance in aligning the images.	

Figure 5. Manual 2D Series window showing the Blended page
















 <b>Magnify image 2.0x</b>	<p>Doubles the size of images, magnifying them to twice their current size. If an image is too large for the current window size, scroll bars appear, and you may need to manually adjust the size of the window.</p> <p>To restore an image to its original size, use , the Magnify image 0.5x icon, as many times as you used this icon.</p>
 <b>Magnify image 0.5x</b>	<p>Reduces the magnification level of images in half. To restore images to their original size, use , the Magnify image 2.0x icon, as many times as you used this icon.</p>
 <b>Window region of image B</b>	<p>Hides the reference image below a windowed mouse cursor; slide alphablending slider to the reference image</p>
 <b>Reset image to original state</b>	<p>Restores image to its original state. This icon deletes all points on the reference image and on the image to be registered. If the Apply button has already been selected, the Reset icon undoes all changes that occurred since the last time the Apply button was selected.</p>
 <b>Apply</b>	<p>Commits the changes that you made to the images, but it leaves the window open for you to make further changes if desired.</p>
 <b>1.0 Set pixel increment</b>	<p>Determines the number of pixels to use when any of the move image icons are used. When you click this icon, the Change Pixel Increment dialog box opens.</p> <div data-bbox="824 1039 1122 1176" data-label="Image"> </div> <p>Although the default is 1.0, when the pixel increment is changed to another number, that number displays on the toolbar rather than 1.0. For example, if you changed the pixel increment to 550.0,  would appear on the toolbar. You can use any pixel increment from 0.01 to 2048.00.</p>
 <b>Translate image</b>	<p>Moves Image B, the image to be registered, in any direction you drag the image by the number of pixels indicated in , the Set pixel increment icon.</p>
 <b>Move image up</b>	<p>Moves Image B, the image to be registered, up by the number of pixels indicated in , the Set pixel increment icon.</p>
 <b>Move image down</b>	<p>Moves Image B, the image to be registered, down by the number of pixels indicated in , the Set pixel increment icon.</p>

Figure 5. Manual 2D Series window showing the Blended page (continued)



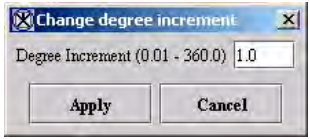





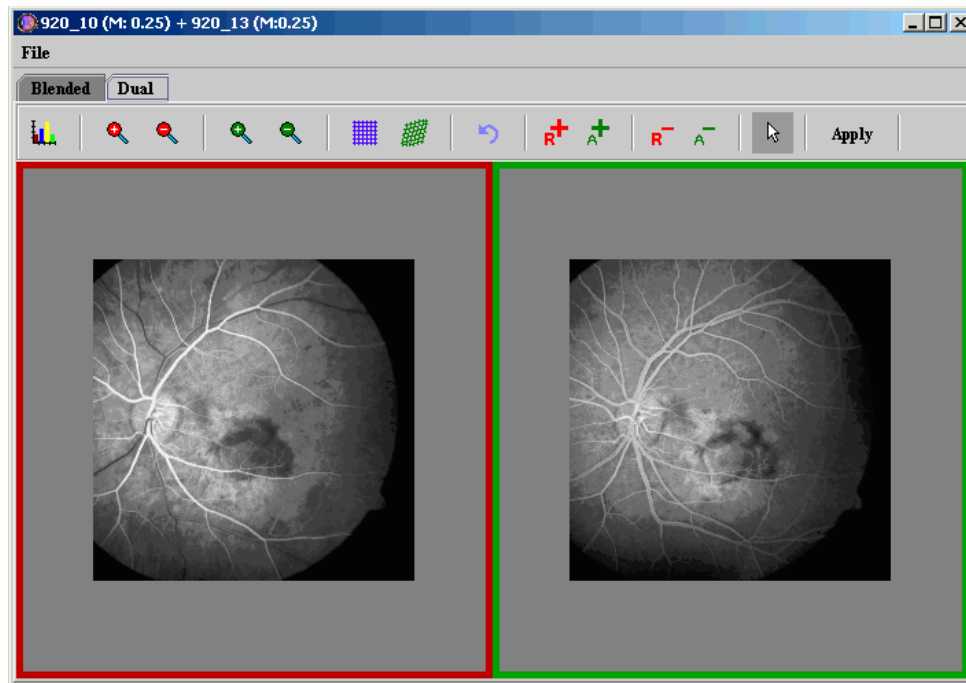
 <b>Move image right</b>	<p>Moves Image B, the image to be registered, right by the number of pixels indicated in <b>1.0</b>, the Set pixel increment icon.</p>
 <b>Move image left</b>	<p>Moves Image B, the image to be registered, left by the number of pixels indicated in <b>1.0</b>, the Set pixel increment icon.</p>
<p><b>1.0</b> <b>Set degree increment</b></p>	<p>Determines the degree to use when rotating images, rotating images clockwise, or rotating images counterclockwise. When you click this icon, the Change Degree Increment dialog box opens.</p> <div data-bbox="820 598 1128 735" style="text-align: center;">  </div> <p>Although the default degree increment is 1.0, when the degree increment is changed to another degree, that degree increment displays on the toolbar rather than 1.0. For example, if you changed the degree increment to 180.0, <b>180.0</b> would appear on the toolbar. You can set the degree increment from 0.01 to 360.0.</p>
 <b>Rotate image</b>	<p>Allows you to select a center of rotation by clicking on the image with the mouse. It then allows you to then move the image above or below the center of rotation by dragging a line from the center of rotation.</p>
 <b>Rotate clockwise</b>	<p>Rotates the image to be registered clockwise by the number of degrees indicated as the degree increment. To restore the image to its original rotation, click , the Reset image to original state icon.</p>
 <b>Rotate counterclockwise</b>	<p>Rotates the image to be registered counterclockwise by the number of degrees incited as the degree increment. To restore the image to its original rotation, click , the Reset image to original state icon.</p>
<p><b>Image area</b></p>	<p>Superimposes the image to be registered on the reference image and allows for the image to be registered to be moved, magnified, or rotated.</p>

Figure 5. Manual 2D Series window showing the Blended page (continued)






File	Close Registration	Closes this window without saving the images.
	Show VOIs in blended window	Displays, on the Blended page, the VOIs that you delineated on the Dual page.
Blended tab	Displays the Blended page, which shows the image to be registered superimposed on the reference image. This page allows you to move and align the image to be registered with the reference image.	
Dual tab	Displays the Dual page, which shows each image side by side, and allows you to delineate three or more landmarks on each image before you apply the Landmark—Least Squares algorithm or the Landmark—Thin Plate Spline algorithm to the images.	
Lookup Table	Displays the Lookup Table window, which includes two tabs: Image A, the reference image, and Image B, the image to be registered.	
Magnify reference image 2.0x	<p>Doubles the size of reference images, magnifying them to twice their current size. If an image is too large for the current window size, scroll bars appear, and you may need to manually adjust the size of the window.</p> <p>To restore reference images to their original size, use , the Magnify reference image 0.5x icon, as many times as you used this icon.</p>	

Figure 6. Manual 2D Series window showing the Dual page



















 <b>Magnify reference image 0.5x</b>	<p>Reduces the magnification level of reference images in half. To restore reference images to their original size, use , the Magnify reference image 2.0x icon, as many times as you used this icon.</p>
 <b>Magnify adjusted image 2.0x</b>	<p>Doubles the size of adjusted images, magnifying them to twice their current size. If an image is too large for the current window size, scroll bars appear, and you may need to manually adjust the size of the window.</p> <p>To restore adjusted images to their original size, use , the Magnify reference image 0.5x icon, as many times as you used this icon.</p>
 <b>Magnify adjusted image 0.5x</b>	<p>Reduces the magnification level of adjusted images in half. To restore adjusted images to their original size, use , the Magnify reference image 2.0x icon, as many times as you used this icon.</p>
 <b>Apply least squares alignment</b>	<p>Applies the Landmark—Least Squares algorithm to the image to be registered. After the algorithm is applied, if needed, you can apply the Landmark—Thin Plate Splines algorithm on the image, or you can manually adjust the alignment of the image on the Blended page.</p>
 <b>Apply thin plate spline alignment</b>	<p>Applies the Landmark—Thin Plate Spline algorithm to the image to be registered. After this algorithm is applied, if needed, you can apply the Landmark—Least Squares algorithm on the image, or you can manually adjust the alignment of the image on the Blended page.</p>
 <b>Reset image to original state</b>	<p>Restores the image to be registered to its original state. Restores image to its original state. This icon deletes all points on the reference image and on the image to be registered. If the Apply button has already been selected, the Reset icon undoes all changes that occurred since the last time the Apply button was selected.</p>
 <b>Reference slice markers</b>	<p>Adds a point to the reference image each time you click the left mouse button. Note that the points are sequentially labeled from 1 on in the order in which you created them. To remove points, return the cursor to its default mode by clicking , select the point to be deleted, and click , Delete selected reference slice markers.</p>
 <b>Adjusted slice markers</b>	<p>Adds a point to the adjusted image each time you click the left mouse button. Note that the points are sequentially labeled from 1 on in the order in which you created them. To remove points, return the cursor to its default mode by clicking , select the point to be deleted, and click , Delete selected adjusted slice markers.</p>
 <b>Delete selected reference slice markers</b>	<p>Removes the point that you selected in the reference image. To remove a point, first return the cursor to its default mode by clicking , select the point to be deleted, and click , Delete selected reference slice markers.</p>

Figure 6. Manual 2D Series window showing the Dual page (continued)






 <b>Delete selected adjusted slice markers</b>	<p>Removes the point that you selected in the image to be registered. To remove a point, first return the cursor to its default mode by clicking , select the point to be deleted, and click , Delete selected adjusted slice markers.</p>
 <b>Return to default mode</b>	<p>Returns the cursor to its default mode, which allows you to change from adding points to deleting points from the image.</p>
 <b>Apply</b>	<p>Commits the changes that you made to the images, but it leaves the window open for you to make further changes if desired.</p>

Figure 6. Manual 2D Series window showing the Dual page (continued)

## Applying the Manual 2D Series algorithm on 2.5 images

To run this algorithm on 2.5D images, complete the following steps:

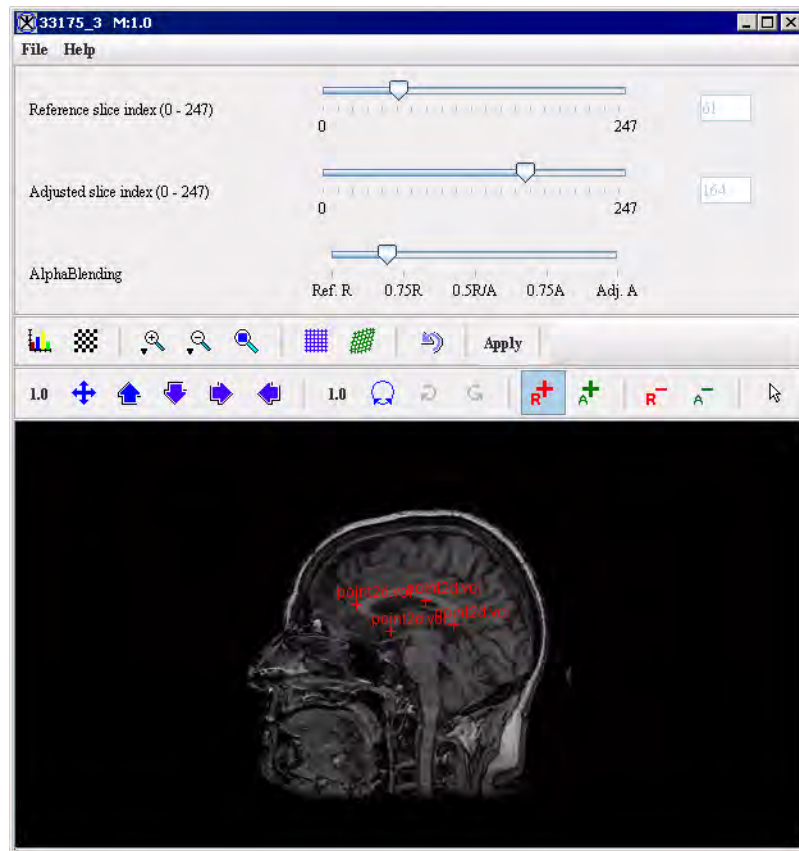
- 1** Open a single 2.5 image dataset. When you apply this algorithm, you select the slices in the dataset that you want to register.
- 2** Perform, as an option, any other image processing, except scaling, on the image dataset.
- 3** Select Algorithms > Registration > Manual 2D Series. The Manual 2D Series window for 2.5D images (Figure 7) opens.

Because you are registering slices of an image, and not separate images, all of the necessary icons and controls appear in a simple window that does not include tabbed pages. Note that there are two sliders at the top of the page:

- *Reference slice slider*—Allows you to choose which slice to use as a reference image.
- *Adjusted slice slider*—Allows you to choose which slice to use as an image to be registered.

The boxes to the right of each slider indicate which slice you are currently viewing.

All of the remaining icons and controls work in the same way that they do on the Manual 2D Series window for 2D images (refer to Figure 5 and Figure 6).



**Figure 7. Manual 2D Series window**

*For 2.5D images, this window does not include the Blended and Dual tabbed pages. Rather it incorporates some of the same interface controls that appear on the Blended and Dual pages, such as icons and the alphablending slider. Note that you can place landmarks on the image in this window.*

**Refer to the instructions for “Applying the Manual 2D Series algorithm on 2D images” on page 630.**

---

## Mosaic Registration

*The Mosaic Registration method is a user interface for a semi-manual image registration. It allows the user, first, manually aligns two images using a mouse, and then it calls the Optimized Automatic Registration 2D algorithm (AlgorithmRegOAR2D) for further precise registration. As a result the method creates a new mosaic image from the two source images. Multiple images can also be added to the mosaic and aligned one at a time.*

### Background

The Mosaic Registration technique can be used for increasing a field of view (FOV) by combining two or more images in a single FOV. Refer to Section “Outline of the method” .

---

### OUTLINE OF THE METHOD

#### Pre-registration steps

- 1** The user opens a reference image, and then adds a tile image.
- 2** Both images are displayed as texture-mapped polygons. The texture sizes match the original image size. If an image is larger than the screen size, the polygon is scaled to fit the image to the screen size.
- 3** The centers for both images are computed, and then a single translation step is performed to align the image centers.

#### Manual mouse based alignment

- 4** The user manually aligns both images using a mouse. Manual translation is done in a screen space – the center of the selected image is at the mouse position. Rotation is about the center of the selected image, the rotation angle is calculated based on the mouse movement. Scale is also based on mouse movement and is relative to the selected image.
- 5** The minimum resolution of the reference and tile images is determined. For more information, refer to “Blurring” on page 652.

- 6** The method registers the tile image to the reference image based on the how the user arrange the images using the mouse. It creates two new image frames, and then it writes the reference and tile mask images to these frames so that the reference and tile image overlap. Refer to “Registering the reference and tile images” on page 648.
- 7** The method saves the reference and tile polygon shapes, borders, images, and transformation matrices.

### Optimized Automatic Registration 2D

- 8** The method calls the Optimized Automatic Registration 2D algorithm to register images created in step 6. See also “AlgorithmOAR2D Optimization Steps” on page 654.

### Forming the mosaic image

- 9** After AlgorithmRegOAR2D finishes running, the method blends the tile image with the reference image, and then displays the result mosaic image. See also “Blending the tile image with the reference image” on page 649.

---

## OPENING TWO IMAGES

When the user opens two images, the algorithm asks to open a reference image first, and then to add the tile image. In that step, the image centers are computed for each image, and then a single translation step is performed to align the image centers.

---

## MANUAL MOUSE-BASED ALIGNMENT

The mouse-based alignment step involves manual definition of identical points or key features between the reference and tile image (which means that the user should figure out the identical points), manual aligning both images using those points, and then saving the transformation vector.

The manual mouse based alignment is a traditional affine transformation that involves rotation, translation and scale. After the manual transformation is done, the image content of the tile image is transformed to the geom-

etry of the reference image and the transformation parameters are stored in the transformation vector.

## Rotation

The method allows the user to rotate the tile image using the left mouse button. The rotation is performed around the center of the image until the identical points of both images were aligned. The rotation angle is then calculated based on the mouse's X- coordinate on the screen.

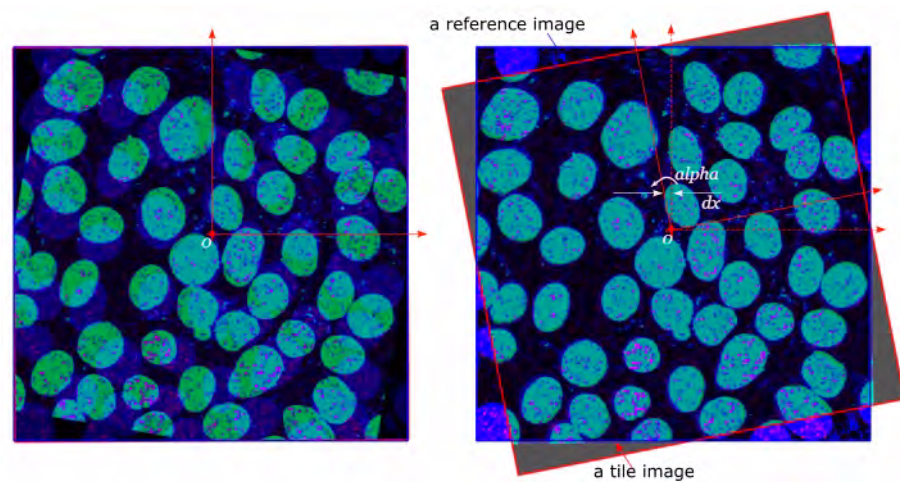


Figure 1. Calculating the rotation angle

## Translation

Manual translation is done using the right mouse button. The right mouse button, when pressed, points the mouse cursor to the center of the image. When the user moves the mouse, by default, the tile image moves along the reference image in the X and Y directions until the identical points in both images were aligned. The translation is then calculated based on the mouse's X and Y coordinates before and after translation. See Figure 2. To move the reference image, make it an active image first, and then use the same technique as was used for moving the tile image.

## Scale

Manual scale can be done using the middle mouse button. The tile (or reference) image is scaled until the identical points in both images were aligned. The scale value is then calculated based on the mouse movement in Y direction on the screen. For example, each time the mouse moves up in Y direc-

tion, the tile image is scaled larger, and each time the mouse moves down in Y direction, the tile image is scaled smaller.

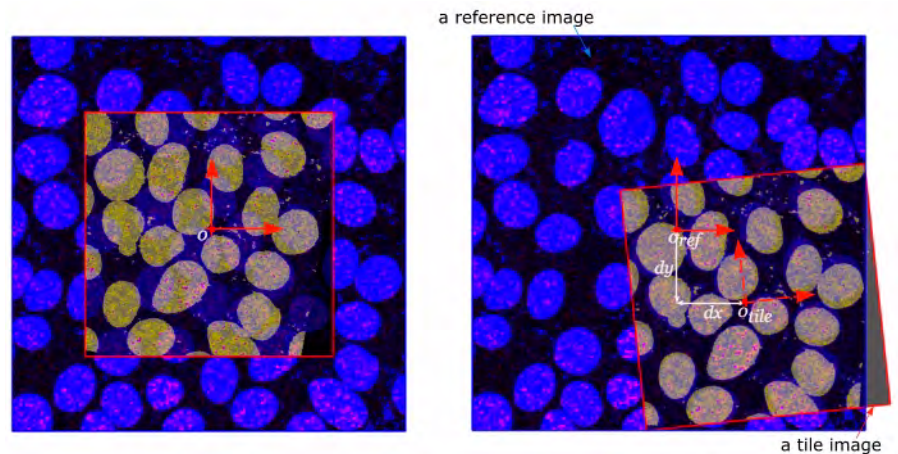


Figure 2. The translation is calculated based on the mouse's X and Y coordinates before and after translation.

### Storing the transformation parameters

After each manual alignment step (e.g. each rotation, translation and scale), the method stores the current transformation for the tile image, so the new transformations can be concatenated onto the current transform. In other words, the method links the current transformations with the previous accumulated transformation, and then saves the transformation matrix.

## REGISTERING THE REFERENCE AND TILE IMAGES

When the user press the Register Images button, the following happens:

- 1 The Inverse Transform for the Reference image is calculated and applied to the reference image.
- 2 The Tile Transform is concatenated with the Inverse Reference Transform so that the tile image is transformed relative to the reference Image.



- 3** The combined mosaic image size is calculated, based on the original reference image size and the transformed extents of the tile image. Now, the new image size is large enough to contain both images.
- 4** The reference image is resized to match the mosaic image.
- 5** The tile image is transformed and placed into a new tile image that has the same size as the reference image.
- 6** Two mask images are created, for the updated reference and tile images. These mask images have a value of one only where the two images overlap and a value of zero otherwise.
- 7** To further refine the registration, the `AlgorithmRegOAR2D` is then called for the reference, tile, and both mask images for a weighted registration, where the images are registered based on the image information only in the areas of the mask images that have the value equal to one. Therefore, the registration is based only on the areas where the two manually-aligned images overlap.
- 8** The Registration Transform is returned by the `AlgorithmRegOAR2D` class. The Registration Transform represents the alignment with the lowest negative cost function value.
- 9** The tile image is then transformed and blended to the reference image.

See also “Using the Optimized Automatic Registration 2D algorithm” on page 650.

---

## BACKUP MOSAIC

Before `AlgorithmRegOAR2D` is called, the method stores the reference and tile polygon shapes, borders, images, and transformation matrices, so that the manual alignment can be undone at any time by the user.

---

## BLENDING THE TILE IMAGE WITH THE REFERENCE IMAGE

Once the two images are manually aligned and the registration is complete by `AlgorithmRegOAR2D`, the images are blended and the final image is then displayed.

---

## BRUTE FORCE

The Brute Force algorithm is a method of registering images, which includes: a) capturing a reference image, b) capturing a tile image, c) coarsely registering the reference image to the tile image to give approximate angle and offset parameters, and then d) finely registering the reference image to the tile image based on the parameters stored from the coarsely matching step.

The coarsely registering step involves, first, resampling both images to a coarse resolution, and then applying a number of rotations and translations to register the tile image to reference image.

The finely registering step employs a control point matching between the reference image and the tile image using small image patches at a scale of one for the reference image and the tile image having lower resolution.

---

No optimization is performed in Brute Force registration.

---

The Brute Force algorithm starts when the user activates the Brute Force Registration option. See also “The Mosaic Registration dialog box options” on page 660 and “Brute Force Settings” on page 663.

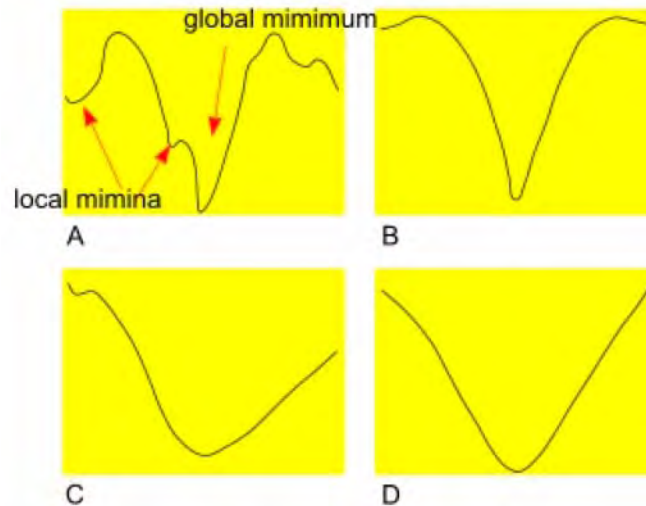
## Using the Optimized Automatic Registration 2D algorithm

---

### BACKGROUND

The Optimized Automatic Registration 2D algorithm searches for an optimal transformation that registers two images. The main approach for determining this optimal transformation is a) to calculate a cost function, b) determine how it changes as individual transformation parameters are varied, and c) find the parameters that minimize the value of the cost function. For example, Figure 3 shows a plot of a sample cost function for a selection of transformation parameters; in each plot, a single parameter is varied while all other are kept constant.

See also “Cost functions used in MIPAV algorithms” on page 63.



**Figure 3.** The plots of the Correlation Ratio cost function versus some of the individual parameter values. In each plot (A, B, C, and D), a single parameter is varied while the other are kept constant

Another fundamental step in this registration method is to resample and interpolate the reference and target images at several different resolutions (starting with a coarse pixel size and gradually increasing the resolution), while searching for the min cost function. An advantage of this multi-resolution approach is that the initial optimization considerably reduces computational cost, since the number of sample points is substantially less. In addition, the overall alignment is easier to find at a coarse resolution.

In that algorithm we use the inverse cost function, so the min cost corresponds to the better registration.

However, the resampling process may influence the computed value of the cost function; therefore, several different interpolation methods are incorporated with this technique. These methods are discussed in more detail later in Section “Resampling Interpolation” on page 657.

Taking into account the resampling and interpolation criteria, AlgorithmRegOAR2D begins searching for an optimum transformation at a coarse resolution of 8mm. Once a transformation has been determined that

minimizes the cost function, the transformation is further refined by progressively increasing the resolution, e.g. 8, 4, 2, 1.

The algorithm offers various resampling interpolation methods combined with the Correlation Ratio, Least Squares, Normalized Cross Correlation, and Normalized Mutual Information as a cost function. See also Section 1.2.5, “Cost functions.”

---

## OUTLINE OF THE METHOD

### Pre-Optimization

The algorithms input are two images after performing the manual alignment. It takes as an input both images as they appear from the Manual mouse-based alignment step, (these two new registered images are now called a reference and a target image), and then performs the pre-optimization which includes blurring and resampling.

### Blurring

The registration algorithm first, determines the minimum resolution of the reference and target images. The reference image is blurred to the resolution of the target image if the resolution of the target image is 50% or larger than the corresponding resolution in the reference image. Likewise, the target image is blurred to the resolution of the reference image if the resolution of the reference image is 50% or larger than the corresponding resolution in the target image. It is possible (but not likely) that both the reference and target image will be blurred. Both images are then resampled into 1mm isotropic pixels.

### Calculating the center of mass

The algorithm calculates centers of mass (COM) for each image, and then a single translation step is performed to align the COMs. To calculate the COM, the method uses the right hand convention 2D coordinate systems (X, Y). Thus, in the image space, the left hand corner of the image is set to (0, 0). The X axis goes left to right and the Y axis goes top to bottom. The rotation about the Y axis can be determined by the angle between the normal to the planes for the two images, while translational compensate along the X axis is given by the difference in COMs in two images.

## Resampling

Once this isotropic 1mm resolution images have been obtained, the 2, 4 and 8mm subsampled versions are also created. The sub-sampling algorithm then simply keeps every n-th point (that is, 2, 4 or 8) on the lattice in each direction. Therefore, the new volume contains  $1/n^2$  as many points as the original. See also “Resampling Interpolation” on page 657.

## Optimization

The optimization function is used to determine the optimum transformation. It takes as an input both images (blurred and resampled), and then progresses them through a series of different levels corresponding to both the reference and target images resampled and interpolated to higher levels of resolutions.

- 1 levelEight optimization.** Both images are resampled and interpolated to a resolution of 8mm. The parameters of the transformation are then systematically varied, where the cost function is evaluated for each setting. The parameters corresponding to the smallest value(s) of the cost function are then used as the initial transformation for the next level in the optimization.
- 2 levelFour optimization.** The images are resampled and interpolated to 4mm and the transformation corresponding to the smallest value(s) of the cost function are determined and used as the initial transformation for the next level in the optimization.
- 3 levelTwo optimization.** Similar processing as the previous two levels except the images are, first, resampled and interpolated to 2mm.
- 4 levelOne optimization** – 1mm resolution images are used in this step and the transformation is generalized to include 12 degrees of freedom.

## Post optimization

Using the same interpolation methods as described in “Pre-Optimization” and the optimal transformation determined above, the method transforms the reference image into the same coordinate system of the target image.

Following sections describe in detail the algorithms and methods used to perform Steps 1–8 of the optimization routine.

## ALGORITHM OAR2D OPTIMIZATION STEPS

**levelEight optimization** is primarily concerned with determining the rotational transformation that will lead to the globally optimal transformation, which corresponds to the *global minimum* of the cost function. levelEight optimization uses the lowest resolution images (8mm) and coarse rotation angles evaluated at relatively large step sizes.

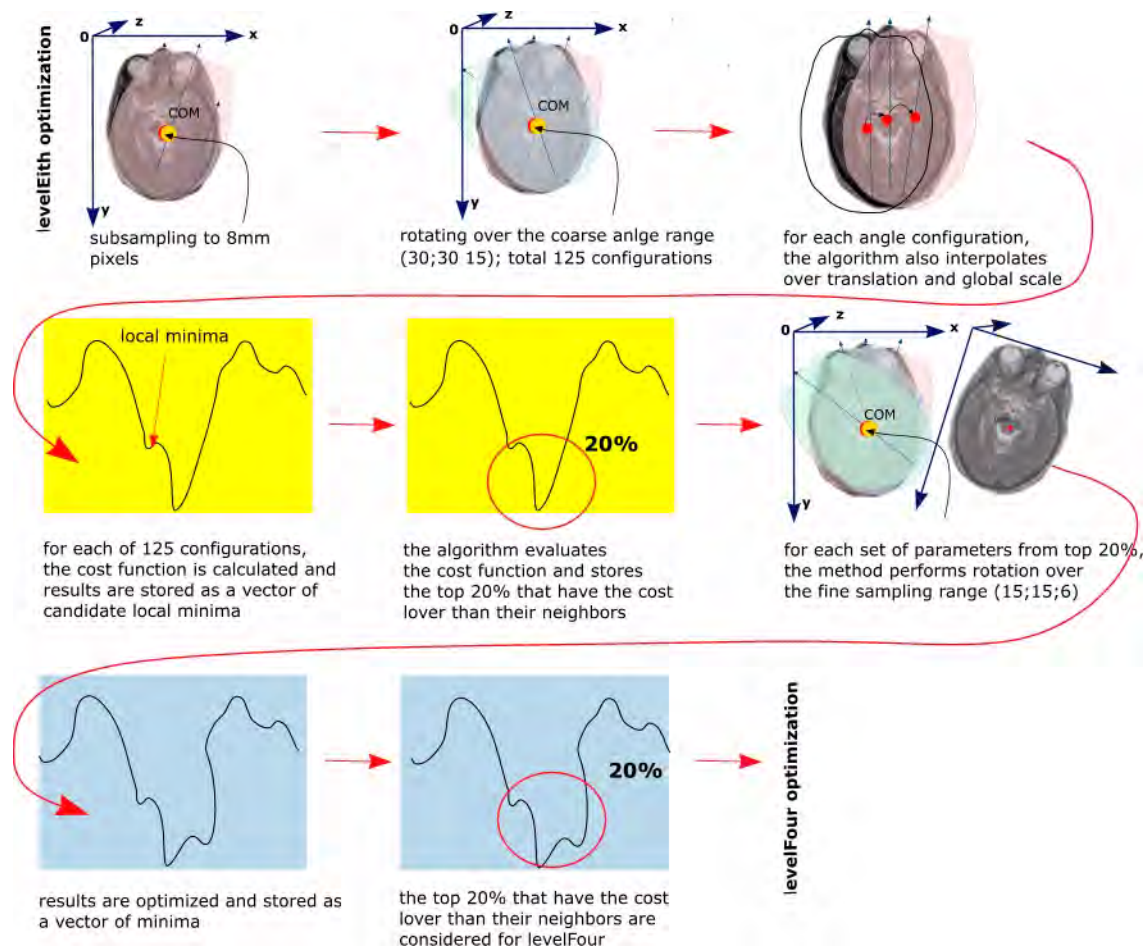


Figure 4. AlgorithmOAR2D the levelEight optimization steps

### Rotation

This AlgorithmRegOAR2D method is based on the empirical observation that finding the correct orientation, or rotation, is the most difficult task in image registration, since the three rotation parameters are highly coupled

and most of the erroneous registrations that have been examined have happened primarily due to an incorrect orientation. Therefore, the search concentrates on the rotational part of the transformation.

Specifically, the reference image is oriented, interpolated, and the cost function evaluated for coarse rotations angles  $(-5, 0)$ ,  $(0, 5)$ , ...,  $(-5, 5)$ , where the values represent the amount of rotation in degrees about the X, and Y axes respectively.

For each angle configuration, a four degrees of freedom (4-DOF) local optimization is also performed to find the optimal translation and (global) scale. By default, the initial values are set to

- zero for translation
- one for scale.

The best 20% of the cost values and the corresponding angle configurations (candidate local minima) are stored in a vector of minima that is used as a starting point for a further optimization, which uses a smaller step size over a narrower range of angles.

For each parameter setting corresponding to the top 20% of the cost function minima, the algorithm performs rotations over the fine grid (which is, by default, 3 degrees with 1 degrees step) and evaluates the cost function.

The method now finds all angle configurations that have the corresponding cost function values lower than their neighbors from the vector of candidate local minima.

For each of these sets of parameters a 7-DOF optimization is then performed, storing the results of the transformation and costs before and after optimization in a vector of minima.

Since the relative costs of each candidate solution may change at higher resolutions, where structures become more sharply defined, the top 20% of the cost function minima points are considered for the next higher resolution (4mm) stage, rather than just the single best solution.

By default, the algorithm uses Bilinear interpolation to transform images to this new orientation. And Least Squares is the cost function implemented in this method.

**levelFour optimization step** uses the interpolated 4mm resolution images to determine the transformation that minimizes the cost function starting with the transformations determined in levelEight. The optimization determines a 7-DOF transformation that corresponds to the minimum value of the cost function. This transformation is then perturbed and the cost function is computed for these new settings. The perturbations correspond to six-degrees for each rotation parameter and a global scaling factor of 0.8, 0.9, 1.1, and 1.2. A type of transformation (translation and/or global scale) is specified by a user (see Section 1.5, Running the Mosaic Registration). It includes the following transformations: Affine-12 (the default), Rigid-6, Global Rescale-7, and Specific Rescale-9. A vector of parameters and top 20% of the min cost function values are considered for the next 2mm step.

**In levelTwo**, the method uses the images interpolated to 2mm. The cost function is evaluated using the transformation parameters corresponding to the top 20% minima obtained from the levelFour. It finds the best minimum, and then optimizes it with the 7-DOF and then 9-DOF transformation. The method then returns the best minimum after optimization.

---

Note: if the user has limited the degrees of freedom to six, as for the rigid transformation, there will only be one optimization run in levelTwo, with six degrees of freedom.

---

**levelOne** step uses the 1mm interpolated images and computes the value of the cost function for each parameter setting determined in the previous levelTwo optimization. The parameters corresponding to the minimum value of the cost function with this resolution are then further optimized for transformations with 7-, 9-, and 12- DOF, unless a rigid (6-DOF) transformation is desired.

### What is saved

The method stores:

- The set of parameters or vector at which the minimum was reached;
- The value of the cost function at that minimum;
- And the matrix, which was the true input into the cost function and represents the transformation that gives the minimum cost of differences between the images.



---

## RESAMPLING INTERPOLATION

AlgorithmRegOAR2D resamples images using an interpolation scheme, where image pixels are resampled into isotropic 1mm pixels. New pixel values are computed using a weighted combination of existing pixel values within a defined neighborhood of the new pixel location. The interpolation methods available in this implementation of the registration technique include Bilinear, B-spline 3-rd order, B-spline 4-rd order, Cubic Lagrangian, Quintic Lagrangian, Heptic Lagrangian, and Windowed sinc.

The major effect that the choice of interpolation could have is in determining the degree when the cost function becomes continuous or discontinuous with respect to changes made in the transformation parameters. Thus, selecting the appropriate interpolation method is important for future calculation of the cost function.

To learn more about the interpolation methods used in MIPAV, refer to “Interpolation methods used in MIPAV” .

---

## COST FUNCTIONS

The Optimized Automatic Registration 2D method searches for the transformation that gives the smallest value of the *inverse cost function*, which we assume is the transformation that also gives the best alignment.

---

Least Squares is the cost function implemented in this method. Refer to “Cost functions used in MIPAV algorithms” for more information.

---

### Powell method

Optimized Automatic Registration 2D uses the Powell method to find the global minimum of the chosen cost function. For more information about the Powell method, refer to < <http://math.fullerton.edu/mathews/n2003/PowellMethodMod.html>>

---

## OUTPUT

The transformed image is displayed. In order to receive a better resolution, the same interpolation methods which were described in Section “Resampling Interpolation” on page 657 can be applied to the transformed image to produce the final output image.

---

## IMAGE TYPES

You can apply this algorithm to all 2D color (RGB) and grayscale images.

## Running the Mosaic Registration method

To run Mosaic Registration (Figure 8),

- 1** Call Algorithms>Registration> Mosaic Registration.
- 2** In the Mosaic Registration dialog box, click the Open Reference Image button and then choose the reference image. The reference image appears in the dialog box. See Figure 5-2.
- 3** Press Add Tile Image and choose the tile image. See Figure 5-3.
- 4** Use the mouse to manually register two images. See Figure 5-4.
- 5** After the manual alignment is done, press Register Images to run the Optimized Automatic Registration 2D algorithm. Figure 5-5.

When the Optimized Automatic Registration 2D algorithm finishes running the result image appears in the dialog box window.

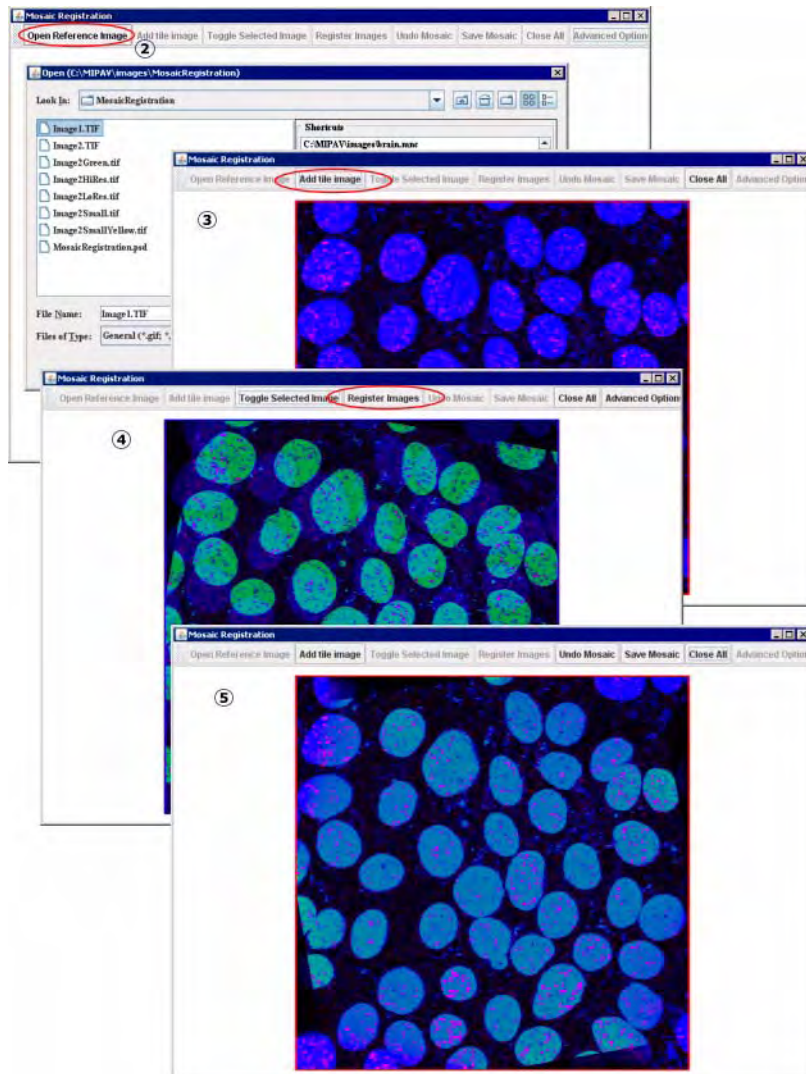


Figure 5. Running Mosaic Registration

## THE MOSAIC REGISTRATION DIALOG BOX OPTIONS

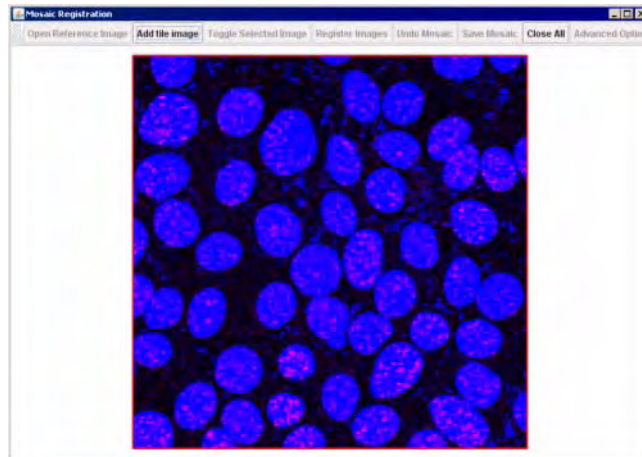


Figure 6. Mosaic Registration dialog box

**Open Reference Image**—browse your image directory to select the reference image.

**Open Tile Image** – browse your image directory to select the reference image.

**Toggle Selected Image**—highlights a selected image. By default, the reference image appears in the blue frame and the tile image is the active image and appears in the red frame. You can manually rotate and translate the toggled image, no matter if it is a reference or tile image.

**Register Images**—registers images using the Optimized Automatic Registration algorithm.

**Undo Mosaic**—undoes the registration.

**Save Mosaic**—saves the result image.

**Close all**—closes all images.

**Advanced Options**—opens the Optimized Automatic Registration 2D dialog box. See “Optimized Automatic Registration 2D dialog box options” .

## OPTIMIZED AUTOMATIC REGISTRATION 2D DIALOG BOX OPTIONS

Input options	
<b>Degrees of freedom</b>	<p>Specifies the spatial model used to restrict the type of linear transformation being applied during registration. Select one of the following: Affine-6 (the default), Rigid-6, Global rescale-7, Specific rescale-9, and Affine-12.</p> <p>See also “Cost functions used in MIPAV algorithms” , “Degrees of freedom” on page 71.</p>
<b>Interpolation</b>	<p>Specifies the type of interpolation the algorithm will use when resampling. The list includes: Bilinear, B-spline 3-rd order, B-spline 4-th order, Cubic Lagrangian, Quintic Lagrangian, Heptic Lagrangian, Windowed sinc. See also “Interpolation methods used in MIPAV” .</p>
<b>Process color image</b>	<p>Check this box to process a color image.</p>
<b>Cost function</b>	<p>Specify a cost function which will be used to evaluate registration. The list includes: Least squares, Correlation ratio, Normalized cross correlation, Normalized mutual information</p>
<b>Rotation angle sampling range</b>	<p>specify the rotation angle. By default the angle is set from -5.0 to 5.0 degrees.</p>
<b>Coarse angle increment</b> <b>Fine angle increment</b>	<p>options can be used for two pass registration. The first registration pass would apply a coarse-level rotation to quickly arrive at an approximate registration, and then the second pass would apply a finer-level rotation to continue from the point where the first pass registration has finished. By default, the coarse angle increment is set to 2.0 degrees and fine angle increment is set to 1.0 degrees.</p>

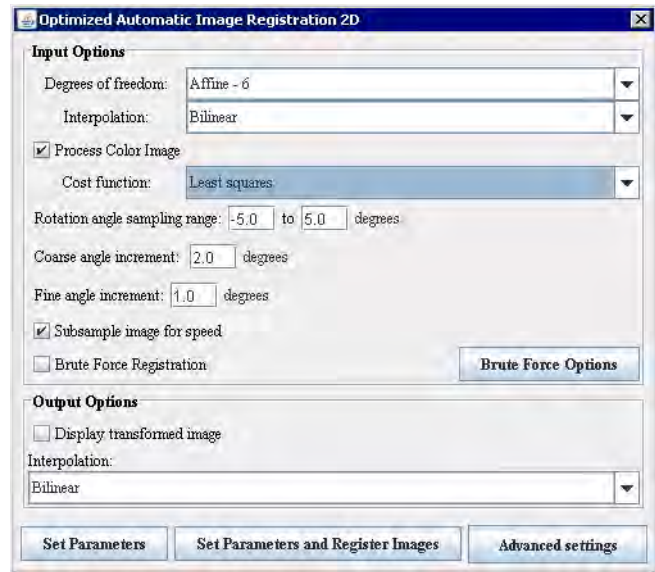


Figure 7. The Optimized Automatic Registration 2D dialog box options

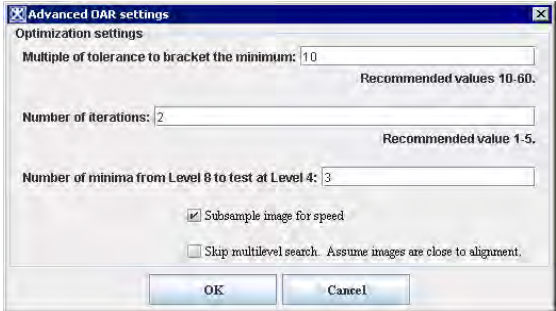
<b>Subsample image for speed</b>	–if checked, this option allows subsampling the image to speedup the registration.	
<b>Brute Force Registration</b>	–if checked, this option calls the Brute Force algorithm. See also “Brute Force” on page 650.	
<b>Output Options:</b>		
<b>Display transformed image</b>	allows to view the transformed image in a separate window. By default, this option is active.	
<b>Interpolation:</b>	select the calculation method, to produce the output image from the array of transformation matrices and input image. You can choose among Bilinear, B-spline 3-rd order, B-spline 4-th order, Cubic Lagrangian, Quintic Lagrangian, Heptic Lagrangian, or Windowed sinc. The default choice is Bilinear.	
<b>Set Parameters</b>	–saves current parameters.	
<b>Set Parameters and Register Images</b>	–saves current parameters and register images.	
<b>Advanced Settings</b>	–opens the Advanced OAR Settings dialog box. Pressing the Advanced button opens the Advanced OAR Settings dialog box where you can specify the additional parameters which will be used to calculate the min cost function.	
<b>Advanced OAR settings</b>		
<b>Multiple of tolerance to bracket the minimum</b>	Recommended values are 10–60.	
<b>Number of iterations</b>	Recommended are 1–5 iterations.	
<b>Number of minima from Level 8 to test at Level 4</b>	By default this is the best 20%, but you can enter your own number here.	
<b>Subsample image for speed</b>	subsamples the image.	
<b>Skip multilevel search. Assume images are close to alignment.</b>		
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.	
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes it.	
<b>Help</b>	Displays online help for this dialog box.	

Figure 7. The Optimized Automatic Registration 2D dialog box options (continued)

## BRUTE FORCE SETTINGS

The Brute Force registration is used, when the user activates the Brute Force Registration option in the Optimized Automatic Registration dialog box, see Figure 9.

**Rotation Range** – enter the rotation angle, use zero for no rotation optimization.

**Translation Range** – enter the translation range in pixels, use zero for no translation.

**Scale in X** – enter the scale value in percentile (%) which will apply to the image to scale it in the X axis direction. A zero value corresponds to no scale in X axis.

**Scale in Y** – enter the scale value in percentile (%) which will apply to the image to scale it in the Y axis direction. A zero value corresponds to no scale in Y axis.

**Number of steps for testing scales** – enter the number of steps which should performed to test the scale values (in X and Y directions). A zero value corresponds to no scale optimization. See Figure 8.

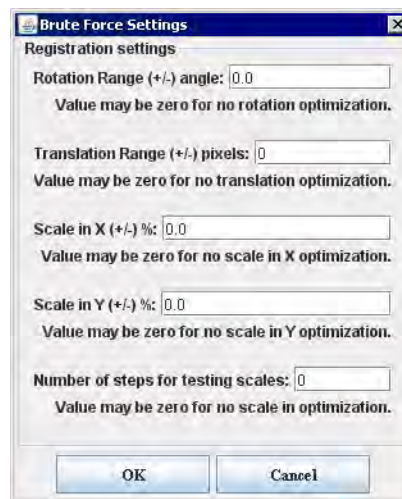


Figure 8. The Brute Force algorithm settings

---

## Registration: Display Pixel Similarity Cost Functions

*The algorithm calculates costs for various voxel similarity cost functions that are used in registration and output them to the Output window.*

### Background

For two chosen images, the algorithm takes pixel (voxel) similarity cost functions implemented in MIPAV, and then, for each cost function, it computes the cost value. The cost values are then displayed in the Data tab in the Output MIPAV window.

---

Note that MIPAV uses inverse cost functions, therefore the min cost function corresponds to the better registration.

---

The following voxel similarity cost functions are utilized in this method:

- “Correlation ratio” on page 64 refer to “Cost functions used in MIPAV algorithms”
- “Normalized cross correlation” on page 67, refer to “Cost functions used in MIPAV algorithms”
- “Normalized mutual information” on page 70, refer to “Cost functions used in MIPAV algorithms”

To apply the method to images, make sure that both images are of the same type and same dimensionality. Also, if while running the method, the second image appears to have a resolution (or a number of bins) that differs from the resolution of the first image, the method will rescale the second image.

---

### IMAGE TYPES

You can apply this algorithm to color, grayscale, and black and white 2D and 3D images. Note that, in order to run the algorithm, both images must



be of the same type (e.g. both color, or both grayscale, or both black and white) and same dimensionality.

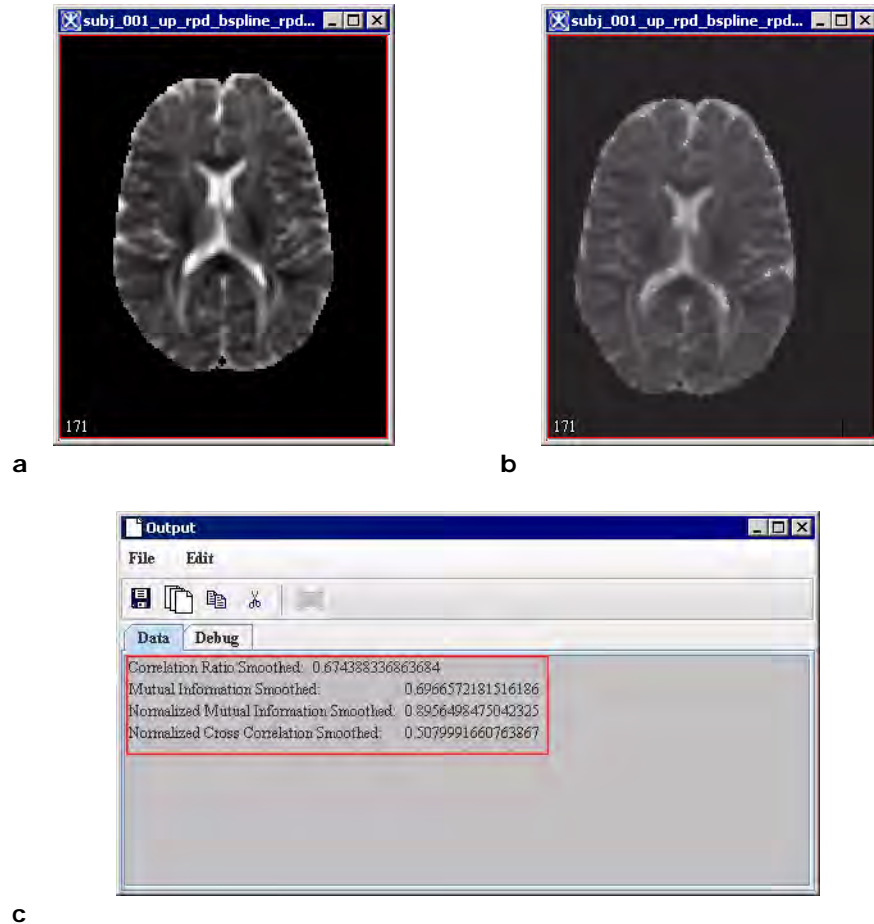


Figure 1. The Display Pixel Similarity Cost Functions method: (a) the first image, (b) the second image, and (c) the output of the method.

## Applying Display Pixel Similarity Cost Function

To use this algorithm, do the following:

- 1 In the MIPAV window, select Algorithms >Registration > Display Pixel Similarity Cost Functions. The Show Pixel Similarity Cost Function Values dialog box (Figure 2) appears.
- 2 Complete the information in the dialog box. See Figure 2.

- 3 Click OK. The algorithm begins to run, and a progress bar appears momentarily with the status. When the algorithm finishes running, the progress bar disappears, and the results appear in the Data tab in the Output MIPAV window.

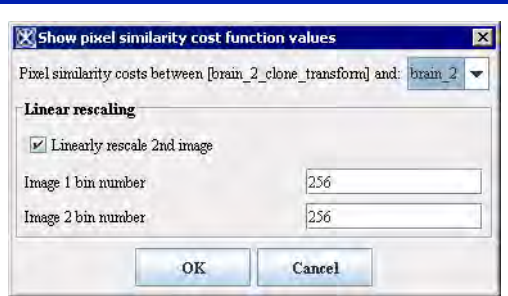
<b>Pixel similarity costs between [Image 1] and [Image 2]</b>	Use the list box, to select the second image from the list of available images. Note that both images must be of the same type– color, or grayscale, or black and white.	
<b>Linearly rescale 2nd image</b>	Because both images must have the same dimensions and extends, the second image might require rescaling in order to match the first one. Check this box to rescale the second image.	
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.	
<b>Cancel</b>	Disregards any changes that you've done in the dialog box and closes this dialog box.	
<b>Help</b>	Displays online help for this dialog box.	

Figure 2. Show Pixel Similarity Cost Function Values dialog box

---

## Registration: Time Series Optimized Automatic Registration

*This algorithm allows you to register the individual 3D volumes within a 4D time series to a reference volume. The reference volume is either a particular one of the 3D volumes or an average of all of the 3D volumes.*

### Background

The Automatic Time Series Registration algorithm is based on MCFLIRT (Motion Correction—FMRIB's Linear Image Registration Tool), a tool developed at the Oxford Centre for Functional Magnetic Resonance Imaging of the Brain at the John Radcliffe Hospital, University of Oxford.

To use this algorithm, first select a 4D time series volume requiring registration. Then select Algorithms > Registration > Time series optimized automatic registration in the MIPAV window. The Time Series Registration dialog box (Figure 1 on page 674) opens.

---

### REGISTERING VOLUMES

Note that the fields in the dialog box appear in two groups: Input Options, which apply to the 4D time series requiring internal registration, and Output Options, which apply to displays for the transformed 4D series. There are two choices for choosing a reference volume. You either choose a specific volume in the time series by selecting Register to reference volume (1 through the number of volumes in the time series) and typing the number of the volume in the box on the right, or you select Register to average volume, which averages all of the volumes in the reference volume. The algorithm rescales the reference volume so that the voxels are isotropic, that is, have equal  $x$ ,  $y$ , and  $z$  dimensions.

If the reference volume exceeds a threshold of 75,000 voxels, the algorithm creates versions of the reference volume that are subsampled first by 2, then by 4, and, finally, by 8. Although three subsamplings are possible, only 2, 1, or 0 subsamplings may occur if the reference volume can be subsampled while it has greater than 75,000 voxels.

The algorithm performs the following cycle with different tolerances used at each level with first the subsampled-by-8 reference volume; then the subsampled-by-4 reference volume; then the subsampled-by-2 reference volume; and, finally, the original reference volume.

For each volume in the time series, the algorithm starts at the reference volume + 1 and proceeds to the last volume in the time series. It then starts at the reference volume - 1 and proceeds to the first volume in the time series.

For each volume, the algorithm does the following:

- 1** Isotropically rescales the current volume
- 2** Subsamples the current volume to the same subsampling level as the reference volume
- 3** Creates a cost function with the reference volume and the current volume and uses Powell's algorithm to optimize with the tolerances specific to the subsampling level
- 4** Stores the answer as a list of matrices, which it uses for the starting point in the next level of subsampling

To perform the cycle with only unsubsampled images at the expense of speed, make sure to clear Subsample image for speed in the Time Series Registration dialog box.

To speed the registration, select Set final subsample at level 2 to cause the last set in the cycle using full-sized images to be skipped.

## CHOOSING A COST FUNCTION

The default cost function is normalized cross correlation. However, you can also select correlation ratio, least squares, and normalized mutual information as cost functions.

### Normalized cross correlation

Normalized cross correlation is:

$$N = \frac{\text{var}(RI)}{\sqrt{\text{var}R \cdot \text{var}I}}$$

where  $N$  is the normalized cross correlation and  $\text{var}$  is the variance.

$$\text{var}RI = \frac{\text{sum}(RI)}{(\text{count} - 1)} - \frac{(\text{sum}R \cdot \text{sum}I)}{((\text{count} - 1) \cdot \text{count})}$$

$$\text{var}R = \frac{\text{sum}(R^2)}{(\text{count} - 1)} - \frac{(\text{sum}R \cdot \text{sum}R)}{((\text{count} - 1) \cdot \text{count})}$$

$$\text{var}I = \frac{\text{sum}(I^2)}{(\text{count} - 1)} - \frac{(\text{sum}I \cdot \text{sum}I)}{((\text{count} - 1) \cdot \text{count})}$$

where

$R$  = the reference image value - reference image minimum value

$I$  = the input image value - input image minimum value

$\text{count}$  = total number of values

## Correlation ratio

For correlation ratio, the algorithm goes over all pixels forming  $numY$ ,  $sumY$ , and  $sumY2$ :

$$iV = (int)(R(index) - minR) \times constant$$

$$numY[iV] = numY[iV] + 1$$

$$sumY[iV] = sumY[iV] + I - minI$$

$$sumY2[iV] = sumY2[iV] + (I - minI)^2$$

where

$iV$  = indexValue

$int$  = integer part of a number

$I$  = input image value

$R$  = reference image value

$minI$  = minimum input image value

$minR$  = minimum reference image value

The algorithm next calculates the individual variances for each ISO-set weighting them by the number of pixels from the input image that contribute.

Summing from  $b = 0$  to  $nBins - 1$ :

$$numTotY = numTotY + numY[b]$$

$$totSumY = totSumY + sumY[b]$$

$$totSumY2 = totSumY2 + sumY2[b]$$

The following should be the  $b$ th ISO-subset variance:

$$varB = (sumY2[b] - sumY[b] \times sumY[b] / numY[b]) / (numY[b] - 1)$$

$$Ratio = corrRatio + varB \times numY[b]$$

The algorithm then normalizes the weighting of  $numY$ ]:

$$corrRatio = (corrRatio) / (numTotY)$$

Next, it calculates the total variance of input image and then normalizes by the following:

$$varTot = (totSumY2 - totSumY \times totSumY / numTotY) / (numTotY - 1)$$

$$corRatio = corRatio / variance$$

## Least squares

For least squares, the algorithm uses the following:

$$LS = \frac{\sum (valueR - valueI)^2}{count}$$

where

*LS* = least squares

*valueR* = reference image[i] - minimum reference image value

*valueI* = input image[i] - minimum input image value

*count* = the number of values over which the sum is performed

## Normalized mutual information

For normalized mutual information, the algorithm uses the following:

$$M = H(R, I) / (H(R) + H(I))$$

where

*M* = normalized mutual information

*R* = reference image values

*I* = input image values

*q* = locations

$$H(R, I) = \sum_{j, k} p_{jk} \log(p_{jk})$$

where

$p_{jk} = (\text{Number of elements for which Value } j < R(q) < \text{Value } j+1 \text{ and Value } k < I(q) < \text{Value } k+1) / \text{total number of elements}$

$H(R) = -\text{sum over } j \text{ of } (p_j \log(p_j))$  where  $p_j = (\text{Number of elements for which Value } j < R(q) < \text{Value } j+1) / \text{total number of elements}$

$H(I) = -\text{sum over } k \text{ of } (p_k \log(p_k))$  where  $p_k = (\text{Number of elements for which Value } k < I(q) < \text{Value } k+1) / \text{total number of elements}$

The normalized mutual information definitions require the specification of a partition of intensities:  $\{\text{Value } 0, \text{Value } 1, \dots, \text{Value } M\}$ .

---

## COMPLETING THE DIALOG BOX

Unless the algorithm is in the final cycle with full-sized images and you selected Finalize with normalized cross correlation sinc, the input image value is obtained with trilinear interpolation. In that case, windowed sinc interpolation is used to obtain the input image value in a normalized cross-correlation routine.

To show the resulting image, select Display transformed image.

By default, the interpolation used to calculate the transformed image from the array of transformation matrices is trilinear. However, you can select bspline 3rd order, bspline 4th order, cubic lagrangian, quintic lagrangian, heptic lagrangian, or windowed sinc.

If you select graph rotations and translations, MIPAV produces two graphs: the first graph has  $x$ ,  $y$ , and  $z$  translations, and the second graph contains  $x$ ,  $y$ , and  $z$  rotations.

---

## IMAGE TYPES

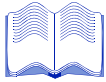
The algorithm works on both 4D color and grayscale images.



---

## SPECIAL NOTES

None.



---

## REFERENCE

Refer to the following references for more information:

For more information on MCFLIRT, visit the MCFLIRT home page at <http://www.fmrib.ox.ac.uk/fsl/mcflirt/index.html>. The main paper is: Jenkinson, M., Bannister, P., Brady, M., and Smith, S., "Improved optimisation for the robust and accurate linear registration and motion correction of brain images," *NeuroImage*, 17(2): 825-841.

Jenkinson, Mark, and Smith, Stephen, "A global optimisation method for robust affine registration of brain images," *Medical Image Analysis*, 5(2): 143-156.

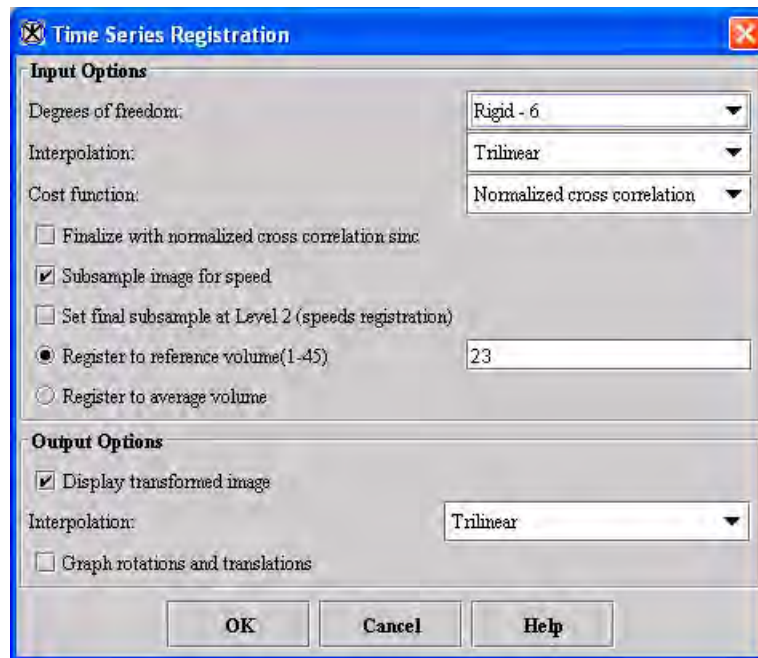
## Applying the Time Series Optimized Automatic Registration algorithm

To run this algorithm, complete the following steps:

- 1 Open an image.
- 2 Perform, as an option, any other image processing on the image.
- 3 Select Algorithms > Registration > Time Series optimized automatic registration. The Time Series Registration dialog box (Figure 1) opens.
- 4 Complete the information in the dialog box.
- 5 Click OK.

The algorithm begins to run, and a progress bar appears with the status. When the algorithm finishes running, the progress bar disappears, and the results appear in a new window.

If you selected Display transformed image, the transformed image appears in a new window. If you also selected Graph rotations and translations, a graph of rotations and a graph of translations appear.



<b>Degrees of freedom</b>	Specifies the spatial model used to restrict the type of linear transformation being applied during registration. Select one of the following: rigid-6, global rescale-7, specific rescale-9, and affine-12.
<b>Interpolation</b>	Select interpolation if isotropic resampling is needed. Select one of the following: trilinear (default choice), bspline 3rd order, bspline 4th order, cubic lagrangian, quintic lagrangian, heptic lagrangian, and windowed sinc.
<b>Cost function</b>	Specifies a function that measures the deviation of output from input. Select one of the following: correlation ratio, least squares, normlized cross correlation, and normalized mutual information.
<b>Finalize with normalized cross correlation sinc</b>	Obtains the input image value for full-sized images by performing, as the last step of registration, a normalized cross correlation cost function, which uses windowed sinc interpolation. The default is not selected.
<b>Subsample image for speed</b>	Subsamples first by 8 times, then by 4 times, then 2 times, and, finally, by an unsampled image if the image is large enough. If not selected, all steps of the algorithm use full-size images. The default is selected.
<b>Set final subsample at Level 2 (speeds registration)</b>	Skips the last step in the algorithm that uses full-size images. The default is not selected.

Figure 1. The Time Series Registration dialog box

<b>Register to reference volume (1-number of volumes in times series)</b>	Registers the volume to be registered, or <i>adjusted volume</i> , to the reference volume you specify. For example, typing the number 22 in the text box on the right indicates that the adjusted volume should be registered to volume 22 of the reference volume.
<b>Register to average volume</b>	Registers all volumes in the time series to the average of all of the volumes.
<b>Display transformed image</b>	Shows the resulting image of the registered volumes.
<b>Interpolation</b>	Calculates the output image from the array of transformation matrices and input image. You can select trilinear, bspline 3rd order, bspline 4th order, cubic lagrangian, quintic lagrangian, helptic lagrangian, or windowed sinc. The default choice is trilinear.
<b>Graph rotations and translations</b>	Produces two graphs: one graph includes the <i>x</i> , <i>y</i> , and <i>z</i> translations and the second graph contains <i>x</i> , <i>y</i> , and <i>z</i> rotations.
<b>Display transformed image</b>	Display transformed image If selected, the transformed image appears in a new window. By default, this option is selected.
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes this dialog box.
<b>Help</b>	Displays online help for this dialog box.

Figure 1. The Time Series Registration dialog box (continued)

## Reslice—Isotropic Voxels

*Image resampling is an interpolation process in which anisotropic voxels (in the Z direction) are resampled into isotropic-sized voxels. The new voxels are formed by taking the weighted combination of the source voxels within a defined neighborhood of the reference voxel. Three interpolation methods are supported: linear, cubic convolution, and cubic B-spline.*

Refer to “Optimized Automatic Registration 3D” on page 596 for resampling an image at various orientations using a choice of interpolation methods.

### Background

This section shows the equations that MIPAV uses for the three interpolation methods for the algorithm. Figure 1 shows an original image and the resulting images after linear, cubic convolution, and cubic B-spline interpolation are applied, see also “Interpolation methods used in MIPAV”.

#### LINEAR INTERPOLATION

The equations for linear interpolation are the following:

$$\begin{aligned} R(u) &= u + 1 && \text{for} && 0 \leq u < 1 \\ R(u) &= 1 - u && \text{for} && 0 \leq u < 1 \end{aligned}$$

#### CUBIC CONVOLUTION INTERPOLATION

The equations that apply to cubic convolution interpolation are:

$$\begin{aligned} R(u) &= \frac{3}{2}|u|^3 - \frac{5}{2}|u|^2 + 1 && \text{for} && 0 \leq |u| < 1 \\ R(u) &= -\frac{1}{2}|u|^3 + \frac{5}{2}|u|^2 - 4|u| + 1 && \text{for} && 1 \leq |u| < 2 \end{aligned}$$

## CUBIC B-SPLINE INTERPOLATION

MIPAV uses the following equations for cubic B-spline interpolation:

$$R(u) = \frac{3}{2} + \frac{1}{2}|u|^3 - |u|^2 \quad \text{for} \quad 0 \leq |u| < 1$$

$$R(u) = \frac{1}{6}[2 - |u|]^3 \quad \text{for} \quad 1 \leq |u| < 2$$

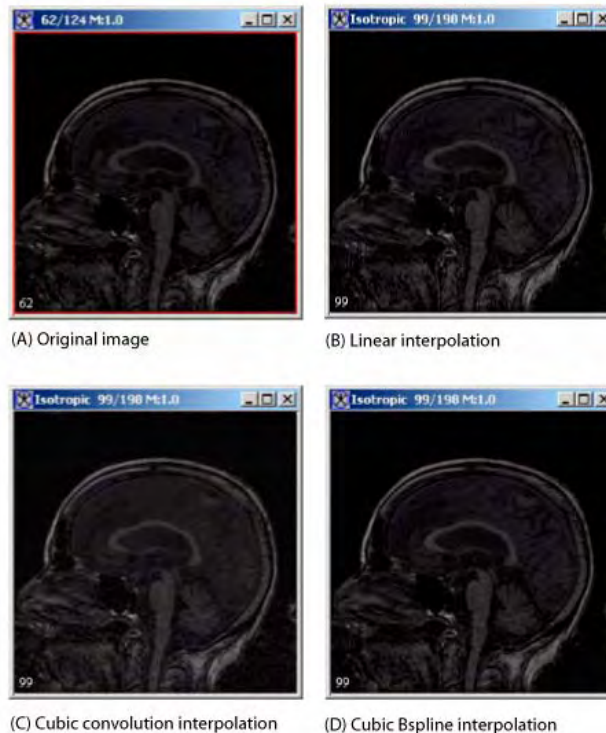


Figure 1. Reslice—Isotropic Voxels algorithm applied to an image

## REFERENCES

None.

---

## IMAGE TYPES

You can apply this algorithm to all image data types except RGB and complex. This algorithm can not be applied to 2D and 4D images.

## Applying the Reslice—Isotropic Voxels algorithm

To run this algorithm, complete the following steps:

- 1 Open an image.
- 2 Select Algorithms > Reslice—Isotropic Voxels. The Reslice dialog box (Figure 2) opens.
- 3 Select an interpolation method in Interpolation.
- 4 Click OK.

A status message appears. When the algorithm is complete, the results appear in a new image window.

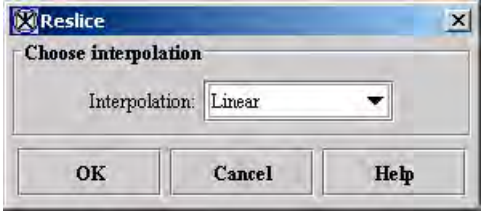
<b>Interpolation</b>	Specifies the type of interpolation to use when applying this algorithm. You can select one of the following: linear, cubic, or cubic B-spline.	
<b>OK</b>	Applies the Reslice—Isotropic Voxels algorithm according to the specifications in this dialog box.	
<b>Cancel</b>	Disregards any changes you made in this dialog box; closes the dialog box.	
<b>Help</b>	Displays online help for this dialog box.	

Figure 2. Reslice dialog box

---

## Registration: VOI Landmark

The VOI Landmark algorithm is a template matching motion correction technique that uses Gradient Magnitude as a similarity measure. This method replaces a tedious manual motion correction by a semi automatic procedure. The user interaction is reduced to a selection of a reference slice (it should be the first image slice) and a manual segmentation of the image in this slice.

The VOI Landmark algorithm can be applied for automatic motion correction in dynamic contrast enhanced MRI (DCE-MRI) perfusion scans which are used for the measurement of renal function.

### Background

There are two main challenges for registration of the images in perfusion scans – the first is the extremely strong intensity variations over the time series, which can lead to apparent geometry changes – and the second is large displacements between single images in the time series.

As for the assessment of kinetic parameters for kidney, the shape of the kidney structures must be maintained during the motion correction process, therefore the algorithm employs the rigid motion model that implicitly maintains the structures during registration. Since the major motion of the kidney during the acquisition can be captured by a translation, the algorithm uses a *template matching technique* that constrains the rigid motion to translational and small rotational movements only. This also allows saving the computation time.

---

### TEMPLATE MATCHING REGISTRATION TECHNIQUE

The template matching registration technique selects only similar areas in the images to determine the registration parameters (instead of using information from entire images). In this method, the user delineates a VOI on the first image slice and this VOI is used as a template for registration.

---

## AS A SIMILARITY MEASURE, THE METHOD USES GRADIENT MAGNITUDE

Two images are considered similar, if intensity changes occur at the same locations. An image intensity change can be detected via the image gradient. Since the algorithm registers the slices of the same image (which automatically means the same modality), the Gradient Magnitude was chosen as a similarity measure. See also “Cost functions used in MIPAV algorithms” .

---

The method uses a negative sum of Gradient Magnitudes as a cost function.

---

---

## TECHNIQUES FOR SEARCHING A GLOBAL MINIMUM

The method uses the coarse-to-fine search approach and two optimization algorithms to find the global minimum. The algorithms are as follows – the Exhaustive search for a global minimum and Downhill Simplex search for a global minimum.

In the coarse-to-fine approach, first, the images are reduced in scale and approximate match positions are determined. Then, the scales of the images are increased in two to three steps until full-resolution images are obtained. At each step, possible match positions are determined with associated confidence levels and a search at a higher resolution is performed only for high confidence matches.

The algorithm resamples images using an interpolation scheme, where anisotropic voxels (in the Z direction) are resampled into isotropic 1mm cubic voxels. New voxel values are computed using a weighted combination of existing voxel values within a defined neighborhood of the new voxel location. The algorithm uses the Gaussian interpolation technique with the scale varies from 0.5 to 5.0.

See also “Cost functions used in MIPAV algorithms” and “Interpolation methods used in MIPAV” .



### Exhaustive search for a global minimum

The exhaustive search for a global minimum, first, produces the topographic map for the chosen cost function - the cost surface. This narrows the search for a global minimum to the large but still finite solution space with the number of combinations of different variable values given by:

EQUATION 1

$$N = \prod_{i=1}^{Nvar} Qi$$

where,

**N** is a number of different variable combinations

**Nvar** is a total number of different variables

**Qi** is a number of different values that i can attain

It, first, searches for minima in a relatively large region of the cost surface at the coarse sampling and stores the best minima in the vector of best minima. Then, it progressively narrows the search to the best solutions until it finds the best minima.

### Downhill simplex search for a global minimum

The Downhill simplex method constructs  $n$  vectors  $p_i$  from the vector of initial parameter values  $x$  and the vector of step-sizes  $s$  as follows:

EQUATION 2

$$p_i = (x_0; x_1; \dots; x_i + s_i; \dots; x_n)$$

These vectors form  $(n + 1)$  vertices of an  $n$ -dimensional simplex. During each iteration step, the algorithm improves parameter vectors  $p_i$  by modifying the vertex with the highest function value by simple geometrical transformations: reflection, reflection followed by expansion, contraction and multiple contraction. Using these transformations, the simplex moves through parameter space towards the deepest minimum, where it contracts itself.

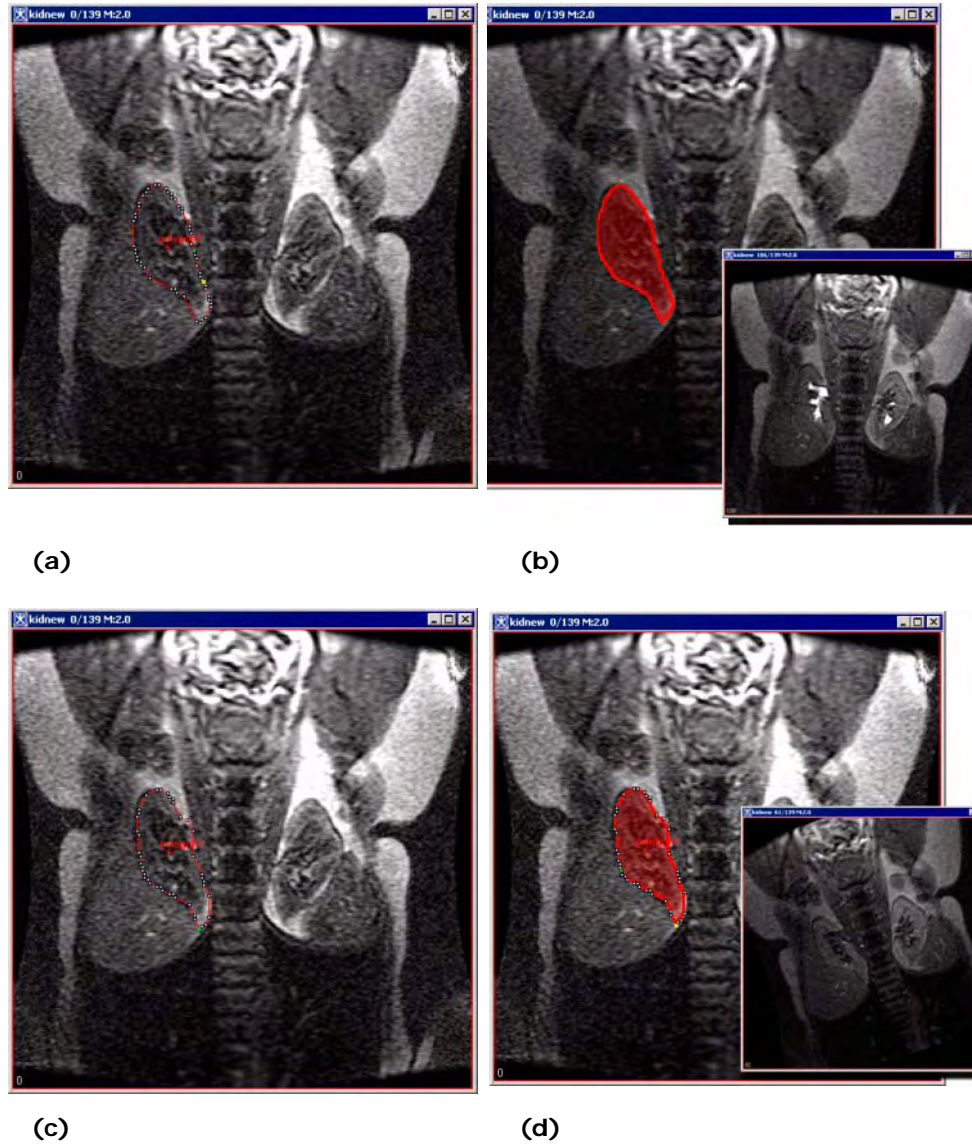


Figure 1. Applying the VOI landmark algorithm: (a) the first image slice with delineated VOI, (b) the result image when using the Exhaustive search for a global minimum, (c) the first slice of another image with delineated VOI, (d) the result image when using the Downhill Simplex search for a global minimum

---

## REFERENCES

The following literature was used:

N. Grenier, O. Hauger, A. Cimpean, and V. Perrot, "Update of renal imaging," *Seminars in Nuclear Medicine* 36(1), pp. 3-15, 2006.

H. J. Michaely, S. Sourbron, O. Dietrich, U. Attenberger, M. F. Reiser, and S. O. Schoenberg, "Functional renal MR imaging: an overview," *Abdom. Imaging*, 2006.

Darko Zikic, Steven Sourbron, Xinxing Feng, Henrik J. Michaely, Ali Khamene, and Nassir Navab, "Automatic Alignment of Renal DCE-MRI Image Series for Improvement of Quantitative Tracer Kinetic Studies" *SPIE Medical Imaging*, San Diego, California, USA, 16-21 February 2008.

---

## IMAGE TYPES

All 3D and 4D images that can be opened in MIPAV.

## Applying the VOI Landmark algorithm

To run this algorithm, complete the following steps:

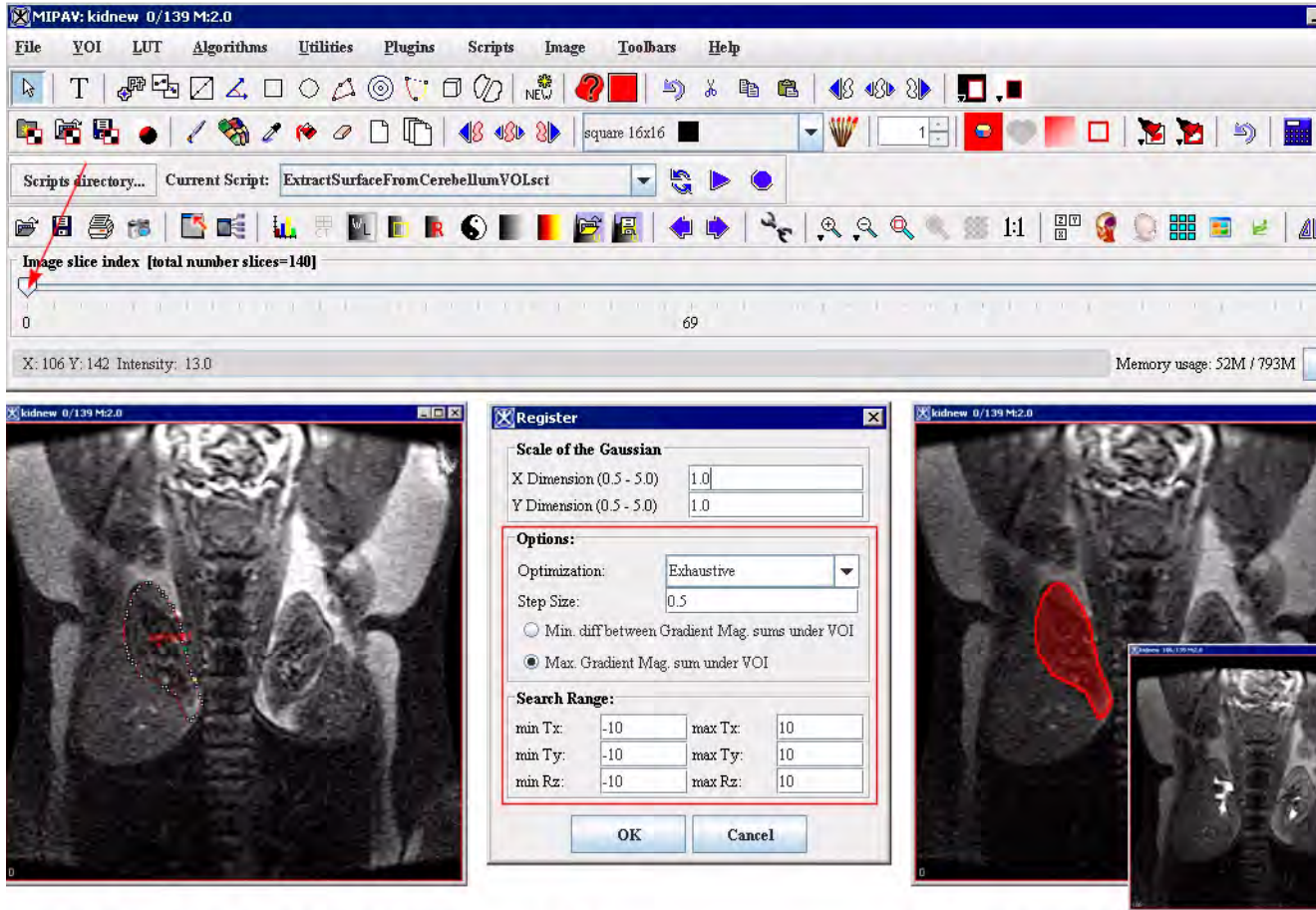
- 1 Open a 3D kidney image.
- 2 Use the Image Slice Slider to navigate to the first slice (the slice #0).
- 3 Use the Draw Polygon/Poly line VOI icon to delineate a VOI around the right or left kidney.
- 4 Call Algorithms> Registration >VOI Landmark.
- 5 In the Register dialog box that appears, specify the scale of Gaussian for X and Y dimensions.
- 6 Then, select the optimization algorithm: **Exhaustive search for a global minimum** or **Downhill Simplex search for a global minimum**. See Figure 1 and Figure 2.

### For Exhaustive search only:

- Specify what value will be used to determine the global minimum, it could be either Minimum Difference between Gradient Magnitude Sums under VOI or negative Maximum Gradient Magnitude Sum under VOI.
  - Specify the search range.
- 7 Press OK.

The algorithm begins to run and the progress window appears displaying

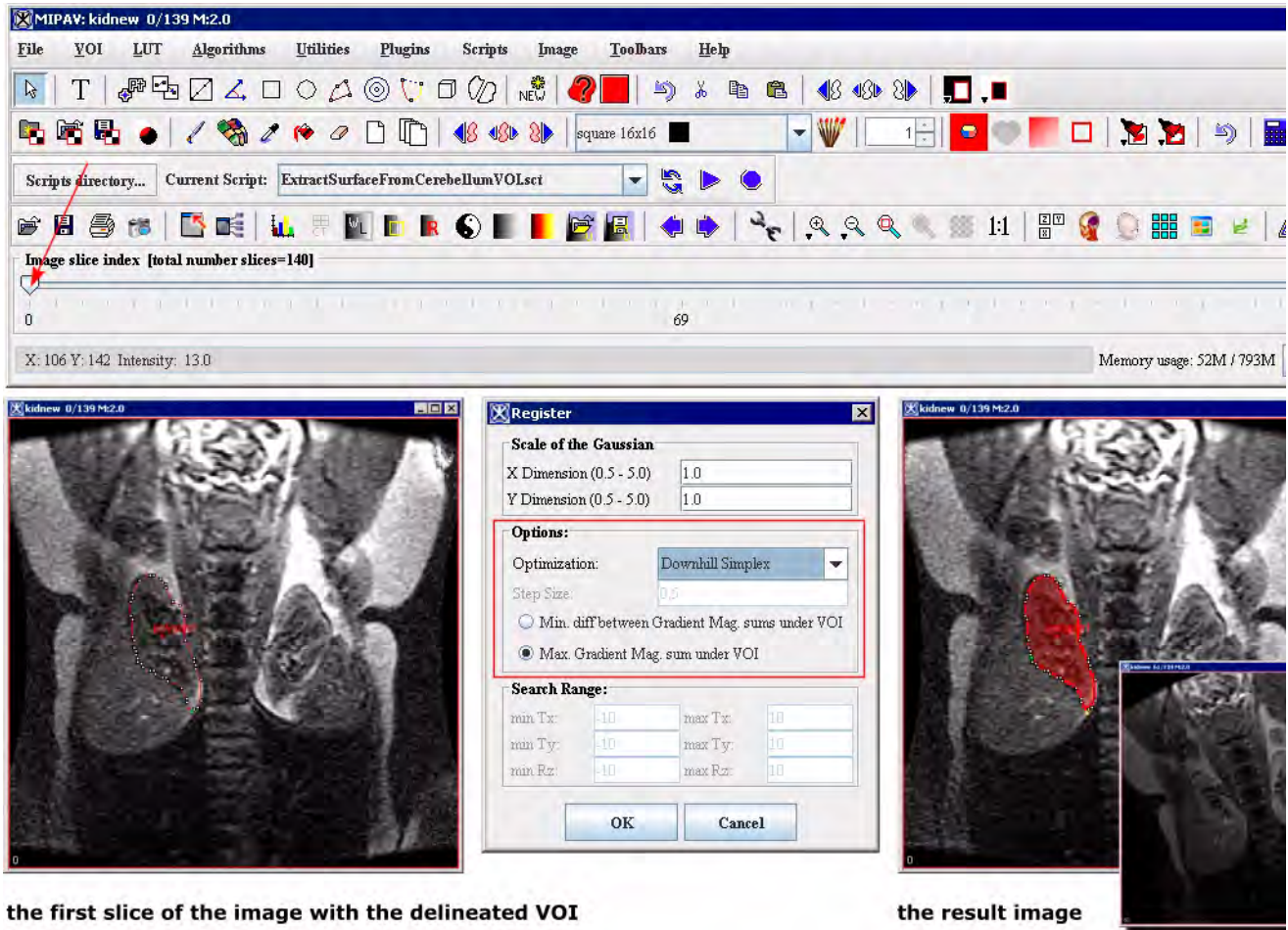
the progress. When the algorithm finishes running, the registered image appears in **the same image frame**.



the first slice of the image with the delineated VOI

the result image

Figure 2. Applying the Registration- VOI Landmark algorithm using the Exhaustive search option



the first slice of the image with the delineated VOI

the result image

Figure 3. Applying the Registration- VOI Landmark algorithm using the Downhill Simplex search option

<b>Scale of Gaussian</b>	Specify the scale of Gaussian used in the Gaussian interpolation (0.5–5.0).	
<b>Options</b>	<b>Optimization</b> – select either Exhaustive search or Downhill Simplex search.	
<b>The following options available only for Exhaustive search</b>		
<b>Step size</b>	– specify the step size (0.5–5.0) used to search for the min cost.	
<b>Min. diff. between Gradient Magnitude under VOI</b>	– if selected, this value will be used to determine the min cost.	
<b>Max. Gradient Magnitude under VOI</b>	– if selected, this value will be used to determine the min cost.	
<b>Search range</b>	Use the text boxes maxTx, maxTy, minTx, minTy, maxRz and minRz to specify the rotation and translation steps.	
<b>OK</b>	Applies the parameters that you specified to register the image.	
<b>Cancel</b>	Disregards any changes you made in this dialog box, closes the dialog box, and does not register the image.	
<b>Help</b>	Displays online help for this dialog box.	

Figure 4. Register (VOI Landmark) dialog box

## Shading Correction: Inhomogeneity N3 Correction

*This preprocessing algorithm corrects for shading artifacts often seen in MRI. The heart of the algorithm is an iterative approach that estimates both a multiplicative bias field and a distribution of true tissue intensities. Referred to as nonparametric intensity nonuniformity normalization (N3), this method makes no assumptions about the kind of anatomy present in a scan and is robust, accurate, and fully automatic.*

### Background

A common difficulty with tissue intensity model based methods is the determination of the model. The modeling assumption in the N3 algorithm is that of a field model: the assumption that nonuniformity blurs the histogram of the data in a way that it can be identified, quantified, and removed. This nonuniformity blurring distribution is referred to as the blurring kernel  $F$ . The following equation is the basis of the N3 method:

EQUATION 1

$$v(x) = u(x) \times f(x) + n(x)$$

Where at location  $x$ ,  $v$  is the measured signal,  $u$  is the true signal emitted by the tissue,  $f$  is an unknown smoothly varying bias field, and  $n$  is white Gaussian noise assumed to be independent of  $u$ . As can be seen, the unknown smoothly varying bias field  $f$  multiplies the true signal  $u$ , thereby corrupting it. Then the additive noise  $n$  is added, resulting in the output measured signal  $v$ . To compensate for the intensity nonuniformity, the first task is to estimate its distribution, or  $f$ .

Then taking the  $\log$  of all parameters, we are left with  $\hat{u} = \log(u)$  and  $\hat{f} = \log(f)$ , both of which we assume to be independent or uncorrected random variables of probability densities  $U$ ,  $V$ , and  $F$ . Then substituting these variables back into Equation 1 and converting the equation of distributions into one of probability densities, the distribution of their sum turns into the following convolution, see Equation 2:

$$V(\hat{v}) = F(\hat{v}) \times U(\hat{u}) = F(\hat{v} - \hat{u})U(\hat{u})d(\hat{u})$$

Equation 2 models the nonuniformity distribution  $F$  viewed as blurring the intensity distribution  $U$ .

Now the model of intensity nonuniformity used here is that of a multiplicative field corrupting the measured intensities. However, viewing the model from a signal processing perspective, one sees that there is a much more specific consequence, a reduction of only high-frequency components of  $U$ . Thus, the approach used in the N3 algorithm to correct for nonuniformity turns into an optimization problem, which goal is to maximize the frequency contents of  $U$ .

---

## HOW THE N<sub>3</sub> METHOD WORKS

We begin with only a measured distorted signal, or image, which we call  $v$ . For computational purposes, we do not deal with simple  $v$ , but instead, with  $\log^{\wedge}v$  and the histogram distribution of the log, and use  $\wedge v$  to denote  $V$ . The same representations are used for the true signal,  $u$ , and the distortion signal,  $f$ .

The model assumed here is that of a multiplicative biasing field that multiplies the true signal to distort it, thereby creating a signal with induced field inhomogeneity. So, just as a multiplication takes place to corrupt the field, a division should undo the corruption. Now carrying this idea over to the frequency domain where all of the computations take place, *multiplications turn into convolutions* and *divisions turn into deconvolutions*.

To begin with, all one has is  $v$ . Users provide a value for the kernel *full-width, half maximum* (FWHM) of  $f$ . Then, the iteration takes place as this estimate of  $f$  is later used to compute  $U$  and then  $U$  input back into the algorithm to create another mapping, another  $f$ , and hence another  $U$ . The process then repeats until convergence. The result is an estimated true value of  $U$ .



## Mapping from $v$ to $f$

The approach adopted for finding  $U$  depends largely on the mapping from  $v$  to  $f$ . The idea is to propose a distribution of  $U$  by sharpening the distribution  $V$  and then estimating the corresponding smooth field  $\log(f)$ , which produces a distribution  $U$  close to the one proposed. However, rather than searching through the space of all distributions  $U$ , one can take advantage of the simple form of the distribution  $F$ . Suppose the distributions of  $F$  is Gaussian. Then one need only search the space of all distributions  $U$  corresponding to Gaussian distributed  $F$  having zero mean and given variance. In this way the space of all distributions  $U$  is collapsed into a single dimension, the width of the  $F$  distribution. Thus, concludes the mapping N3 creates from  $v$  to  $f$ .

Estimating  $u$  throughout an entire volume:

EQUATION 3

$$E(\hat{u}) = \frac{\int_{-\infty}^{\infty} \hat{u} F(\hat{v} - \hat{u}) U(\hat{u}) d(\hat{u})}{\int_{-\infty}^{\infty} F(\hat{u}) U(\hat{u}) d(\hat{u})}$$

Estimating  $u$  (voxel by voxel):

EQUATION 4

$$\hat{v} - \hat{u} = E[\hat{u} | \hat{v}]$$

## Algorithm flow chart

This algorithm performs the process shown in the flow chart in Figure 1. What was previously described was just the basic algorithm idea. The equations behind the steps in the process are shown in an enlarged section of the flow chart in Figure 2.

The algorithm does the following:

- 1** Identifies the foreground. In this step, the algorithm thresholds the background.
- 2** Resamples to working resolution. Use the input working resolution “subsampling factor.”
- 3** Logs transform intensities.
- 4** Obtains the expectation of the real signal. Given a measured signal, the algorithm obtains the expectation of the real signal by performing a voxel by voxel difference between the logs of signal and estimate bias field.
- 5** Obtains an initial estimate of distribution U. The algorithm estimates distribution U, where U is a sharpened distribution of V. The sharpening is done using input parameters FWHM and Z. Given a distribution F and the measured distribution of intensities V, the distribution U can be estimated using a deconvolution filter as follows:

EQUATION 5

$$\tilde{G} = \frac{\tilde{F}^*}{\tilde{F}^2 + \tilde{Z}^2}$$

EQUATION 6

and

$$\hat{U} \approx \hat{G}\hat{V}$$

where

F\* denotes complex conjugate

$\tilde{F}$  is the fourier transform of F

Z is a constant term to limit the magnitude of  $\tilde{G}$

This estimation of U is then used to estimate the corresponding field.

- 6** Computes the expectation of real signal—The algorithm computes the expectation of the real signal given measured signal (both logged). Full volume computation done using distributions U and Gaussian blurring

kernel  $F$  (whose FWHM parameter is given by users). A basic mapping from  $v$  to  $f$ . This is the iterative portion, which is shown the main loop in Figure 1.

- 7** Obtains a new field estimate—The difference between the measured signal and the newly computed expectation of the real signal given the measured signal (*all variables logged*).

EQUATION 7

$$\hat{f}_e(\hat{v}) = \hat{v} - E[\hat{u}|\hat{v}]$$

- 8** Smooths the bias field estimate—Using the B-spline technique with the parameter  $d$  field distance input provided by users.

EQUATION 8

$$f_s(\hat{v}) = S\{f_e(\hat{v})\}$$

- 9** Terminates iterations—The algorithm then checks for the convergence of field estimates. If the subsequent field estimate ratio becomes smaller than the end tolerance parameter input by users, the iterations are terminated. If not, a new  $F$  is used in step 2, and the iteration repeats step 4 through step 9.

EQUATION 9

$$e = \frac{s\{r_n\}}{m\{r_n\}}, n= 1\dots N$$

where  $r_n$  is the ratio between subsequent field estimates at the  $n$ -th location,  $s$  -denotes standard deviation, and  $m$  denotes mean. Typically after 10 iterations,  $e$  drops below 0.001.

- 10** Completes the final steps—Given the measured signal, obtain the expectation of the real signal by calculating a voxel by voxel difference between logs of signal and an estimate of the bias field. Once iteration is complete, exponential transform intensities  $V$  and  $F$ , extrapolate field to volume, resample data back to original solution. Then divide  $v$  by  $f$ . The result is  $u$ .

EQUATION 10

$$u = \frac{v}{f}$$

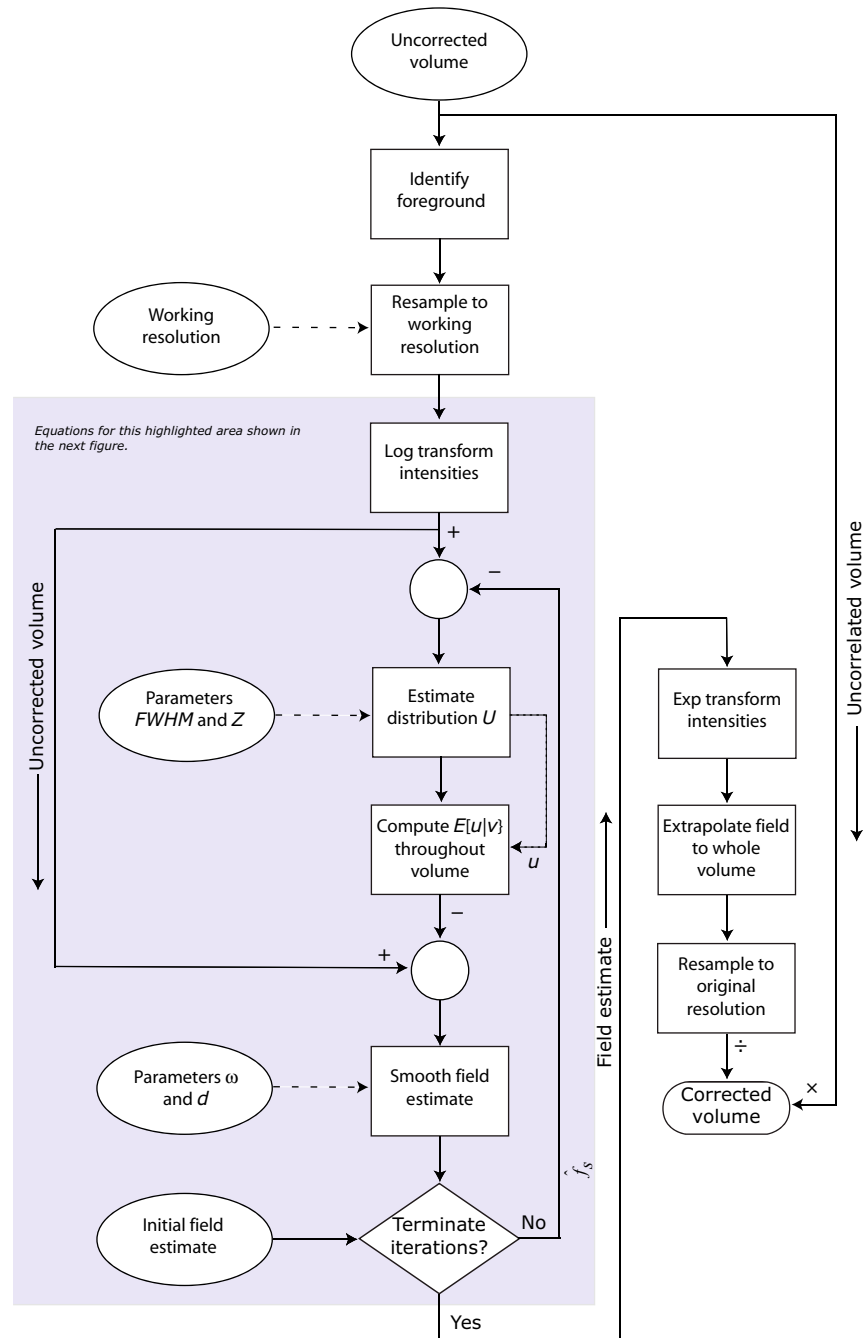


Figure 1. Flow chart describing the N<sub>3</sub> method

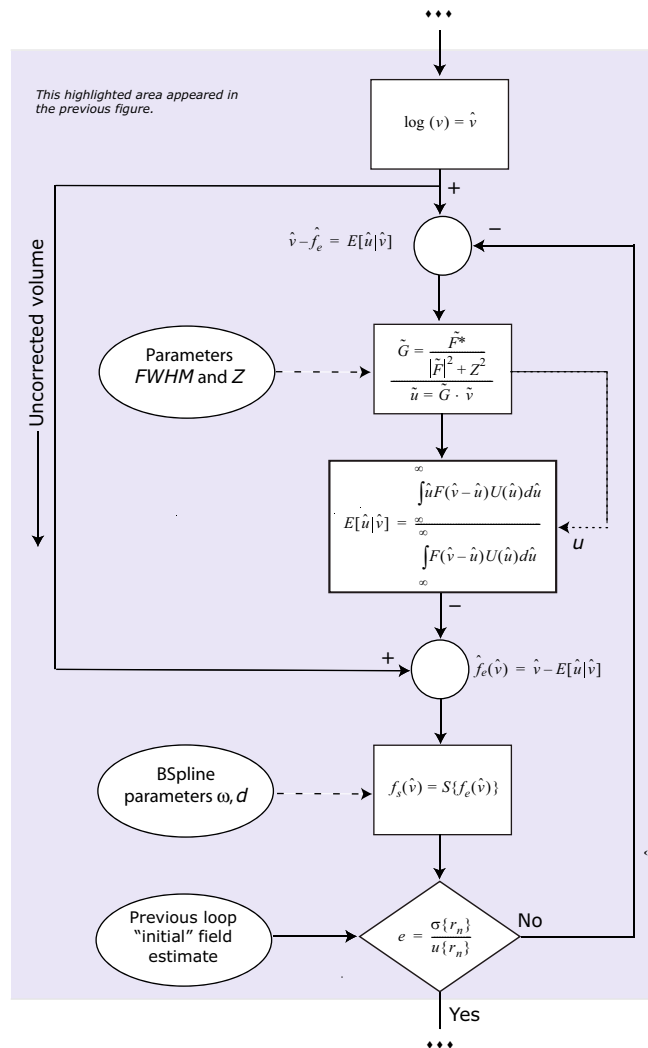
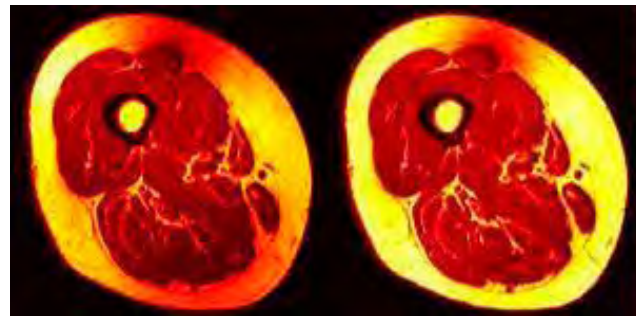


Figure 2. The N<sub>3</sub> method showing the equations



(A) Original image      (B) After N3 correction

**Figure 3. The effect of  $N_3$  on an image: (A) original image and (B) image after  $N_3$  correction**

Figure 3 shows an original image A and the image B produced from using the best  $N_3$  parameters:

- Threshold = 100
- Maximum iterations = 100
- End tolerance = 0.0001
- Field distance = 33.33
- Subsampling factor = 2
- Kernel FWHM = 0.8
- Wiener = 0.01

Figure 4 shows an original image and the effect of different parameters used in the Inhomogeneity  $N_3$  Correction algorithm on the image.

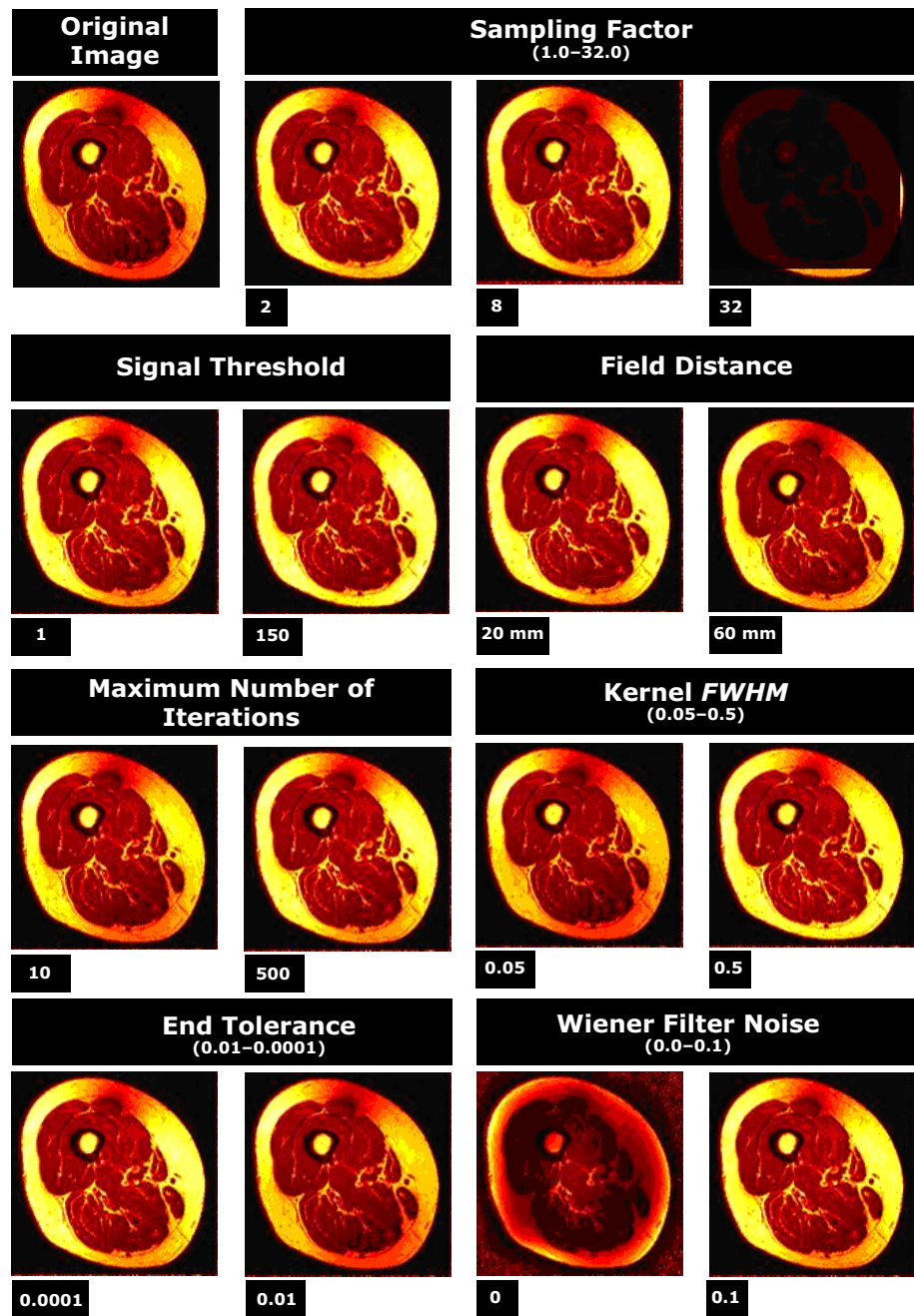


Figure 4. The effects of different settings for the Inhomogeneity N3 algorithm on an image

---

## REFERENCES

Refer to the following references for more information about the N3 algorithm.

Sled, John G., Alex P. Zijdenbos, and Alan C. Evans. "A Nonparametric Method for Automatic Correction of Intensity Nonuniformity in MRI Data." *IEEE Transactions on Medical Imaging*, 17(February 1998):87–97.

Sled, John G. "A Non-Parametric Method for Automatic Correction of Intensity Non-uniformity in MRI Data." Masters thesis, Department of Biomedical Engineering, McGill University, May 1997.

---

## IMAGE TYPES

You should apply this algorithm only to 2D and 3D MRI images.

## Applying the Inhomogeneity N3 algorithm

To run this algorithm, complete the following steps:

- 1** Open an image.
- 2** Select Algorithms > Shading correction > Inhomogeneity N3 correction. The N3 Inhomogeneity Correction dialog box opens.
- 3** Complete the information in the dialog box (refer to "Choosing good parameters for N3 Algorithm" on page 697 for advice on which values to select).
- 4** Click OK. The algorithm begins to run.
- 5** A pop-up window appears with the status. A series of status messages appear in the window.
- 6** When the algorithm finishes running, the pop-up window closes.
- 7** Depending on whether you selected New image or Replace image, the results appear in a new window or replace the image to which the algorithm was applied.



## Choosing good parameters for N3 Algorithm

**Signal threshold**—Check the histogram for the minimum image values in the image. Make sure these values are smaller than the shading artifact. Generally, this value is about 100. To save time, use as high a threshold as possible.

**Maximum number of iterations**—Generally, this parameter is only important if you wish to terminate the process early by entering a number short of the number of iterations necessary for the algorithm to complete the process. However, to allow the algorithm to work to completion, a safe value to enter would be at least 100. Making it too high should have no consequence.

**End tolerance**—When the remaining parameters are chosen optimally, this parameter tends not to have much of an affect. However, with less effective parameters, the end tolerance can make a large difference, bringing the algorithm to work at least sparsely on large artifacts when a minimal end tolerance is chosen. The alternative is a high end tolerance, which can bring the shading correction to be almost nonexistent. In this case, the default value for this parameter is generally all right. For best results, however, use 0.0001. As far as timing, even with the minimum value, the processing time does not vary much.

**Field distance**—A large field distance takes all shading artifacts to be large. Small independent ones are ignored, while small clustered ones are smeared into one. So with large smear-like gradual (light) artifacts, a large parameter works well. However, a small field distance is much more precise. Every little artifact is independently corrected, and the intensity with which the algorithm works is much higher as can be noted in the field images. The field image of an image corrected with a small field distance has several dark clusters, generally distributed throughout the image even to parts where artifact may not be noticeable even to users. The field image of an image corrected with a large field distance may have one or two large smears, and nothing more.

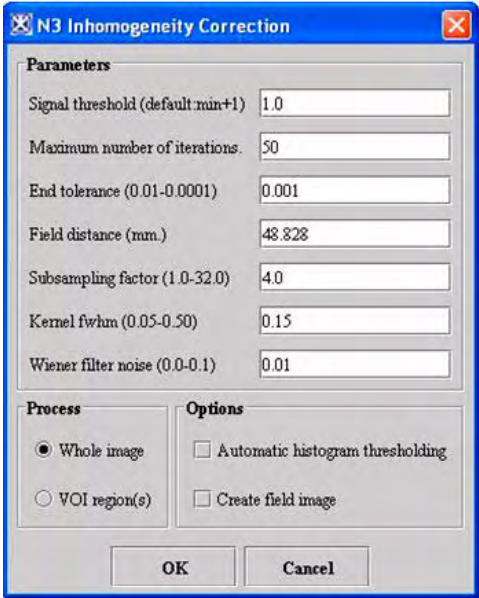
The main trade-off here is time. Very small field distances can increase the processing time immensely. So the lowest one should go for optimal results yet without spending too much time on one image, which has been found to be about 10. That value should really only be used with several intense and isolated artifacts. For regular images a third of the default is sufficient. For

quick results or for artifacts largely spread and not too strong, choose a value that is a little less than the default value, but more than half of the default value.

**Subsampling**—For best results, specify 1. However, keep in mind that the higher the precision (lower the subsampling), the higher the likelihood that certain low-intensity pixels inside the image are treated as background creating a speckle artifact.

**Kernel FWHM**—Use 0.5.

**Wiener**—Use 0.01.

<b>Signal threshold</b>	Treats those voxels of intensity that are lower than this threshold value as background.  The default value is 1.0.	
<b>Max number iterations</b>	Specifies the maximum number of iterations allowed before the process is terminated.  The default value is 50.	
<b>End tolerance</b>	Specifies convergence when all membership functions (of the fuzzy C cluster segmentation) over all pixel locations $j$ change by less than this tolerance value between two iterations. The default value is 0.001.	
<b>Field distance</b>	Specifies the characteristic distance over which the field varies. The distance between adjacent knots in B-spline fitting with at least 4 knots going in every dimension. The automatic default is one third of the distance (resolution * extents) of the smallest dimension.	
<b>Subsampling Factor</b>	Specifies the value by which data is subsampled to a lower resolution in estimating the slowly varying nonuniformity field. It reduces sampling in the finest sampling direction by the shrink factor. Sampling in other directions is reduced only if reduced sampling in direction of finest resolution is less than current sampling. The default value is 4.	

<b>Kernel FWHM</b>	Specifies the width of deconvolution kernel $F$ (a Gaussian) used to sharpen the $V$ (pdf of $\log(v)$ , where $v$ is the measured signal) histogram. Larger values give faster convergence while smaller values give greater accuracy. The default value is 0.01.
<b>Wiener Filter Noise</b>	Indicates the noise used in Wiener filter.
<b>Whole image</b>	Applies the algorithm to the whole image.
<b>VOI region(s)</b>	Applies the algorithm to the volumes (regions) delineated by the VOIs.
<b>Automatic histogram thresholding</b>	Computes a threshold value through histogram analysis that should remove just the background from the computation. Selecting this check box disables the Signal threshold box, the Whole image, and VOI regions.
<b>Create field image</b>	Indicates whether to create a field image.
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.
<b>Cancel</b>	Disregards any changes you made in this dialog box and closes the dialog box.
<b>Help</b>	Displays online help for this dialog box.

**Figure 5. N3 Inhomogeneity Correction dialog box**

# Standard Deviation Threshold

## Background

Standard Deviation Threshold works by, first, examining an active VOI for which the standard deviation (st.dev) of pixel intensities and other statistics are calculated. The algorithm, then, thresholds an image using the user defined parameters, such as a number of standard deviations and/or values outside the range. The thresholding can be done on the whole image or, alternatively, on other not active VOI(s) that might be on the image.

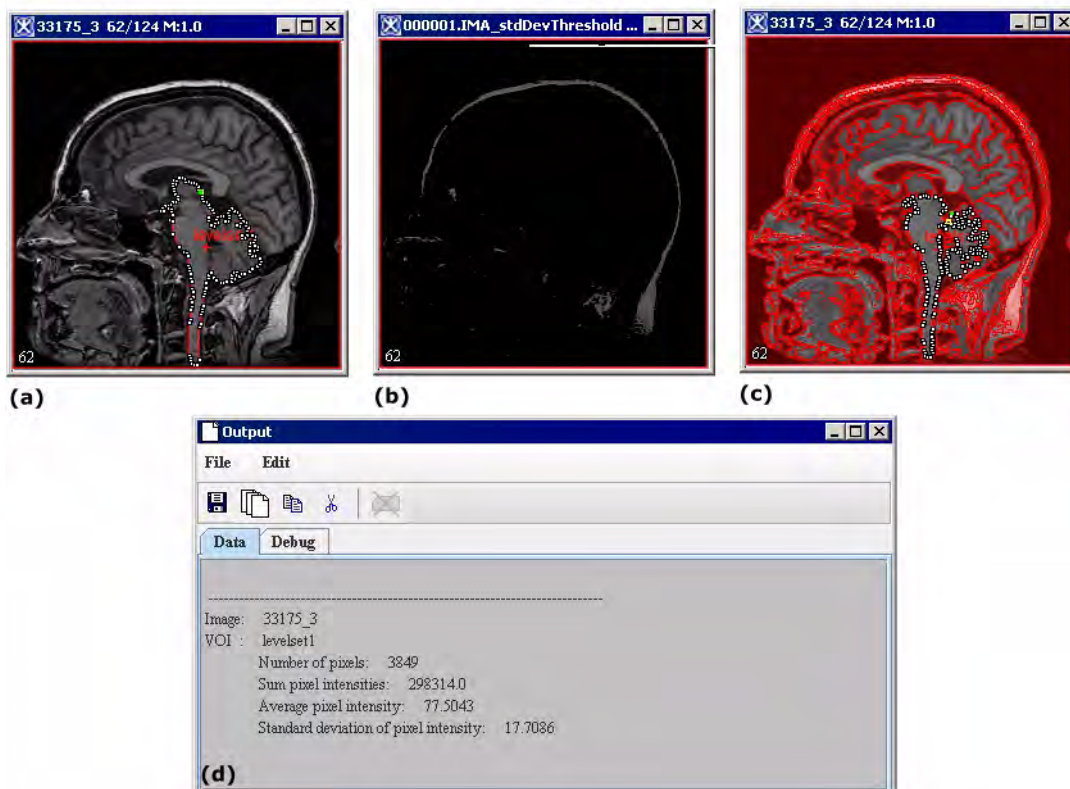


Figure 1. (a): an original image with delineated single VOI, (b): the image thresholded using the default parameters opened in a new image frame, (c) the thresholded image painted over the original image, (d) VOI statistics for the original image

---

## IMAGE TYPES

You can apply this algorithm to all 2D, 3D, and 4D color (RGB) and grayscale images.

---

## APPLYING THE ALGORITHM TO GRAYSCALE IMAGES

---

Note that in order to run Standard Deviation Threshold you should have at least one active VOI on your image.

---

To use this algorithm, do the following:

- 1** Select at least one VOI on your image.
- 2** Launch the tool from Algorithms > Segmentation > Threshold > Threshold Using Standard Deviation.
- 3** The top of the Threshold Using Standard Deviation dialog box will show the statistics for the active VOI(s).
- 4** In the middle Threshold Parameters box enter the user input parameters, such as
  - **Number of standard deviations**, which is used to create the threshold range. Once you've defined the standard deviation for the normal tissue, the range is calculated using the **Number of standard deviations** parameter.
  - **Values outside threshold** are the values that will be assigned to pixels which are outside of the threshold range. The default value is set to zero. This parameter is later used to for calculating an inverse threshold.
  - **Inverse threshold** – this option is checked by default. When the Inverse threshold option is active, the image intensity values **outside** the threshold range are kept, and the ones in the threshold are set to the values outside.
  - **Set min threshold value to image min** – once the intensity range is calculated, you might want to stretch out the range to *image*

*min*. This eliminates the background noise.

- **Set max threshold to image max** – use this option to stretch out the range to *image max*.
- 5 In the Destination Box, specify where you want the thresholded image to appear. You might choose between opening the thresholded image in a new image frame or paint it on the existing image.
  - 6 You can apply the tool to the whole image or only to other VOI(s) that might be delineated in the image.
  - 7 Press OK to run the algorithm.

Results get displayed in the Output window. See also Figure 1 and Figure 2.

## THE THRESHOLD USING STANDARD DEVIATION DIALOG BOX OPTIONS FOR GRAYSCALE IMAGES:

<p><b>Active VOI statistics</b></p>	<p>This box shows statistics for the selected VOI(s) which were used to calculate a standard deviation.</p> <p><b>Number of pixels</b> – a total number of pixels in selected VOI(s);</p> <p><b>Sum pixel intensities</b> – a sum of intensities over all pixels in VOI(S);</p> <p><b>Average pixel intensity</b> equals <i>Sum pixel intensities</i> divided to <i>Number of pixels</i>;</p> <p><b>Standard deviation of pixel intensity</b> is a standard deviation calculated based on the following values Number of pixels, Sum pixel intensities, and Average pixel intensity.</p>	
<p><b>Threshold parameters</b></p>		

Figure 2. Threshold Using Standard Deviation dialog box options for grayscale images

<b>Number of standard deviations</b>	is used to create the threshold range. Once you've defined the standard deviation for the normal tissue, the range is calculated using that parameter.
<b>Values outside threshold</b>	are the values that will be assigned to pixels which are outside of the threshold range. The default value is set to zero.
<b>Inverse threshold</b>	– this option is checked by default. When the Inverse threshold option is active, the image intensity values <b>outside</b> the threshold range are kept, and the ones in the threshold are set to the values outside.
<b>Set min threshold value to image min</b>	– once the intensity range is calculated, use this parameter to stretch out the intensity range to <i>image min</i> . This helps to eliminate the background noise.
<b>Set max threshold to image max</b>	– use this option to stretch out the range to <i>image max</i> .
<b>Destination</b>	In that box, specify where you want the thresholded image to appear. You might choose between opening the thresholded image in a new image frame or paint it on the existing image.
<b>Threshold</b>	Select <b>Whole Image</b> to apply the tool to the whole image; Select <b>VOI(S) regions</b> to apply the tool only to other VOI(s) that might be delineated in the image.
<b>OK</b>	Applies the algorithm according to the specifications made in the dialog box.
<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes this dialog box.
<b>Help</b>	Displays online help for this dialog box.

Figure 2. Threshold Using Standard Deviation dialog box options for grayscale images

## APPLYING THE ALGORITHM TO RGB IMAGES

Note that in order to run Standard Deviation Threshold you should have at least one active VOI on your image.

To use this algorithm, do the following:

- 1** Select at least one VOI on your image.
- 2** Launch the tool from Algorithms > Segmentation > Threshold > Threshold Using Standard Deviation.

- 3** The top of the Threshold Using Standard Deviation dialog box will show the statistics (per channel) for the active VOI(s).
- 4** In the middle Threshold Parameters box enter the user input parameters, such as
  - **Number of standard deviations**, which is used to create the threshold range. Enter the number of st.dev for each channel – R, G, and B channels.
  - **Values outside threshold** are the values that will be assigned to pixels which are outside of the threshold range. The default values for each channel are set to zero. These values are later used to for calculating an inverse threshold.
  - **Inverse threshold** – this option is checked by default. When the Inverse threshold option is active, the image intensity values **outside** the threshold range are kept, and the ones in the threshold are set to the values outside.
  - **Set min threshold value for R, G, and B to image min** – once the intensity range is calculated, you might want to stretch out the intensity range for each channel to *image min*. This eliminates the background noise.
  - **Set max threshold value for R, G, and B to image max** – use this option to stretch out the intensity range for each channel to *image max*.
- 5** In the Destination Box, specify where you want the thresholded image to appear. You might choose between opening the thresholded image in a new image frame or paint it over the existing image.
- 6** You can apply the tool to the whole image or only to other VOI(s) that might be delineated in the image.
- 7** Press OK to run the algorithm.

Results get displayed in the Output window. See also Figure 3 and Figure 4.



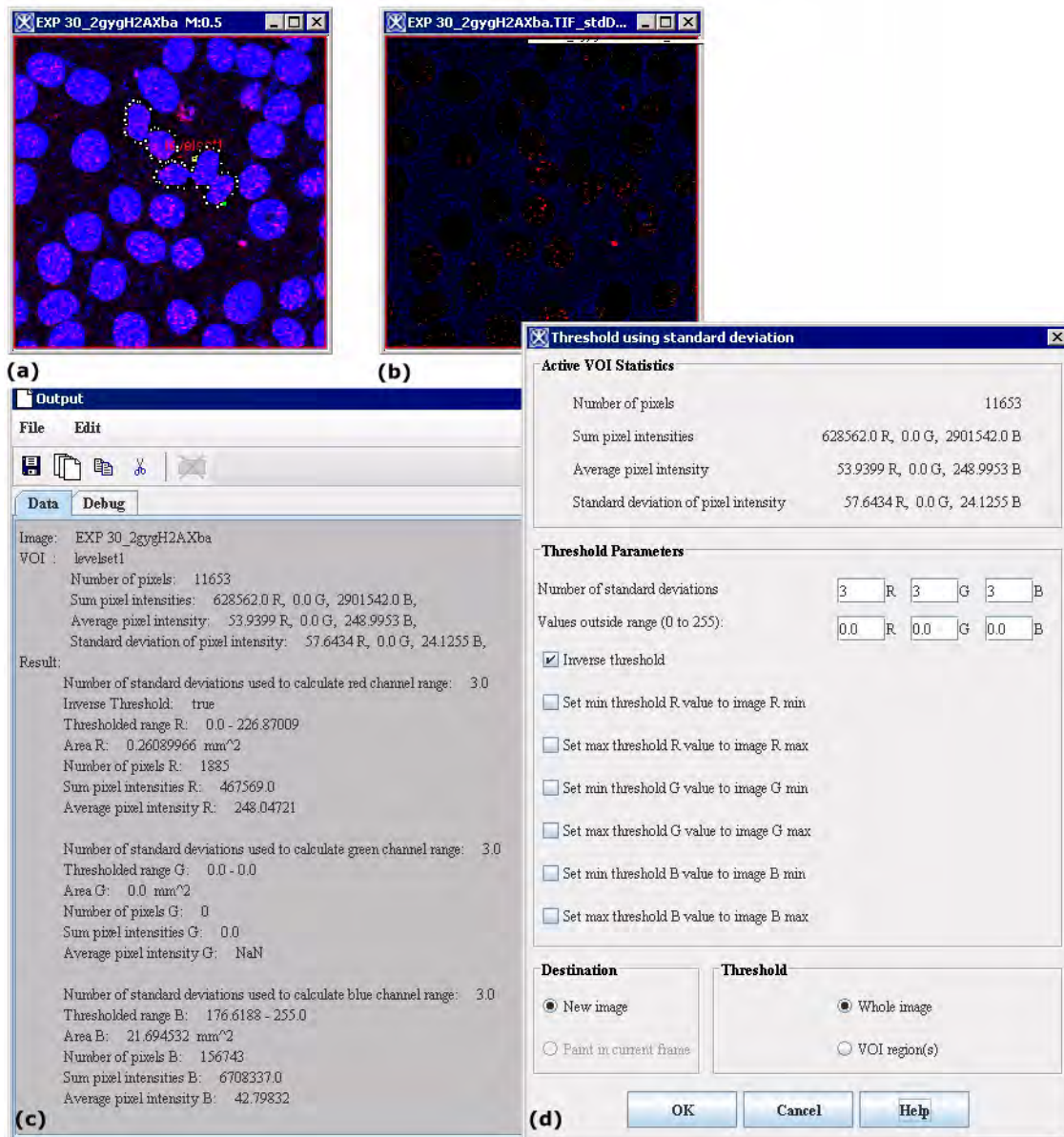


Figure 3. (a): an original RGB image with delineated single VOI, (b): the same image thresholded using the default parameters opened in a new image frame, (c) statistics for the original image appears in the Output MIPAV window, (d) the Threshold Using Standard Deviation dialog box for RGB images with the default settings

## THE THRESHOLD USING STANDARD DEVIATION DIALOG BOX OPTIONS FOR RGB IMAGES:

<p><b>Active VOI statistics</b></p>	<p>This box shows statistics for the selected VOI(s) which were used to calculate a standard deviation.</p> <p><b>Number of pixels</b> – a total number of pixels in selected VOI(s);</p> <p><b>Sum pixel intensities</b> – a sum of intensities over all channels for all pixels in VOI(s);</p> <p><b>Average pixel intensity</b> equals <i>Sum pixel intensities</i> divided to <i>Number of pixels</i>;</p> <p><b>Standard deviation of pixel intensity</b> is a standard deviation calculated based on the following values Number of pixels, Sum pixel intensities, and Average pixel intensity.</p>		
<p><b>Threshold parameters</b></p>			
<p><b>Number of standard deviations per channel (R, G, and B)</b></p>	<p>is used to create the threshold range. Once you've defined the standard deviation for each channel R, G, and B for the normal tissue, the range is calculated using that parameter.</p>		
<p><b>Values outside range (0 to 255)</b></p>	<p>are the values that will be assigned to pixels which are outside of the threshold range. The default value is set to zero.</p>		
<p><b>Inverse threshold</b></p>	<p>– this option is checked by default. When the Inverse threshold option is active, the image intensity values <b>outside</b> the threshold range are kept, and the ones in the threshold are set to the values outside.</p>		
<p><b>Set min threshold value to image min for each channel R, G, and B.</b></p>	<p>– once the intensity range is calculated, use this parameter to stretch out the intensity range for each channel R, G, and B to <i>image min</i>. This helps to eliminate the background noise</p>		

Figure 4. Threshold Using Standard Deviation dialog box options for RGB images

<b>Set max threshold to image max for each channel R, G, and B</b>	– use this option to stretch out the intensity range for each channel to <i>image max</i> .
<b>Destination</b>	In that box, specify where you want the thresholded image to appear. You might choose between opening the thresholded image in a new image frame or paint it on the existing image.
<b>Threshold</b>	Select <b>Whole Image</b> to apply the tool to the whole image; Select <b>VOI(S) regions</b> to apply the tool only to other VOI(s) that might be delineated in the image.
<b>OK</b>	Applies the algorithm according to the specifications made in the dialog box.
<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes this dialog box.
<b>Help</b>	Displays online help for this dialog box.

**Figure 4. Threshold Using Standard Deviation dialog box options for RGB images (continued)**

## Subsampling images

The subsample algorithm in MIPAV allows you to reduce an image in size by a factor of 2, 4, or 8 times. Each pixel of the subsampled image is a Gaussian-weighted average of the original image's 8 neighboring pixels for 2D images or 26 neighboring voxels for 3D images. For example, subsampling a 3D image with the  $x$ ,  $y$ , and  $z$  dimensions of  $256 \times 256 \times 32$ , respectively, by a factor of 2 produces a new image with  $x$ ,  $y$ , and  $z$  dimensions of  $128 \times 128 \times 16$ , respectively.

### To subsample the images

- 1 Open an image (Figure 1).
- 2 Select Algorithms>Transformation tools > Subsample. The Subsample dialog box (Figure 2) opens.



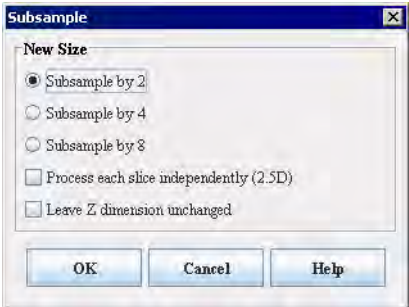
Figure 1. Original image before subsampling

- 3 Select one of the following:

- Subsample by 2
- Subsample by 4
- Subsample by 8
- Select, as an option for 2.5D images only, the Process each slice independently (2.5D) check box.

**Tip:** If you are *not* working with 2.5D images, the Process each slice independently (2.5D) check box does *not* appear in the Subsample dialog box.

**4** Click OK. A status message appears briefly while the program generates the subsampled image in a new image window.

<b>Subsample by 2</b>	Subsamples each image dimension by a factor of 2.	
<b>Subsample by 4</b>	Subsamples each image dimension by a factor of 4.	
<b>Subsample by 8</b>	Subsamples each image dimension by a factor of 8.	
<b>Process each slice independently (2.5D)</b>	Filters each slice of the dataset independently of adjacent slices.	
<b>OK</b>	Applies the parameters that you specified to subsample the image.	
<b>Cancel</b>	Disregards any changes you made in this dialog box, closes the dialog box, and does not subsample the image.	
<b>Help</b>	Displays online help for this dialog box.	

**Figure 2. Subsample dialog box**

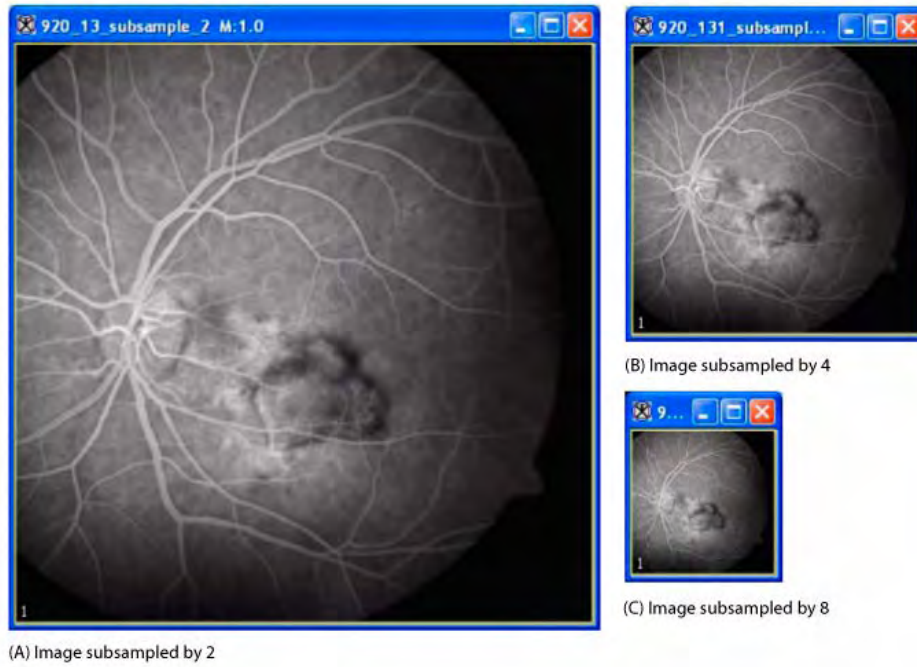


Figure 3. An image subsampled by 2, by 4, and then by 8

---

## Threshold

*The Threshold algorithm replaces the image pixel values with the fill values specified by a user. The pixel values change or remain unchanged depending on whether the original color value is within the threshold range. For RGB or RGBA images, the Threshold command works on each individual RGB channel.*

## Background

---

### IMAGE TYPES

You can apply this algorithm to all 2D, 3D, and 4D color (RGB) and grayscale images.

---

### APPLYING THE THRESHOLD ALGORITHM

To use this algorithm, do the following:

- 1** Select Algorithms > Segmentation > Threshold in the MIPAV window. The Threshold dialog box (Figure 1) appears.
- 2** Complete the information in the dialog box.
- 3** Click OK. The algorithm begins to run, and a progress bar appears momentarily with the status. When the algorithm finishes running, the progress bar disappears, and the results appear in a separate image window.

## THE THRESHOLD DIALOG BOX OPTIONS:

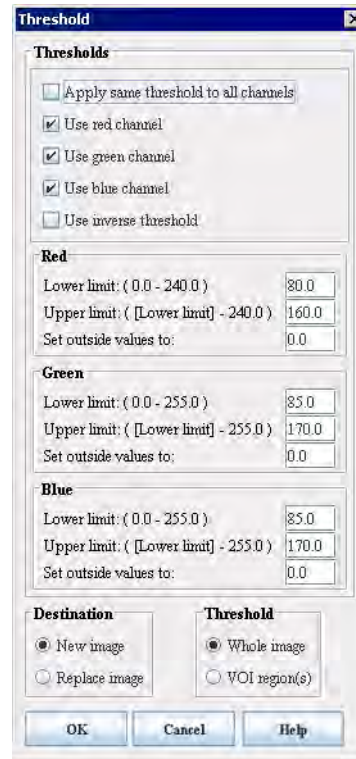


Figure 1. The Threshold dialog box.

### Apply same threshold to all channels

If checked, applies the same threshold values to all RGB channels. In that case only the Red box becomes available for entering the threshold values. Other boxes became unavailable.

### Use <red, green, or blue> channel

Applies the threshold operation to each channel independently. For each pixel, the R, G, and B values will set to 255 if it is in range for that particular channel and to 0 otherwise.

### Use inverse threshold

If checked, changes all pixels which are outside of the specified range. In that case you must specify the desired pixel values into the **Set outside values** text box.

### Red, Green, Blue

For each channel, specify the threshold values in the corresponding boxes – Red, Green or Blue:



**Lower limit** – end of threshold range;

**Upper Limit** – beginning of threshold range;

**Set outside values to:** enter the pixel values for pixels outside the threshold.

**New image** – places the result of the algorithm in a new image window.

**Replace image** – replaces the current active image with the results of the algorithm.

**Whole image** – applies the algorithm to the whole image.

**VOI region(s)**– applies the algorithm to the selected VOI(s).

**OK** – applies the algorithm according to the specifications in this dialog box.

**Cancel**– disregards any changes that you made in the dialog box and closes this dialog box.

**Help** –Displays online help for this dialog box.

---

## Transform

*The Transform/Resample dialog box is a user interface for transforming and (or) resampling images. It offers multiple options that help a user to, first, define the transformation matrix, and then execute the transformation and (or) resampling. The interpolation method can also be specified in the dialog box. The following interpolation techniques are implemented: Nearest-Neighbor, Bilinear, Trilinear, 3rd order B-spline, 4th order B-spline, Cubic Lagrangian, Quintic Lagrangian, Heptic Lagrangian, or Windowed sinc.*

To learn more about interpolation techniques used in the method, refer to Chapter “Interpolation methods used in MIPAV” .

## Background

The dialog box has two tabs – Transform and Resample, see Figure 1. The Transform tab allows the user to set up the transformation matrix and choose the interpolation method. The Resample tab is designed to set up the resampling interpolation.

---

### TRANSFORMING IMAGES

The Transform tab of the dialog box allows the user to transform an image using the transformation matrix that can be either

- A matrix associated with the image,
- A matrix stored in a separate file,
- Or the user defined matrix.

#### The Transform tab options

##### TRANSFORM

**No transformation** – if selected, this option disables all transformation options, but the interpolation options are still available. Therefore, the user can not perform the transformation, but is still able to run the interpolation for a chosen image.

**Use image's associated matrix** – if checked, this option allows using a transformation matrix associated with the image. See Figure 1.

**Read matrix from file** – if checked, reads the transformation matrix from a file. The user should select the file where the transformation matrix is stored.

**Use defined transformation matrix** – if checked, makes available the transformation text boxes, which are used to design the user defined transformation matrix. In these boxes, the user can enter values for translation, rotation, scale, and skew. Refer to Figure 2.

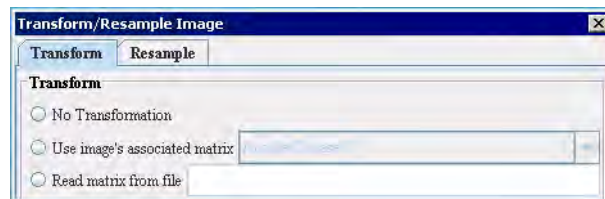


Figure 1. Transformation options

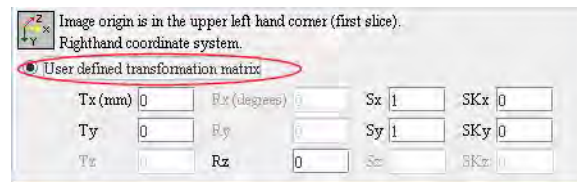


Figure 2. The options available for defining a transformation matrix: Tx, Ty, Tz in mm - translations in the X, Y, and Z directions; Rx, Ry, Rz in degrees - rotations along the X, Y, and Z axis; Sx, Sy, Sz - scale in the X, Y, and Z directions; SKx, SKy, and SKz – skew in the X, Y, and Z directions

#### OPTIONS

**Interpolation** list box is designed to help the user to select the interpolation technique. The list of available interpolation methods includes: Bilinear, Trilinear, 3rd order B-spline, 4th order B-spline, Cubic Lagrangian, Quintic Lagrangian, Heptic Lagrangian, Nearest-Neighbor, and Windowed sinc.

The default interpolation is Bilinear for 2D images and for 3D or 4D images with each slice processed independently. The default interpolation is Trilin-

ear for 3D or 4D images when not using slice processing. To learn more about interpolation techniques used in the method, refer to “Interpolation methods used in MIPAV” .

**Rotate about** radio buttons are designed to specify the image rotation. If Origin is selected, the image will be rotated about the MIPAV origin which is the upper left hand corner of the image. If Center is selected, the image will be rotated about its center. If Use Scanner Center is selected, the image will be rotated about the scanner's center, this choice is available only for DICOM images. See Figure 3.

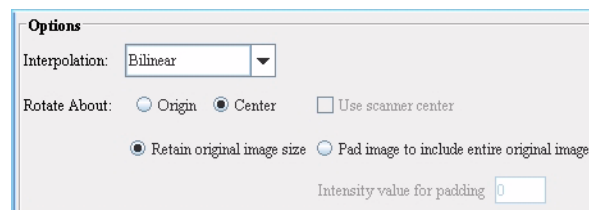


Figure 3. The rotation choices.

**Retain original image size** – if selected, this option preserves the image size. The image will be either padded by pixels with zero intensity (black pixels) to preserve its size, or cropped. See Figure 3 and Figure 4(b) – padded and (c) – cropped.

**Pad image to include entire original image** – if selected, this option pads the image. While using this option the user can also specify the intensity value for padding pixels (this works for 2D images only). See Figure 4(d).

---

In order to include the entire original image, you must select **Pad image....**

---

**Clip output values to input image** – if checked, the option preserves the original image type. That means that, the transformed image minimum value cannot decrease below the original image minimum value, and the transformed image maximum value cannot increase above the original image maximum value. This check box is disabled if Nearest-Neighbor, Bilinear, or Trilinear interpolation is selected, because when using these interpolations, minimum values can not decrease and maximum values can not increase.

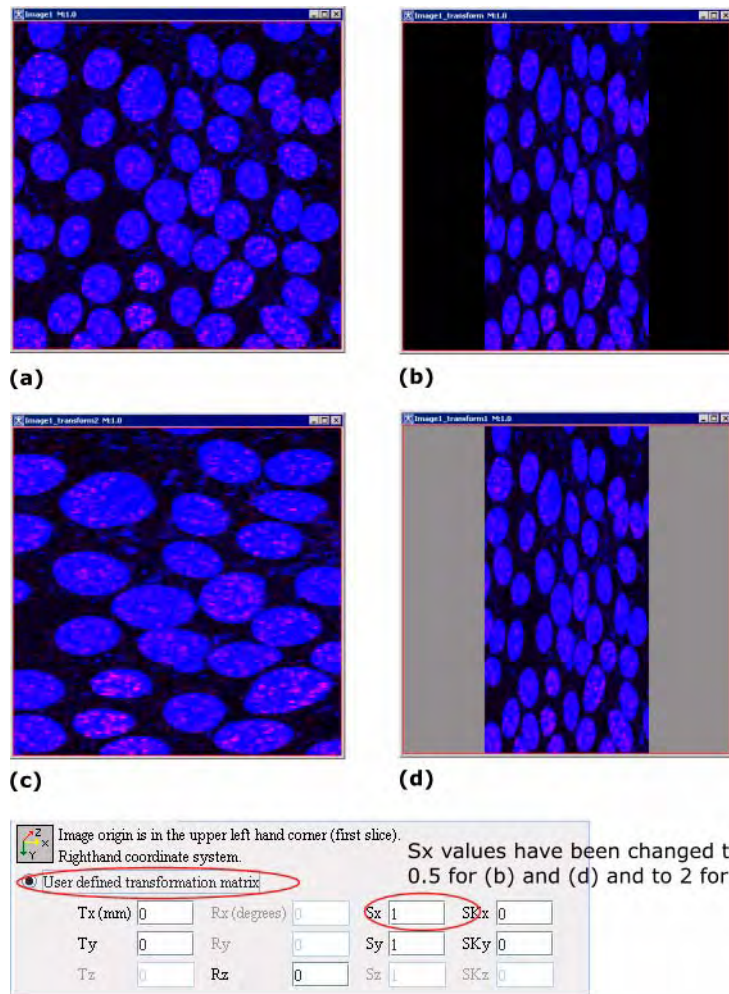


Figure 4. Two examples of using the Retain original image size option for 2D images: (b) with padding and (c) with cropping. An example of using the Pad image option to include entire original image option is (d). (a) – the original 2D image, (b) – the image scaled to 0.5 in X direction, (c) – the image scaled to 2 in X direction, (d) – the image scaled to 0.5 in X direction and padded with pixels with intensities equal to 140

**Process each slice independently (2.5D)** – if checked, this option allows transforming each image slice separately. This works for 3D and 4D images.

**Update origin** – if checked, this option updates the image origin according to the new image size and resolution. By default, this option is activated.

**Transform VOIs** – if checked, the transformation is applied to both the image and to all the VOI(s). Regardless of what interpolation is applied to

the image, Nearest-Neighbor interpolation is always applied to the VOI(s).

**Invert matrix** – if checked, this option inverts the transformation matrix, first, and then applies the transformation with the inverted matrix. This option can only be activated if the **Use image's associated matrix** or **Read matrix from file** option is selected. See Figure 1.

See also “To transform an image:” on page 720.

---

## RESAMPLING IMAGES

The Resample tab of the dialog box allows the user to resample images:

- to the size of a specified image;
- to the user defined size;
- or by the user defined factor.

New pixel values are computed using a weighted combination of existing pixel values within a defined neighborhood of the new pixel location. The interpolation methods used in this implementation of the resampling technique include all available via the Transform tab. If the user did not change the interpolation method in the Transform tab, Trilinear (or Bilinear for 2D images and for 3D and 4D with independent slice processing) interpolation is used.

---

[For more information about interpolation methods used in the Transform/Resample algorithm, refer to “Interpolation methods used in MIPAV” .](#)

---

### The Resample tab options

The Resample tab offers the following options, see Figure 6:

**Resample to size of** is used to select the image which has resolution, dimensions, and units of measurement used to resample the current image.

**Set pixels to preserve FOV** – if selected, preserves the image's field of view (FOV) while resampling the image.

**User defined size** option allows the user to specify both the resolution

and dimension to which the image should be resampled.

**Preserve field of view** – if this option is activated, the algorithm preserves the image's FOV while resampling the image.

**Lock XY aspect ratio** allows the user to preserve the XY aspect ratio.

**Lock XYZ aspect ratio** allows the user to preserve the ratio between Z and the XY plane for 3D and 4D images not using independent slice processing.

**ResX, ResY, ResZ** text boxes are used to specify the new image resolution in each direction.

**DimX, DimY, DimZ** text boxes are used to specify the new image size in each direction.

**Resample by factor** slider is used to choose the resampling factor. The slider allows selecting factor values from 0.25 to 4.0.

See also “To resample an image:” on page 721.

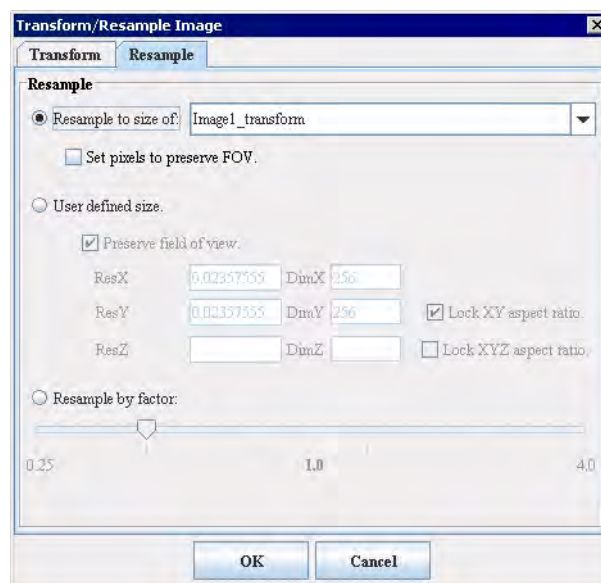


Figure 5. The Resample tab

---

## IMAGE TYPES

The Transform/Resample algorithm works on 2D, 3D and 4D color and grayscale images.

---

## REFERENCES

Refer to the following references for more information about the Transform algorithm:

Paul Bourke, Interpolation methods, <http://local.wasp.uwa.edu.au/~pbourke/other/interpolation/index.html> - Trilinear interpolation.

B-splines: <http://www.cl.cam.ac.uk/teaching/1999/AGraphHCI/SMAG/node4.html>

Chapter 3: “Registration Methodology: Concepts and Algorithms” by Derek L.G.Hill and Philippe Batchelor in Medical Image Registration edited by Joseph V. Hajnal, Derek L.G.Hill, and David J. Hawkes, CRC Press, 2001, pp. 40-70.

Chapter 33 “Within-Modality Registration Using Intensity-Based Cost Functions” by Roger P. Woods in Handbook of Medical Image Processing and Analysis, Editor, Isaac N. Bankman, Academic Press, 2000, pp. 529-553.

Thomas M. Lehmann, \* Member, IEEE, Claudia Gonner, and Klaus Spitzer Survey: Interpolation Methods in Medical Image Processing. IEEE Transactions on Medical Imaging, Vol. 18, No. 11, November 1999, pp. 1049-1075.

---

## OUTPUT

The transformed and (or) resampled image is displayed, see Figure 4 and Figure 6.

## Applying the Transform/Resample algorithm

To transform an image:

- 1** Open an image of interest.
- 2** Call Algorithms>Transformation tools >Transform.
- 3** In the Transform/Resample dialog box, open the Transform tab and do the following:



- To specify a transformation matrix, select one of the following options: No Transformation, Image Associated Matrix, Read Matrix from File, or User Defined Matrix.
- Select the interpolation method that will be used to create a transformed image.
- Specify the image origin.
- Specify whether to preserve the image size or not.
- Specify whether to clip the output image if the intensity values are outside the original image range.
- Indicate if you wish to apply the transformation to all image's VOI(s) in addition to the image.
- Other options.

**4** Click OK. The algorithm begins to run, and a progress bar appears momentarily with the status. When the algorithm finishes running, the progress bar disappears, and the result appears in a separate image window.

The transformation algorithm begins to run. When the algorithm finishes running, the transformed image appears in a new image frame. See also Section “The Transform tab options” on page 714.

### To resample an image:

- 1** Open an image of interest.
- 2** Call Algorithms>Transformation tools >Transform.
- 3** In the Transform/Resample dialog box, navigate to the Resample tab.
- 4** Use the Resample tab options to specify the size of the new resampled image.
  - You can choose to resample the original image to the size of some other image, or you can choose to specify the new image size, or you can use the Resample Factor slider.
  - When selecting the user defined size, you might also choose to preserve a field of view. You can lock the XY aspect ratio for 2D images and for 3D and 4D images using independent slice processing. You can lock the XYZ aspect ratio for 3D and 4D images not using independent slice processing.
  - The interpolation method that will be used to create a resampled image must be specified in the Transform tab, refer to “The Transform tab options” on page 714.

- 5 Press OK. The algorithm begins to run. When the resampling algorithm finishes running, the result image appears in a new image frame. See Figure 6. See also Section 1.2.1 “The Resample tab options.”

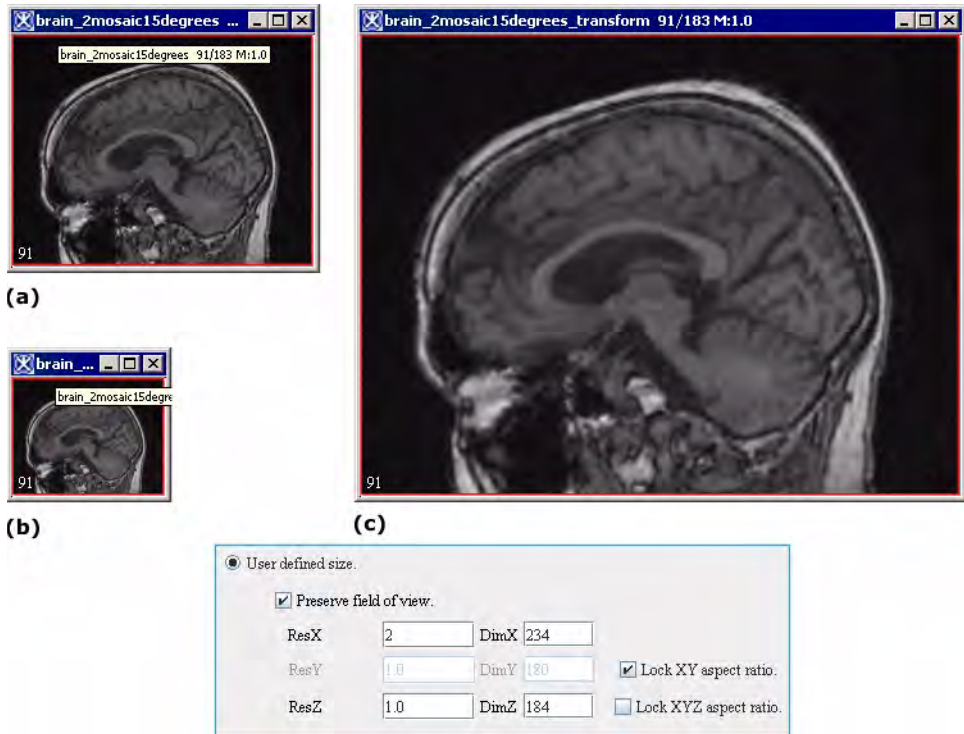


Figure 6. Resampling images: (a) the original image, (b) the image resampled by 0.5 in X direction while the XY aspect ratio has been preserved, (c) the image resampled by 2 in X direction while the XY aspect ratio has been preserved

## Transform: Conformal Mapping Algorithms

*The methods described in this document use conformal mapping to transform points in a circular sector, circle, ellipse, or nearly circular region to points in a circle or rectangle.*

### Background

A conformal mapping, is a transformation  $w=f(z)$  that preserves local angles. An analytic function is conformal at any point where it has a nonzero first derivative. A complex function is analytic on a region R if it is complex differentiable at every point in R. In a conformal mapping, in any small neighborhood the relative angle and shape are preserved.

### CIRCULAR SECTOR TO RECTANGLE

The algorithm uses a 2D conformal mapping to convert a circular sector, defined by four user points at the sector corners, to a rectangle of the user specified size. The circular sector has an inner radius  $r_{min}$ , an outer radius  $r_{max}$ , and extends over an angle  $\theta = \alpha * \pi$  radians, with  $0 < \alpha \leq 1$ .

Let,  $z_1$  and  $z_2$  belong to  $r_{max}$ , and  $z_3$  and  $z_4$  belong to  $r_{min}$  as shown in Figure 1. The mapping to the user specified rectangle is performed in the following steps (1–6):

- 1** The maximum radius can be facing to the top, bottom, right, or left. So the algorithm, first, determines which of 4 orientations is present.
- 2** Calculating the center of the circular segment. The center is calculated as a point  $(x_c, y_c)$  where two lines  $(z_3, z_2)$  and  $(z_4, z_1)$  cross. Therefore,

EQUATION 1

$$x_c = \frac{(x_3 * y_2) - (x_2 * y_3)}{(x_3 - x_2)} - \frac{(x_4 * y_1) - (x_1 * y_4)}{(x_4 - x_1)}$$

$$y_c = x_c \frac{y_4 - y_1}{x_4 - x_1} + \frac{(x_4 * y_1) - (x_1 * y_4)}{(x_4 - x_1)}$$

**3** Calculating the angle of the sector in radians, see Figure 2.

The angle is calculated as follows  $theta = theta_2 - theta_1$ .

For the maximum radius facing to the left, we must take into account the discontinuity of the angle at the negative x axis. The angle changes from  $-Pi$  to  $Pi$ , so the equation for  $theta$  is now:

$$theta = abs(theta_1) + abs(theta_2) - 2.0 * Pi$$

The angle  $theta_1$  is along the line from center to  $z_4$  to  $z_1$  in  $-Pi$  to  $Pi$  radians:

$$tan(theta_1) = ((y_1 - y_4)/(x_1 - x_4)) \Rightarrow theta_1 = arctan((y_1 - y_4)/(x_1 - x_4))$$

the angle  $theta_2$  is along the line from center to  $z_3$  to  $z_2$  in  $-Pi$  to  $Pi$  radians:

$$tan(theta_2) = ((y_2 - y_3)/(x_2 - x_3)) \Rightarrow theta_2 = arctan((y_2 - y_3)/(x_2 - x_3))$$

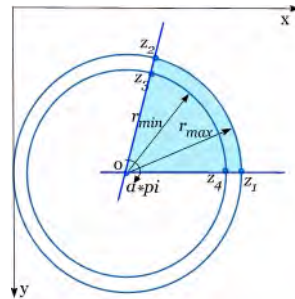


Figure 1. Calculating the center of the circle to which the circular segment belongs

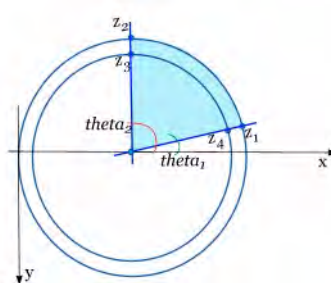


Figure 2. Calculating the angle of the circular segment

**4** For the conformal mapping, it requires that  $r_{max} > 1$  and  $0 < r_{min} < 1$ . Refer to “A Domain Decomposition Method for Conformal Mapping onto a Rectangle” Remark 4.7 (see “References” on page 733). The transformation theory is based on a mathematical crosscut along the unit circle.

Therefore, the algorithm divides all the distances from the circular center by  $\sqrt{r_{max} * r_{min}}$ . It produces a value of r so that

$$\sqrt{r_{min} * r_{max}} < r < \sqrt{r_{max} * r_{min}}$$

**5** To convert a circular sector to a rectangle, the algorithm uses the following transformation function

EQUATION 2

$$w = f(z) = u + iv = \log(z)/(i * \alpha * \pi)$$

Let  $z = r * \exp(i * \theta)$

Then,

EQUATION 3

$$\begin{aligned} w &= \log(r * \exp(i * \theta)) / (i * \alpha * \pi) \Rightarrow \\ w &= \theta / (\alpha * \pi) - i * \log(r) / (\alpha * \pi) \\ u &= \theta / (\alpha * \pi), \\ v &= -\log(r) / (\alpha * \pi) \end{aligned}$$

Since conformal mapping requires that  $f(z)$  must be analytic on all points, the Cauchy-Riemann equations must be satisfied and  $f'(z)$  must not equal zero.

### Satisfying the Cauchy-Riemann equations

Suppose that  $f(z) = f(x + iy) = u(x, y) + iv(x, y)$  is differentiable at the point  $z_0 = x_0 + iy_0$ . Then, the partial derivatives of u and v exist at the point  $(x_0, y_0)$ , and

EQUATION 4

$$\begin{aligned} f'_x(z_0) &= \lim_{\Delta x \rightarrow 0} \frac{\Delta z}{\Delta x} \\ f'_x(z_0) &= \lim_{\Delta x \rightarrow 0} \left( \frac{\Delta u_x(x_0, y_0) + i \Delta v_x(x_0, y_0)}{\Delta x} \right) = \lim_{\Delta x \rightarrow 0} \left( \frac{u_x(x_0, y_0)}{\Delta x} + i \frac{v_x(x_0, y_0)}{\Delta x} \right) = \frac{du}{dx} + i \frac{dv}{dx} \\ f'_y(z_0) &= \lim_{\Delta y \rightarrow 0} \frac{\Delta z}{\Delta y} \\ f'_y(z_0) &= \lim_{\Delta y \rightarrow 0} \left( \frac{\Delta u_y(x_0, y_0) + i \Delta v_y(x_0, y_0)}{i \Delta y} \right) = \lim_{\Delta y \rightarrow 0} \left( \frac{\Delta v_y(x_0, y_0)}{\Delta y} - i \frac{\Delta u_y(x_0, y_0)}{\Delta y} \right) = \frac{dv}{dy} - i \frac{du}{dy} \end{aligned}$$

Equating the real and imaginary parts gives

EQUATION 5

$$u_x(x_0, y_0) = v_y(x_0, y_0) \text{ and } u_y(x_0, y_0) = -v_x(x_0, y_0)$$

The Cauchy-Riemann equations in polar form are:

EQUATION 6

$$\begin{aligned} du/dr &= (1/r) (dv/d\theta) \\ -dv/dr &= (1/r) (du/d\theta) \end{aligned}$$

The first equation gives zero on both sides and the second equation gives  $(\alpha * \pi)/r$  on both sides, so the Cauchy-Riemann equations hold.

**Satisfying the non zero first derivative**

EQUATION 7

$f(z) = 1/(i * \alpha * \pi * z)$ ,  $f(z) \neq 0$  except at infinity.

- 6** Now, the algorithm performs a conformal mapping from the circular segment to a rectangle of width 1 and height  $\log(r_{max}/r_{min})/\theta$ . Values for  $f(z_1)$ ,  $f(z_2)$ ,  $f(z_3)$ , and  $f(z_4)$  are as follows:

EQUATION 8

$$\begin{aligned} z'_4 : r_{min} &\Rightarrow \frac{r_{min}}{\sqrt{r_{max} * r_{min}}} \\ f(z'_4) &= \frac{\log(\sqrt{r_{min} * r_{max}})}{i * \alpha * \pi} = \frac{-i * \log(\sqrt{r_{max}/r_{min}})}{\alpha * \pi} \end{aligned}$$

EQUATION 9

$$z'_3 = \sqrt{r_{min} * r_{max}} * \exp(i * \alpha * \pi)$$

$$f(z'_3) = 1 + i * \frac{\log(\sqrt{r_{max} * r_{min}})}{\alpha * \pi}$$

EQUATION 10

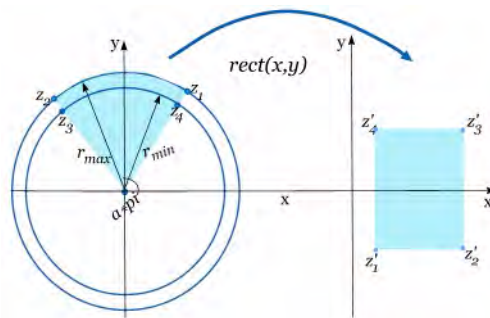
$$z'_1 = \sqrt{r_{max}/r_{min}}$$

$$f(z'_1) = \frac{\log(\sqrt{r_{max}/r_{min}})}{i * \alpha * \pi} = -i * \frac{\log(\sqrt{r_{max}/r_{min}})}{\alpha * \pi}$$

EQUATION 11

$$z'_2 = \sqrt{r_{max}/r_{min}} * \exp(i * \alpha * \pi)$$

$$f(z'_2) = 1 - i * \frac{\log(\sqrt{r_{max}/r_{min}})}{\alpha * \pi}$$



**Figure 3.** The region inside a circular sector is mapped to a rectangle:  $z_1$  is the upper right point on  $r_{max}$ ,  $z_2$  is the upper left point on  $r_{max}$ ,  $z_3$  is the lower left point on  $r_{min}$ , and  $z_4$  is the lower right point on  $r_{min}$

**7** Then, the rectangle from step 6 is linearly scaled to a rectangle of user specified  $x_{Dim}$  and  $y_{Dim}$ . Note that the linear scaling of one rectangle to another is not a conformal mapping unless the width and height are scaled by the same factor. When the maximum radius faces to the top, this gives us the following values for  $z'_1$ ,  $z'_2$ ,  $z'_3$  and  $z'_4$ :

$$z'_1 = (x_{Dim-1}, 0),$$

$$z'_2 = (0, 0),$$

$$z'_3 = (0, y_{Dim-1}),$$

$$z'_4 = (x_{Dim-1}, y_{Dim-1})$$

## TRANSFORMATION: CIRCLE TO RECTANGLE

The algorithm uses a 2D conformal mapping to convert a circle to a rectangle of the user specified size. The circle has the radius  $r$ .

The user inputs 2 point VOIs. The first point is located at the center of the circle. The second point can be any point located on the actual circle. The user also inputs the X and Y dimensions (xDimDest and yDimDest) of the output image. See Figure 4. The algorithm performs three mappings to go from the circular area specified by the user to an output rectangle.

### Mapping 1

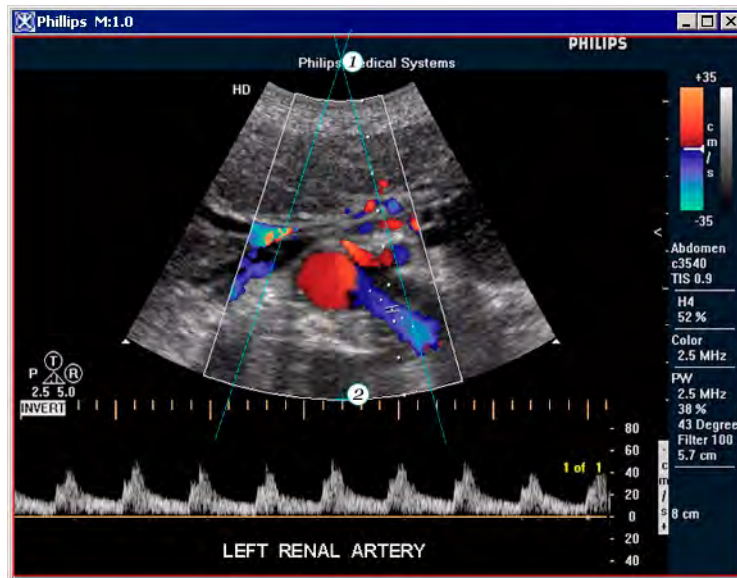


Figure 4. The user inputs 2 point VOIs. The first point is located at the center of the circle. The second point can be any point located on the actual circle

It maps from the destination rectangle to a rectangle centered on the origin. This is the mapping from a rectangle with the edge points  $(0,0)$ ,  $(xDimDest-1, 0)$ ,  $(xDimDest-1, yDimDest-1)$ ,  $(0, yDimDest-1)$  to a rectangle with the edge points  $(-K,Kp)$ ,  $(K,Kp)$ ,  $(K,-Kp)$ ,  $(-K,-Kp)$ .

Where  $K$  is the complete elliptic integral of the first kind using the modulus and  $Kp$  is the complete elliptic integral of the first kind using the



complementary modulus:

$$\text{complementary modulus} = \sqrt{1 - m^2}$$

The modulus  $m$  belongs to the interval  $[0,1]$  and is arbitrarily chosen as 0.5.

### Mapping 2

This is a conformal mapping from the origin centered rectangle to the unit circle using:

EQUATION 12

$$w = \sqrt{\frac{1 - cn(z, m)}{1 + cn(z, m)}}$$

where  $z$  is the input complex number,  $w$  is the output complex number, and  $cn(z, modulus)$  is the Jacobian Elliptic function of type  $cn$ .

### Mapping 3

The algorithm scales and translates from the unit circle to the user selected circle. It uses bilinear interpolation to find the contributions from the four nearest neighbors in the original user selected circle.

---

## TRANSFORMATION: ELLIPSE TO CIRCLE

This algorithm performs the conformal mapping of an ellipse to a circle with the user defined radius.

Let's consider the original ellipse as having a tilt  $theta$  with the X axis. Then the conformal mapping from an ellipse to a circle is done in three following steps.

**In the first step**, the algorithm translates the circle from the center of the destination square image to the origin, rotates the circle by the angle  $-theta$  and converts it to a unit disc by dividing the distance from the center by the radius.

The algorithm places the center of the circle on the center of the destination image. The algorithm considers only the points located on the destination circle, e.g.  $(j - y_c)^2 + (i - x_c)^2 \leq \text{radSq}$ .

Then, it translates the circle to the image origin, so that

EQUATION 13

$$\begin{aligned}x_p &= i - x_c \\y_p &= j - y_c\end{aligned}$$

After that, it rotates around the circle by the angle  $-\theta$

EQUATION 14

$$\begin{aligned}x_{rot} &= x_p \cdot \cos(\theta) + y_p \cdot \sin(\theta) \\y_{rot} &= -x_p \cdot \sin(\theta) + y_p \cdot \cos(\theta)\end{aligned}$$

And the following step scales the circle to a unit disc:

EQUATION 15

$$\begin{aligned}x_{rot} &= x_{rot} / \text{radius} \\y_{rot} &= y_{rot} / \text{radius}\end{aligned}$$

**In the second step**, it maps from the unit disc to a standard ellipse on the X axis with foci at (-1,0) (1,0) and with the same major axis/minor axis ratio as the original ellipse.

Let  $2a$  = major axis,  $2b$  = minor axis of the original ellipse.

For an ellipse with its center at the origin and foci on the X axis:

EQUATION 16

$$\left[ \frac{x^2}{a^2} \right] + \left[ \frac{y^2}{b^2} \right] = 1$$

The standard ellipse used here has foci at (-1,0) (1,0) and is of the form:

EQUATION 17

$$\left( \frac{x^2}{\cosh^2(x)} \right) + \left( \frac{y^2}{\sinh^2(x)} \right) = 1$$

To scale from the original ellipse to the standard ellipse, the algorithm uses the following transform:

EQUATION 18

$$\tanh(x_i) = \sinh(x_i) / \cosh(x_i) = b/a = \text{minor axis} / \text{major axis} = \text{axisRatio}$$

$$x_i = \arg \tanh(\text{axisRatio})$$

$$x_i = 0.5 * \log((1 + \text{axisRatio}) / (1 - \text{axisRatio}))$$

To conformal map from the unit disc to the standard ellipse, the algorithm uses the following transform T1:

EQUATION 19

$$T_1 = \sin \left[ \frac{\pi}{2 * K(s)} * F\left(\frac{z}{\sqrt{s}}, s\right) \right]$$

Where, K(s) is the complete elliptic integral of the first kind and F is the incomplete or normal elliptic integral of the first kind such that:

EQUATION 20

$$F(z, s) = \int_0^z \frac{dx}{\sqrt{(1-x^2)(1-s^2*x^2)}}$$

Because the elliptical integral functions in MIPAV all use the form:

EQUATION 21

$$F(k, \phi) = \int_0^\phi \frac{d\theta}{\sqrt{(1-k^2 \sin^2(\theta))}}$$

So x in the elliptical integral form used in Equation 20 equals sin(.) in the elliptical integral form used in MIPAV, or  $\phi = \arcsin(x)$ :

For complex z

EQUATION 22

$$\arcsin(z) = -i * \log(i * z + \sqrt{(1-z^2)})$$

Now how to find s:

$$u(s) = 2xi, 0 \leq s \leq 1; u(s) = \frac{\pi}{2} * \frac{K(\sqrt{1-s^2})}{K(s)}$$

Where u(s) decreases from infinity to zero as s moves from zero to one.

**In the third step**, the algorithm maps from the standard ellipse to the original ellipse by rotating, scaling, and translating. It uses bilinear interpolation to find the contributions from the four nearest neighbors in the original ellipse space.

## TRANSFORMATION: NEARLY CIRCULAR REGION TO CIRCLE

The algorithm uses 2D conformal mapping to convert a not quite circular region, defined by the user defined VOI, to a circle of the user specified size.

The conformal mapping is done in three steps:

- 1** It translates the circle from the center of the square image to the origin, and then converts it to a unit disc by dividing the distance from the center by the radius.
- 2** It maps from the unit circle to a unit nearly circular region at the origin.
- 3** It maps from the unit nearly circular region at the origin to the original nearly circular region by scaling and translating.

According to the Nehari book (see "References" on page 733) the function F(z):

$$F(z) = z + \frac{\xi z}{2\pi} \int_0^{2\pi} \frac{\exp(i * \theta + z)}{\exp(i * \theta - z)} * p(\theta) d\theta$$

maps  $|z| < 1$  onto the nearly circular domain which boundary has the polar equation  $r = 1 + \xi * p(\theta)$  where  $p(\theta)$  is bounded and piecewise continuous and  $\xi$  is a small positive parameter. For more information, refer to *Conformal*

*Mapping* by Zeev Nehari, Dover Publications, Inc., 1952 see “References” .

---

## REFERENCES

“2D-Shape Analysis Using Conformal Mapping” by E. Sharon and D. Mumford, International Journal of Computer Vision, Volume 70, Issue 1 (October 2006) pp: 55 - 75, 2006, ISSN: 0920-5691.

A Domain Decomposition Method for Conformal Mapping onto a Rectangle”, N. Papamichael and N. S. Stylianopoulos, Constructive Approximation, Vol. 7, 1991, pp. 349-379. Relevant result is given in Remark 4.7 on pages 374-375.

*Advanced Calculus for Applications*, Second Edition by F. B. Hildebrand, Section 10.4 “Analytic Functions of a Complex Variable” pages 550-554 and Section 11.4 Conformal Mapping pages 628-632, Prentice-Hall, Inc., 1976.

*Applied and Computational Complex Analysis*, Volume I by Peter Henrici, Chapter “On Schwarz-Christoffel Mapping Function”, pp. 411-412., John Wiley and Sons, Inc., 1974.

*Conformal Mapping* by Zeev Nehari, Dover Publications, Inc., 1952, Chapter V, “Conformal Mapping of Simply Connected Domains”, Section 11, “Conformal Mapping of Nearly Circular Domains”, pp. 263 - 265 and Chapter VI, “Mapping Properties of Special Functions”, Section 3, “Elliptic Functions”, pp. 280-299.

“On Conformal Representations of the Interior of an Ellipse” by Stanislaw Kanas and Toshiyuki Sugawa, Annales Academiae Scientiarum Fennicae Mathematica, Volume 31, 2006, pp. 329-348.

“On the Conformal Mapping of Nearly-Circular Domains” by Zeev Nehari and Vikramaditya Singh, Proceedings of the American Mathematical Society, 1956, pp. 370-378.

Jacobi's elliptic functions, refer to [http://en.wikipedia.org/wiki/Jacobian\\_elliptic\\_functions](http://en.wikipedia.org/wiki/Jacobian_elliptic_functions)

Complete Elliptic Integral of the First Kind, refer to <http://mathworld.wolfram.com/CompleteEllipticIntegraloftheFirstKind.html>

<http://mathworld.com/InverseSine.html>

<http://mathworld.wolfram.com/LemniscateFunction.html>

## Applying the Circular Sector to Rectangle algorithm

---

### IMAGE TYPES

The algorithms can be applied to all ultrasound images that can be opened in MIPAV.

## RUNNING THE ALGORITHM

To run the Circular Sector to Rectangle algorithm, do the following:

- 1** Open an image of interest. The image must be a 2D color (RGB) or grayscale image.
- 2** Call Algorithms > Transformation > Circular Sector to Rectangle.
- 3** Put four points (use the Point VOI tool) at the edges of the ultrasound image.
- 4** Make sure that points 1 and 2 are located at the edges of the *max* radius boundary and points 3 and 4 located at the edges of the *min* radius boundary.
- 5** Specify the X and Y output dimensions. Press OK.
- 6** The algorithm begins to run and the output image appears in a new image frame. See Figure 5.

To verify that the circle center and points  $z_3$ , and  $z_2$  are collinear and also the circle center and  $z_4$ , and  $z_1$  are collinear, use the MIPAV line drawing tool, see Figure 5.

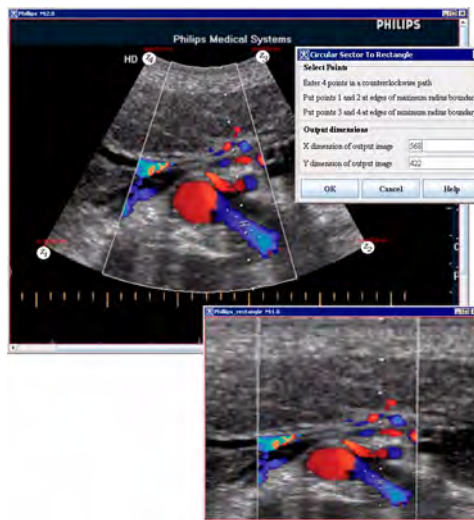


Figure 5. Applying the Circular Segment to Rectangle algorithm

## Applying the Circle to Rectangle algorithm

### IMAGE TYPES

The Ellipse to Circle algorithm works on 2D color and black and white images.

### RUNNING THE ALGORITHM

To run the Circle to Rectangle to Rectangle algorithm, do the following:

- 1** Open an image of interest. The image must be a 2D color (RGB) or grayscale image.
- 2** Call Algorithms > Transformation > Circle to Rectangle.
- 3** Put two point VOIs on the image the first point must be located at the center of the circle and the second point can be any point located on the actual circle. See Figure 6.
- 4** Specify the X and Y output dimensions. Press OK.

The algorithm begins to run and the output image appears in a new image frame.

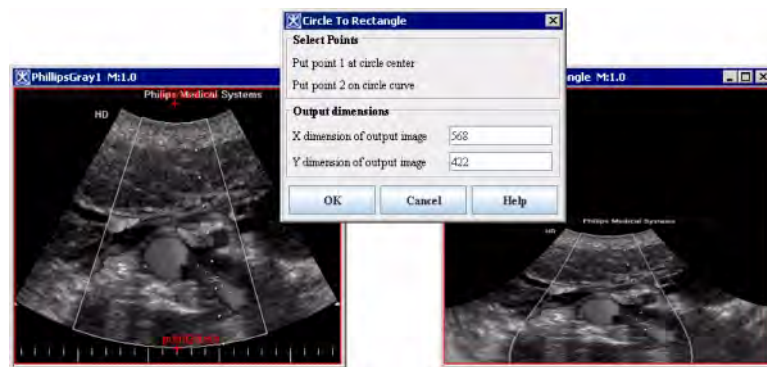


Figure 6. Applying the Circle to Rectangle algorithm

## Applying the Ellipse to Circle algorithm

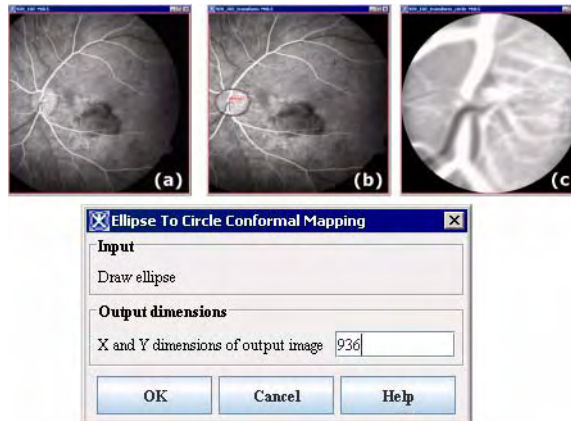
### IMAGE TYPES

The Ellipse to Circle algorithm works on 2D color and black and white images.

### RUNNING THE ALGORITHM

To run the Ellipse to Circle algorithm, do the following:

- 1** Open an image of interest. The image must be a 2D color or black and white image.
- 2** Call Algorithms > Transformation > Ellipse to Circle.
- 3** Delineate an elliptical VOI on the image. See Figure 7.
- 4** In the Ellipse to Circle Conformal Mapping dialog box, specify the X and Y output dimensions. Note that because the output image must be a square image you need to specify only one dimension. Press OK.



**Figure 7. Applying the Ellipse to Circle algorithm: (a)-the eye image, (b)-the distorted eye image with an elliptical VOI, (c)-the part of the image under the VOI is transformed to a circle using the dialog box settings**

The algorithm begins to run and the output image appears in a new image frame. See Figure 7.



## Applying the Nearly Circle to Circle algorithm

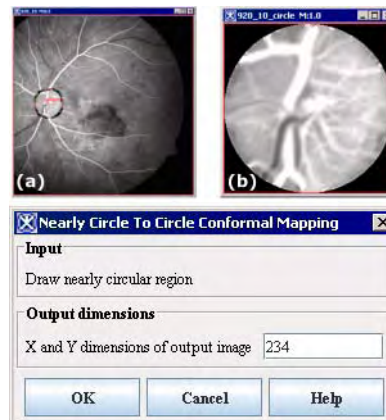
### IMAGE TYPES

The Nearly Circle to Circle algorithm works on 2D color and black and white images.

### RUNNING THE ALGORITHM

To run the Nearly Circle to Circle algorithm, do the following:

- 1 Open an image of interest.
- 2 Call Algorithms > Transformation > Nearly Circle to Circle.
- 3 Delineate a VOI on the image. See Figure 8.
- 4 In the Nearly Circle to Circle Conformal Mapping dialog box, specify the output dimension. Note that because the output image must be a square image, you need to specify only one dimension. Press OK.



**Figure 8. Applying the Nearly Circle to Circle algorithm: (a)-the eye image with a nearly circular VOI, (b)-the part of the image under the VOI is transformed to a circle using the dialog box settings**

The algorithm begins to run and the output image appears in a new image frame.

---

## Transform Nonlinear

*The Transform Nonlinear (TransformBSpline) algorithm takes a source image and uses information read in from a nonlinear transformation (.nlt) file to perform a nonlinear B-Spline transformation on the image. For the B-Spline transformation, the dimensionality of the transformed image, the degree of the B-Spline (ranging from 1 to 4), the number of control points, and the values of the control points are obtained from the .nlt file. See Figure 2.*

The .nlt file is an output file that appears after applying the “B-Spline Automatic Registration” algorithm to images. It can be found in the directory where your image of interest is located. The name of the .nlt file is the same as the name of the image file.

---

**Note:** make sure that the dimensions of the source image used in the .nlt file are the same as the dimensions of the source image used in the Transform Nonlinear algorithm.

---

## Background

Refer to MIPAV Uses Guide Volume 2, Algorithms, “B-Spline Automatic Registration” .

---

### IMAGE TYPES

You can apply this algorithm to color (RGB) and grayscale 2D and 3D images.

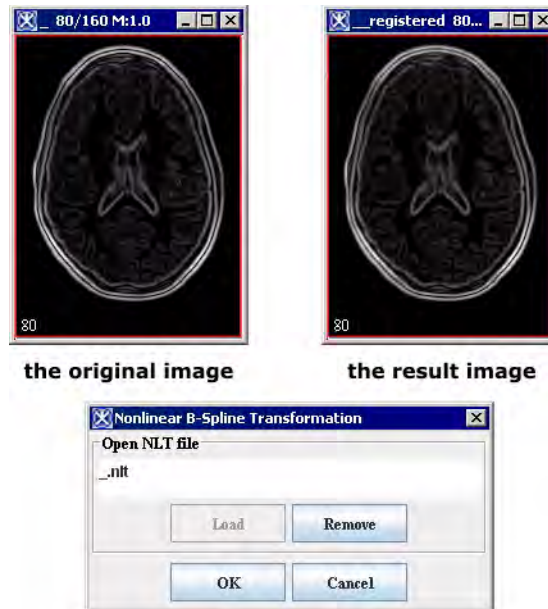
## Applying the Transform Nonlinear algorithm

**To run the algorithm, complete the following steps:**

- 1** Open an image of interest;
- 2** Select Algorithms > Transformation tools > Transform Nonlinear;
- 3** The Nonlinear B-Spline Transformation dialog box appears;

- 4** Use the Load button to select and open an appropriate .nlt file;
- 5** Press OK.

The algorithm begins to run, and a progress bar appears with the status. When the algorithm finishes running, the transformed image appears in a new image frame.



**Figure 1. Applying the Transform Nonlinear algorithm**

```
# B-spline registration of _ to _1
# Number of image dimensions
3
# Target image resolutions
1.0 1.0 1.0
# Target image dimensions
176 236 161
# B-Spline Degree (same for all axes)
1 1 1
# B-Spline Control Points (same for all axes)
8 8 8
# Final values of control points
# iControlX = 0 iControlY = 0 iControlZ = 0
0.0 0.0 0.0
# iControlX = 0 iControlY = 0 iControlZ = 1
0.0 0.0 0.14285715
# iControlX = 0 iControlY = 0 iControlZ = 2
0.0 0.0 0.2857143
# iControlX = 0 iControlY = 0 iControlZ = 3
0.0 0.0 0.42857143
# iControlX = 0 iControlY = 0 iControlZ = 4
0.0 0.0 0.5714286
# iControlX = 0 iControlY = 0 iControlZ = 5
0.0 0.0 0.71428573
# iControlX = 0 iControlY = 0 iControlZ = 6
0.0 0.0 0.85714287
# iControlX = 0 iControlY = 0 iControlZ = 7
0.0 0.0 1.0
# iControlX = 0 iControlY = 1 iControlZ = 0
0.0 0.14285715 0.0
# iControlX = 0 iControlY = 1 iControlZ = 1
0.0 0.14285715 0.14285715
# iControlX = 0 iControlY = 1 iControlZ = 2
0.0 0.14285715 0.2857143
# iControlX = 0 iControlY = 1 iControlZ = 3
```

Figure 2. A sample .nlt file

---

## Transform to power of 2

*The Transform to power of 2 algorithm resamples the original image to dimensions that are powers of 2.*

### Background

By default, this algorithm will use Bilinear Interpolation for 2D images or Trilinear Interpolation for 3D images to resample the original image to dimensions that are powers of 2.

Optionally, you can select another type of interpolation that will be used in the algorithm.

The parameters required are the expected extents in all dimensions.

The algorithm uses the following to calculate the appropriate new dimension size as a power of 2.

User inputs of  $\leq 16$  will result in 16.

User inputs of  $>16$  and  $\leq 40$  will result in 32

User inputs of  $>40$  and  $\leq 80$  will result in 64

User inputs of  $>80$  and  $\leq 160$  will result in 128

User inputs of  $>160$  and  $\leq 448$  will result in 256

---

### IMAGE TYPES

You can apply this algorithm to color and black-and-white images as well as 2D and 3D image types.

## Applying the Transform to Power of 2 Algorithm

To run the algorithm, complete the following steps:

- 1 Open an image of interest;
- 2 Select Algorithms > Transformation tools > Transform to power of 2;
- 3 Input the expected dimension sizes;
- 4 Optionally, select type of interpolation to be used in algorithm;
- 5 Press OK.

The algorithm begins to run, and a progress bar appears with the status. When the algorithm finishes running, the progress bar disappears, and the results appear as a file in selected format in a chosen location. For more information about the dialog box options, refer to Figure 1.

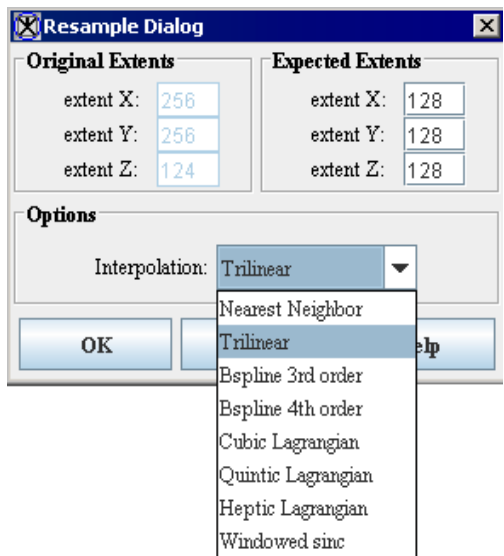
<b>Original Extents extent X</b>	X dimension of original image.	
<b>Original Extents extent Y</b>	Y dimension of original image.	
<b>Original Extents extent Z</b>	Z dimension of original image (if image is 3D).	
<b>Expected Extents extent X</b>	Expected X dimension.	
<b>Expected Extents extent Y</b>	Expected Y dimension.	
<b>Expected Extents extent Z</b>	Expected Z dimension (if image is 3D).	
<b>Interpolation</b>	Specifies the type of interpolation the algorithm will use when resampling. The list includes: Trilinear, B-spline 3-rd order, B-spline 4-th order, Cubic Lagrangian, Quintic Lagrangian, Heptic Lagrangian, Windowed Sinc.	
<b>OK</b>	Applies the algorithm according to the specifications in this dialog box.	
<b>Cancel</b>	Disregards any changes that you made in this dialog box and closes it.	
<b>Help</b>	Displays online help for this dialog box.	

Figure 1. The Resample dialog box options

## Watershed

*Watershed segmentation in MIPAV is an interactive algorithm that allows automatically segment the image regions of interest. The watershed transformation performs gray scale segmentation using the topographical approach. Interpreting an image as a topographical relief, the algorithm performs flooding starting from regional minima, which are seen as plateaus of constant altitude surrounded only by pixels of higher gray levels. Dams or watershed lines are built when pixels from different basins meet or when the pixel's gray level reaches a certain threshold value which represents the image's background.*

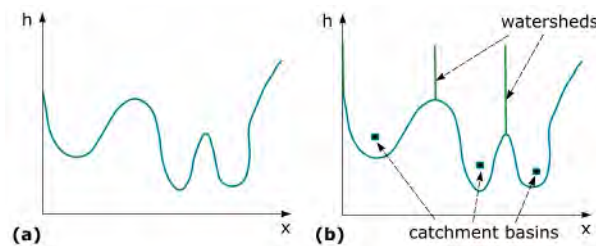


Figure 1. 1D example of watershed segmentation: (a) grey level profile of the image data, (b) watershed segmentation

## Background

- 1** The user is asked to identify the *background pixels* and *initial region(s)* using VOIs. At least two VOIs are required – the first one to specify the image's background and the second one to outline the image region(s) the user wish to segment (this VOI(s) represents the initial region(s)).

---

Note that the order in which VOIs are defined on the image is important. The first VOI must always outline background pixels.

---

- 2** Inside each initial VOI, the algorithm finds the seed point and forms the seed vector.
- 3** For each seed point, it finds the local minimum and labels it as a catchment basin.

- 4** Then, the algorithm starts filling all catchment basins. E.g. imagine that there is a hole in each local minimum, and that the topographic surface is immersed in water and water starts filling all catchment basins from the bottom. If two catchment basins merge as a result of further immersion or when the pixel gray values from one of the basins reach a threshold determined by the background intensity values, a dam is built all the way to the highest surface altitude and the dam represents the watershed line. See Figure 1.
- 5** The algorithm, first, sorts the pixels in increasing order of their gray values, and then it performs a flooding step.
- 6** During the sorting step, the algorithm computes a gray-level histogram. And it also creates a vector **H** of pointers to pixels of gray-level  $h$ . For all gray-levels, starting from below pixels having gray-level  $h$  are processed (flooded). These pixels are directly accessed through the vector **H**.
- 7** Then, the algorithm computes the geodesic influence zones of the already labeled catchment basins at the next level  $h-1$ . The geodesic influence zone of a basin  $B_i$  are the pixels of gray-level  $h_i$  which are closer to  $B_i$  than to any other catchment basin  $B_j$ . See Figure 1 and Figure 2.
- 8** The algorithm assigns the pixels having a certain gray-level  $h_i$  to the corresponding geodesic influence zone.
- 9** Pixels that do not belong to the labeled catchment basin or geodesic influence zone represent the background.

Raw watershed segmentation produces a severely over segmented image with hundreds or thousands of catchment basins. To overcome that problem the following techniques are used in this implementation of the algorithm:

- Smoothing using Gaussian blur to remove small local minima before applying the algorithm.
- The user is asked to specify the image region(s) he (she) wishes to segment using VOIs.
- Post-processing that includes merging of found regions.



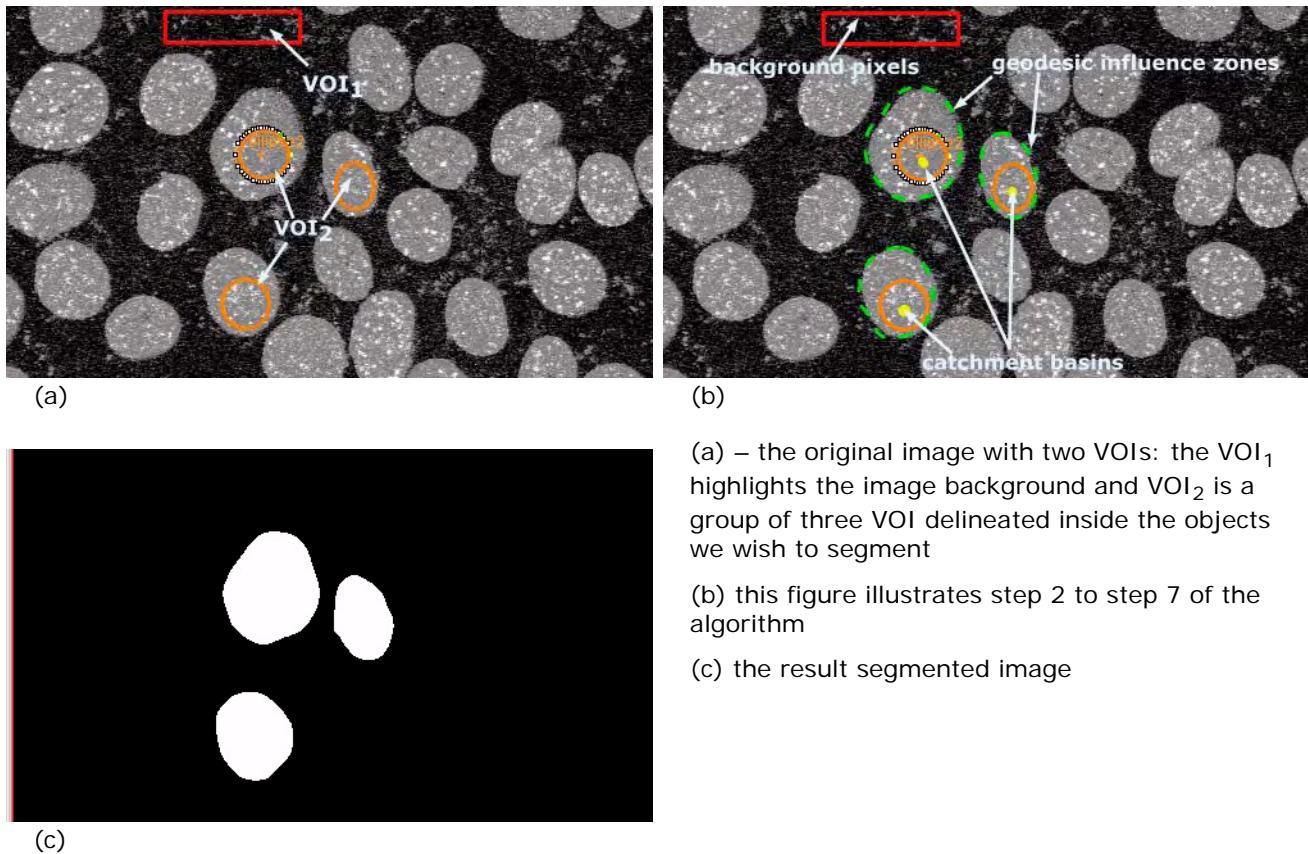


Figure 2. Segmenting images using the interactive Watershed algorithm

## REFERENCES

The following sources were used to create this help.

<http://www.icaen.uiowa.edu/~dip/LECTURE/Segmentation3.html#watershed>

<http://www.itk.org/HTML/WatershedSegmentationExample.html>

## IMAGE TYPES

You can apply this algorithm to 2D and 3D grayscale images.

## Applying the Watershed algorithm

Before applying the algorithm,

- Convert color images to grayscale ones;
- Smooth the image to remove noise if needed, use the algorithms available via the Algorithms > Filters (spatial) menu.

To run the algorithm

- 1 Open an image of interest.
- 2 In the image, delineate at least two VOIs – the first one for background and the second one for the object(s) you wish to segment.
- 3 In the MIPAV window, select Algorithms > Segmentation > Watershed. The Watershed dialog box (see Figure 3) appears.
- 4 Complete the information in the dialog box.
- 5 Click OK. The algorithm begins to run, and a progress bar appears momentarily with the status. When the algorithm finishes running, the progress bar disappears, and the result appears in a separate image window. See Figure 2-c.

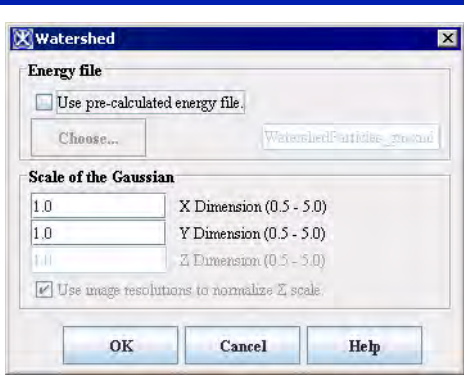
<b>Energy file</b>		
<b>Use pre-calculated energy file</b>	The energy file is generally a gradient magnitude image and the intensity peaks in this image are the watersheds boundaries. The algorithm creates and saves the energy file automatically when you first time run the algorithm for a chosen image. When you run the algorithm for the same image next time you have an option to open and use the previously saved energy file for your segmentation.	
<b>Scale of the Gaussian</b>	Specify the scale of the Gaussian for each dimension. The Z Dimension option is only available for 3D images.	
<b>Use image resolutions to normalize Z scale</b>	If checked, this option allows using the image resolutions to normalize the voxel size in Z dimension.	
<b>OK</b>	Applies the Watershed algorithm according to the specifications in this dialog box.	

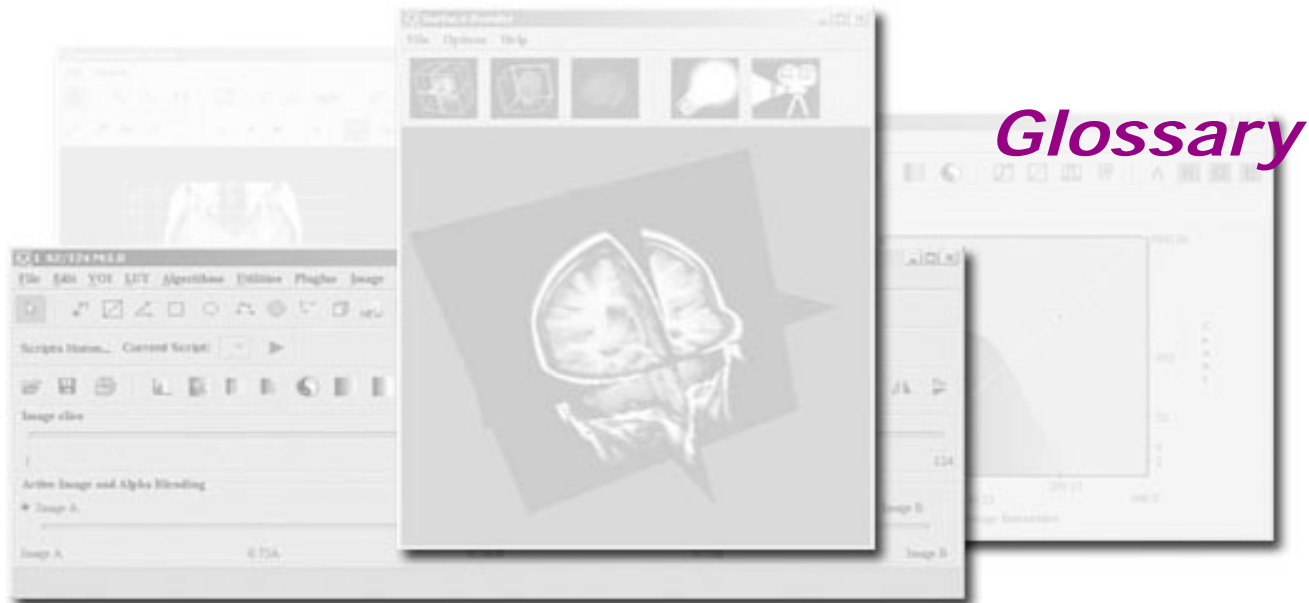
Figure 3. Watershed dialog box

---

<b>Cancel</b>	Disregards any changes that you made in the dialog box and closes this dialog box.
<b>Help</b>	Displays online help for this dialog box.

---

**Figure 3. Watershed dialog box (continued)**



This glossary defines all acronyms and selected terms used in this guide.

## Numerics

**2D.** Two dimensional.

**3D.** Three dimensional.

**4D.** Four dimensional.

**5D.** Five dimensional.

## A

**ACR.** American College of Radiology. The ACR, in conjunction with National Electrical Manufacturers Association, developed the Digital Image Communication in Medicine standard.

**AE.** Application entity.

**Analyze.** Unix-based medical-imaging display and analysis software developed by the Mayo Foundation. MIPAV allows researchers to process, analyze, and visualize Analyze-formatted image files on virtually any platform.

---

**API.** Application program interface. Pieces of code that are used to perform common tasks, such as generate a standard window frame. Software developers often incorporate these pieces of code into their programs. An API is analogous to a package of form letters; APIs reduce programming time because common functions have already been written.

**BMP.** Extension for Windows Bitmap formatted files. BMP is the standard bitmap graphics file format that is used in the MS Windows environment.

**boolean.** This data type refers to data that represents symbolic relationships between entities, such as those implied by the logical operators AND, NOT, and OR. Examples of valid boolean values are TRUE and FALSE.

**bytecode.** Compiled format for Java code. Bytecode is analogous to object code. When the Java program is written and compiled, the compiled program is written in bytecode. When you execute the bytecode program, it is interpreted by the platform-specific Java Virtual Machine, which serves as an interface between your platform and the platform-independent bytecode. Java bytecode can be ported to almost any platform and executed, provided the correct Java Virtual Machine has been installed.

**CIT.** Center for Information Technology. CIT provides, coordinates, and manages information technology so that computational science at the National Institutes of Health is advanced.

**color 24.** Color 24 is commonly referred to as 24-bit color images. Full [RGB](#) color requires that the intensities of three color components be specified for each and every pixel. It is common for each component intensity to be stored as an 8-bit integer, and so each pixel requires 24 bits to completely and accurately specify its color. Image formats that store a full 24 bits to describe the color of each and every pixel are therefore known as 24-bit color images.

**CR.** Computed radiography.

**C-STORE.** Composite Storage.

**CT.** Computed Tomography.

**data type.** A set of values from which a variable, constant, function, or expression may take its value. MIPAV accommodates the following data types: Boolean, Signed Byte, Unsigned Byte, Signed Short, Unsigned Short, Integer, Long, Float, Double, and Color 24.

**DCB.** Division of Computational Bioscience. DCB is a research and development organization that provides engineering and computer science expertise to support biomedical research activities at the National Institutes of Health (NIH). DCB applies image processing and medical imaging technologies, high-performance parallel computing, high-speed networking, signal processing, state-of-the-art optical and electronic devices, bioinformatics, database technology, mathematical and statistical techniques, and modern hardware and software engineering principles to help solve biomedical research problems at NIH.

**DICOM.** Digital Image Communication in Medicine. Standard protocol developed by the American College of Radiology (ACR) and National Electrical Manufacturers Association (NEMA). Specifies a standard method of communication between two devices.

**Double.** Primitive, 64-bit, data type. Double is a floating point data type that accommodates decimal values, up to 14 or 15 significant digits of accuracy. Valid values can range from  $-1.7 \times 10^{308}$  to  $1.7 \times 10^{308}$ .

**Endian.** Data organization strategy. Refers to the way computer processors store data in memory. Big-endian format stores the most significant byte (MSB) first. Little-endian format stores the least significant byte (LSB) first.

**FF.** Feet first.

**Float.** Primitive, 32-bit, data type. Float is a floating point data type that accommodates decimal values, up to 6 or 7 significant digits of accuracy. Valid values can range from  $-3.4 \times 10^{38}$  to  $3.4 \times 10^{38}$ .

**FTP.** File Transfer Protocol.

**GIF.** Graphic Interchange Format. A compressed, bit mapped, graphics file format that supports color and various resolutions.

**GUI.** Graphical user interface. A user interface that is based on graphics rather than text.

**header offset.** Space reserved at the beginning of some graphic files that contain non-image data.

**HF.** Head First

**HP.** Hewlett-Packard.

**HSB.** Hue Saturation Brightness. In this color model, hue is the color, saturation is the purity of the color; and brightness indicates the brightness or darkness of the color.

**ID.** Identifier.

**IE.** Information Entity.

**integer.** Primitive, 32-bit, data type. Integer is sometimes abbreviated as int. Integer accommodates values that are whole numbers. Valid values range from  $-2,147,483,648$  to  $+2,147,483,648$ .

---

**interlaced.** A display technique that increases resolution. Half of the horizontal lines are drawn on the monitor during the first pass; the other half are drawn during the second pass. For example, the odd numbered lines may be drawn during the first pass and the even numbered lines during the second pass.

**interpolation.** The generation of intermediate values based on known values.

**IOD.** Information Object Definition

**IOD.** Information Object Definition. Provides an abstract definition of real-world objects applicable to the communication of digital medical information.

**IP.** Internet Protocol.

**Java.** High-level, object-oriented, platform-independent programming language developed by Sun Microsystems.

**Java VM.** Java Virtual Machine.

**JIT.** Just-In-Time compiler. The JIT converts Java bytecode into machine language instructions.

**JPEG.** Extension for Joint Photographics Experts Group formatted files. Also refers to a compression type.



---

**Linux.** An operating system that is an open source implementation of UNIX.

**Long.** Primitive, 64-bit data type. Long is a variation of the integer data type. Long accommodates values that are whole numbers. Valid values range from -9,223,372,036,854,775,808 to +9,223,372,036,854,775,808.

**LSB.** Least Significant Byte. Also see endian.

**LUT.** Lookup Table.

**Mac OS.** Macintosh Operating System

**MB.** Megabyte

**MIPAV.** Medical Image Processing, Analysis, and Visualization program. MIPAV is an n-dimensional, general-purpose, extensive image processing, and visualization program. It is platform-independent and assists researchers with extracting quantitative information from various medical imaging modalities.

**MR.** Magnetic Resonance.

**MSB.** Most Significant Byte. See endian for more details.

**MSEE.** Master of Science in Electrical Engineering

**MTX.** Extension for MIPAV's transform matrix files.

**NEMA.** National Electrical Manufacturers Association.

**NIH.** National Institutes of Health.

**NM.** Nuclear medicine.

**OS.** Operating system

**PACS.** Picture Archiving System.

**PCX.** Extension for PC Paintbrush formatted graphic files.

**PDU.** Protocol Data Unit.

**PET.** Positron Emission Tomography.

**PICT.** Extension for Macintosh formatted graphic files.

**PLT.** Extension for MIPAV's graphics files.

**PNG.** Extension for Portable Network Graphic formatted graphic files.

**PSD.** Extension for Adobe Photoshop formatted graphic files.

**RAM.** Random Access Memory

**Raster.** Bitmap file type.

**Raw.** File type.

**resolution.** The sharpness and clarity of an image.

**RGB.** Red Green Blue.

**RIS.** TBD.

**RLE (Run Length Encoding).** The file extension for graphics that have been reduced using run-length encoding. RLE is a compression method that converts consecutive identical characters into a code consisting of the character and the number marking the length of the run.

**ROI.** Region of Interest.

**RS.** Extension for Sun Raster formatted graphics files.

---

**SCP.** Service Class Provider.

**SCU.** Service Class User.

**SGI.** Silicon Graphics Incorporated.

**short.** Primitive, 16-bit data type. Short is a variation of the integer data type. Short accommodates values that are whole numbers. Valid values range from 0 to +32,767.

**signed byte.** Primitive, 8-bit, data type. Signed byte is a variation of the integer data type. The signed byte data type signifies that valid values fall within a range of whole numbers. Valid values range from -128 to +128. Negative values (indicated by the negative sign) are permitted, hence the term, signed byte.

**signed short.** Primitive, 16-bit data type. Signed short is a variation of the integer data type. The signed short data type signifies that valid values fall within a range of whole numbers. Valid values range from -32,768 to +32,767. Negative values (indicated by the negative sign) are permitted, hence the term, signed short.

**Solaris.** Unix-based operating environment that was developed by Sun Microsystems. Solaris consists of the Sun operating system and a windowing system.

**SOP.** Service Object Pair

**SOP.** Service Object Pair.

**SPECT.** TBD.

**TCP/IP.** Transmission Control Protocol/Internet Protocol. The suite of communications protocols used to connect hosts on the Internet.

**TGA.** Extension for Truevision Graphics Adapter formatted graphics files.

**TIFF.** Extension for Tag Image File Format formatted graphics files.

---

**UID.** Unique Identifier.

**UNIX.** Multi-tasking, multi-user operating system developed by Bell Labs. Many versions of UNIX abound, including Linux.

**unsigned byte.** Primitive, 8-bit, data type. Unsigned byte is a variation of the integer data type. The unsigned byte data type signifies that valid values must fall within a specified range of positive, whole-number values. Valid values range from 0 to +128. Negative values (indicated by the negative sign) are not valid, hence the term, unsigned byte.

**unsigned short.** Primitive, 16-bit data type. Unsigned short is a variation of the integer data type. The unsigned short data type signifies that valid values must fall within a specified range of positive, whole-number values. Valid values range from 0 to +32,767. Note that negative values (indicated by the negative sign) are not valid, hence the term, unsigned byte.

**US.** Ultrasound

**VM.** Virtual Machine.

**VOI.** Volume of interest (used interchangeably with ROI).

**voxel.** Smallest distinguishable cube-shaped part of a 3D image.

**XBM.** X BitMap file format.

**XPM.** X PixMap file format.