MIPAV

Medical Image Processing, Analysis, & Visualization

User's Guide

Volume 2

ALGORITHMS

August 31, 2007

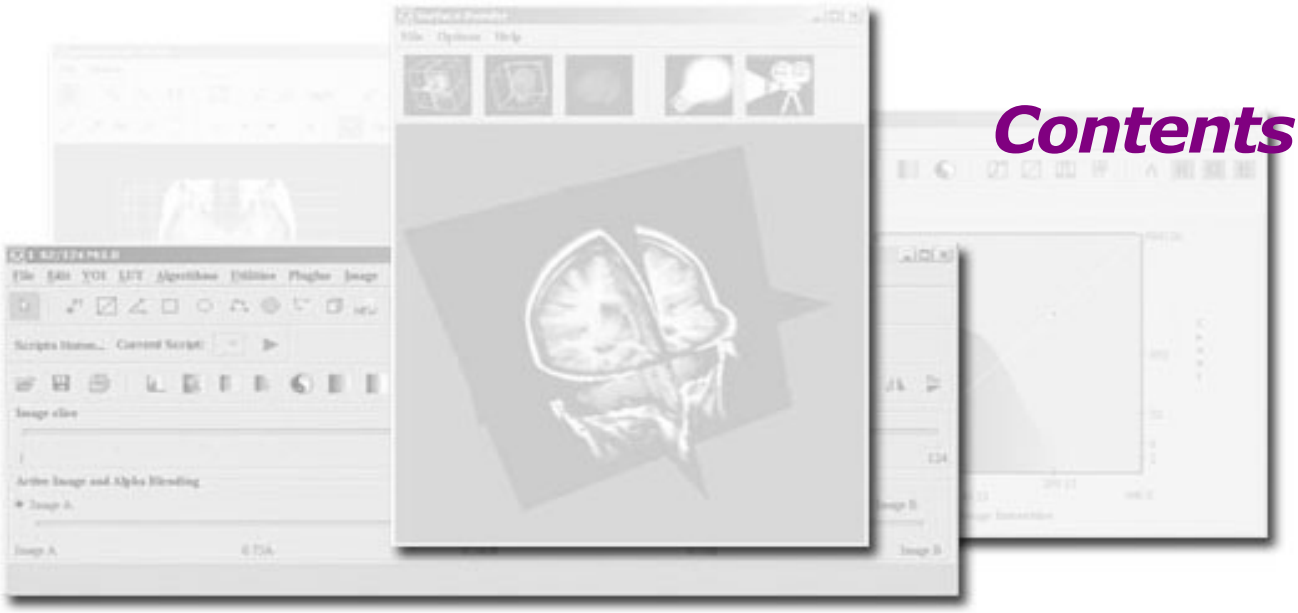
National Institutes of Health
Center for Information Technology
Rockville, Maryland

August 31, 2007

Matthew McAullife, PhD 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. 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.



Contents

List of Figures 14

List of Tables 22

Preface 23

Scope of this guide..... 23

Who should use this guide..... 23

Skills you need to perform tasks in MIPAV 24

How this guide is organized 24

Volume 1, Basics 24
Volume 2, Algorithms..... 26

Where to find more information 26

Conventions used in this guide..... 26

Chapter 1
MIPAV Algorithms Overview29

Understanding MIPAV capabilities 30

Developing new tools using the API 31

Chapter 2 **Using MIPAV Algorithms** 32

Extract Brain: Extract Brain Surface (BET) 38

Background 38
Step 1, Estimating image parameters *38*
Step 2, Selecting an initial closed mesh *39*
Step 3, Evolution of the mesh *42*
Step 4, Selecting brain voxels *45*
Image types *46*
Special notes *47*
Reference *47*
Applying the Extract Brain Surface algorithm 47

Extract Brain: Extract Brain Surface (BSE) 50

Background 50
Step 1, Filtering the image to remove irregularities *50*
Step 2, Detecting edges in the image *50*
Step 3, Performing morphological erosions and brain isolation *51*
Step 4, Performing surface cleanup and image masking *51*
Selecting parameters *51*
Edge detection kernel size parameter 52
Image types *52*
References *52*
Applying the Extract Brain Surface (BSE) algorithm 53

Face Anonymizer (BET) 57

Background 57
Step 1, Extract the brain *57*
Step 2: Buffer the brain *58*
Step 3: Define the face *58*
Step 4: Extract the face *58*
Step 5: Smooth the face *58*
Image types *59*
Notes *59*
Reference: *59*
Applying the Face Anonymizer Algorithm 59

Fast Fourier Transformation (FFT) 62

Background 63
Finite Impulse Response (FIR) filters with Hamming windows *64*
Gaussian and Butterworth filters *65*
Butterworth 2D and 3D filters 65
Butterworth low-pass, high-pass, band-pass, and band-stop filters 65
Gaussian low- and high-pass filters 66

Image types *67*
Special notes *67*
References *68*
Applying the FFT algorithm 68
Step 1: Converting dataset from spatial to fourier domain using the forward FFT command *69*
Step 2: Enhancing and attenuating frequency components using Frequency Filter *70*
Optional Step: Attenuating frequency components using the Paint tool *72*
Step 3: Reconverting the image to the spatial domain *74*

Filters (Spatial): Adaptive Noise Reduction 75

Background 75
Image types *77*
Special notes *77*
References *78*
Applying the Adaptive Noise Reduction algorithm 78

Filters (Spatial): Adaptive Path Smooth 80

Background 80
Image types *84*
Special notes *84*
Reference *84*
Applying the Adaptive Path Smooth algorithm 84

Filters (Spatial): Anisotropic Diffusion 87

Background 88
Image types *89*
Special notes *89*
References *89*
Applying the Anisotropic Diffusion algorithm 90

Filters (Spatial): Coherence-Enhancing Diffusion 92

Background 93
Image types *95*
Special notes *95*
References *96*
Applying the Coherence-Enhancing Diffusion algorithm 96

Filters (Spatial): Gaussian Blur 98

Background 98
Image types *101*
Special notes *101*
References *102*
Applying the Gaussian Blur algorithm 102

Filters (Spatial): Gradient Magnitude 104

Background 104
Image types *105*
Special notes *106*
References *106*
Applying the Gradient Magnitude algorithm 106

Filters (Spatial): Laplacian 108

Background 108
Image types *109*
Special notes *109*
References *110*
Applying the Laplacian algorithm 110

Filters (Spatial): Mean 112

Background 112
Image types *115*
Special notes *115*
References *115*
Applying the Mean algorithm 115

Filters (Spatial): Median 117

Background 117
Image types *120*
Special notes *120*
References *121*
Applying the Median Filter algorithm 121

Filters (Spatial): Mode 124

Background 124
Image types *126*
Notes *126*
References *126*
Applying the Mode algorithm 126

Filters (Spatial): Nonlinear Noise Reduction 129

Background 129
Image types *131*
Special notes *131*
References *131*
Applying the Nonlinear Noise Reduction algorithm 132

Filters (Spatial): Nonmaximum Suppression 134

Background 134
Image types *138*
Special notes *138*
References *138*
Applying the Nonmaximum Suppression algorithm 139

Filters (Spatial): Regularized Isotropic (Nonlinear) Diffusion... 141

Background ... 141
Image types ... *144*
Special notes ... *144*
References... *144*
Applying the Regularized Isotropic (Nonlinear) Diffusion algorithm ... 145

Filters (Spatial): Slice Averaging ... 146

Background ... 146
Image types ... *148*
Special notes ... *149*
References... *149*
Applying the Slice Averaging algorithm ... 149

Filters (Spatial): Unsharp Mask ... 151

Background ... 151
Image types ... *151*
Special notes ... *152*
References... *152*
Applying the Unsharp Mask algorithm ... 153

Fuzzy C-Means: Multispectral and Single Channel Algorithms . 155

Hard segmentation ... *155*
Soft, or fuzzy, segmentation ... *155*
Background ... 155
Adding images to the list ... *158*
Removing images from the list ... 158
Image types ... *159*
Special notes ... *159*
References... *159*
Applying the Fuzzy C-means algorithm ... 160
On single images ... *161*
On multispectral images ... *162*

Histogram Equalization: Regional Adaptive ... 165

Background ... 165
Image types ... *169*
Special notes ... *169*
References... *170*
Applying the Histogram Equalization: Regional Adaptive algorithm ... 172

Histogram Equalization: Neighborhood Adaptive... 174

Background ... 174

Image types *182*
Special notes *182*
References *182*
Applying the Histogram Equalization: Neighborhood Adaptive algorithm 183

Histogram Matching 186

Background 186
Image types *189*
Notes *189*
References *189*
Applying the Histogram Matching algorithm 189

2D Histogram 191

Background 191
Image types *192*
Notes *192*
Applying the 2D Histogram algorithm to grayscale images 193
Applying the 2D Histogram algorithm to color images 194

Creating a histogram summary 196

Applying Histogram Summary to the image 196
The histogram summary table includes the following four columns, see Figure 73: 196

Mask 199

Background 199
Image types *200*
Special notes *200*
References *200*
Applying the Mask algorithm 200

Microscopy: Colocalization Orthogonal Regression 203

Background 204
Images types *210*
Special notes *210*
References *211*
Applying the Microscopy: Colocalization Orthogonal Regression algorithm 212
Colocalization histogram window *216*

Microscopy: Fluorescence Resonance Energy Transfer (FRET)—Acceptor Photobleaching 221

Background 221

Image types *223*
Notes *223*
References *224*
Applying the FRET algorithm 224

Microscopy: FRAP (Fluorescence Recovery After Photobleaching) 230

Background 230
Image types *243*
Special notes *243*
References *243*
Applying the FRAP algorithm 244

Microscopy: Blind Deconvolution 259

Background 259
Computing convolution *261*
PSF constraints *261*
Image types *262*
Notes *262*
References *263*
Useful links *263*
Applying the Maximum Likelihood Iterative Blind Deconvolution algorithm 264
Selecting the algorithm parameters 266

Optimized Automatic Registration 3D 269

Background 269
Image types *270*
Outline of the method *271*
Pre-optimization steps 271
Optimization steps 271
Post optimization 272
Pre Optimization *272*
Blurring 272
Resampling 272
Calculating the center of mass *273*
Process *274*
levelEight optimization 274
Optimization 274
levelFour optimization 276
levelTwo 277
levelOne 277
What is saved 277
Additional registration parameters 278
Resampling Interpolation *278*
Trilinear Interpolation 278
B-spline basis interpolations 279
Lagrange interpolations 281

Windowed sinc 283
Cost Functions 284
Powell algorithm *285*
Correlation ratio *285*
Image types 285
Least Squares *286*
Image types 286
Normalized cross correlation *287*
Normalized cross correlation 287
Image types 288
Normalized mutual information *289*
Image types 290
Degrees of freedom *290*
References *291*
Applying the Algorithm 292
To run the algorithm, complete the following steps: 292
Optimized Automatic Registration dialog box options *294*

Registration: Landmark—Least Squares 297

Background 297
Image types *300*
Special notes *300*
Reference *300*
Applying the Landmark—Least Squares Registration algorithm 301

Registration: Landmark—TPSpline 303

Background 303
Image types *304*
Special notes *304*
References *304*
Applying the Registration: Landmark—TPSpline algorithm . 305

Registration: Manual 2D Series 307

Background 307
Image types *309*
Special notes *309*
References *309*
Applying the Manual 2D Series algorithm on 2D images 310
Making manual adjustments to the image to be registered 312
Moving the adjusted image in any direction *315*
Moving the adjusted image with the direction arrows *316*
Rotating the adjusted image *316*
Applying the Manual 2D Series algorithm on 2.5 images ... 323

Registration: Time Series Optimized Automatic Registration ... 325

Background 325

Registering volumes *325*
Choosing a cost function *327*
Normalized cross correlation 327
Correlation ratio 328
Least squares 329
Normalized mutual information 329
Completing the dialog box *330*
Image types *330*
Special notes *331*
Reference *331*
Applying the Time Series Optimized Automatic Registration algorithm 331

Reslice—Isotropic Voxels 334

Background 334
Linear interpolation *334*
Cubic convolution interpolation *334*
Cubic B-spline interpolation *335*
Image types *335*
Special notes *336*
References *336*
Applying the Reslice—Isotropic Voxels algorithm 336

B-Spline Automatic Registration 337

Background 337
Error measures 341
Least squares measure *342*
Correlation ratio measure *342*
Normalized mutual information measure *343*
Improving performance 344
Initial placement of control points for identity map 345
Composition for two-pass registration *347*
Deformation 348
Detect folding 349
Support for color images 351
Support for 2.5D images 351
Examples of B-spline registration *352*
User Dialogs in MIPAV 353
Transformation options 363
Class implementation for MIPAV 365

Shading Correction: Inhomogeneity N3 Correction 366

Background 366
How the N3 method works *367*
Mapping from v to f 368
Algorithm flow chart 368

Image Types *375*
Special Notes *375*
References *375*
Applying the Inhomogeneity N3 algorithm 375
Choosing good parameters for N3 Algorithm 376
Extract Surface (Marching Cubes) 379
Background 379
References *382*
Image types *383*
Output Files 383
Applying the Extract Surface Algorithm 383
To run the algorithm, complete the following steps: 383
Examples *385*
Transform to power of 2 391
Background 391
Image types *391*
Applying the Transform to Power of 2 Algorithm 392
To run the algorithm, complete the following steps: 392
Registration: Manual 2D Series 398
Background 398
Image types *398*
Special notes *399*
References *399*
Applying the Manual 2D Series algorithm on 2D images 399
To run this algorithm, complete the following steps: 399
Options available via the dialog menu 400
Blended tab 400
Changing displaying proportion *401*
The Blended tab options: *401*
Moving and rotating the adjusted image *403*
Moving the adjusted image *404*
Moving the adjusted image with the direction arrows *404*
Rotating the adjusted image *405*
Dual tab *406*
The Dual tab offers the following options *407*
Using the image registration algorithms *409*
Delineating the landmark points *410*
Applying the image registration algorithms *410*
making manual reconcilements to the adjusted image *411*
Using LUT to arrange the images *413*
Applying the Manual 2D Series algorithm to 2.5D images .. 415
Applying the Manual 2D Series algorithm to 3D images 417
Threshold 420

Background 420
Image types *420*
Applying the Threshold algorithm *420*
The Threshold dialog box options: *421*

Subsampling images 423

Glossary 426

List of Figures

Figure 1. The initial ellipsoids intersected with the middle slices 40

Figure 2. Subdivision of a triangle 41

Figure 3. An initial ellipsoid: (A) solid and (B) wireframe 42

Figure 4. Two views of a brain surface mesh 45

Figure 5. MRI slices with colored brain voxels 46

Figure 6. Extract Brain dialog box 48

Figure 7. Examples of Extract Brain Surface (BSE) image processing..... 53

Figure 8. Extract Brain Surface (BSE) algorithm dialog box..... 54

Figure 9. 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. 56

Figure 10. An example of the Face Anonymizer (BET) image processing. The image has been processed using default parameters 60

Figure 11. Face Anonymizer dialog box options..... 61

Figure 12. FFT algorithm processing 63

Figure 13. FFT dialog box 71

Figure 14. Adaptive Noise Reduction processing..... 77

Figure 15. Adaptive Noise Reduction dialog box 79

Figure 16. Example of adaptive path smooth processing 83

Figure 17. Adaptive Path Smooth dialog box for grayscale images 85

Figure 18. Adaptive Path Smooth dialog box for 3D color images 86

Figure 19. A 1D example of anisotropic diffusion 87

Figure 20. An image before and after applying the Anisotropic Diffusion algorithm 89

Figure 21. Anisotropic Diffusion dialog box..... 91

Figure 22. Example of coherence-enhancing diffusion 95

Figure 23. Coherence Enhancing Diffusion dialog box 96

Figure 24. Status message that appear when the Coherence-Enhancing Diffusion algorithm is running..... 97

Figure 25. Gaussian Blur algorithm processing..... 100

Figure 26. Gaussian function 101

Figure 27. Gaussian Blur dialog box 103

Figure 28. Edge detection using derivative operators 104

Figure 29. Gradient Magnitude algorithm processing 105

Figure 30. Gradient Magnitude dialog box..... 107

Figure 31. Edge detection using laplacian operators 108

Figure 32. (A) Original MR image; (B) laplacian results; and (C) extraction of the zero crossing of the laplacian (object edges)..... 109

Figure 33. Laplacian dialog box 111

Figure 34. Slice averaging..... 113

Figure 35. Image processing using the Mean algorithm 114

Figure 36. Mean Filter dialog box 116

Figure 37. An example of median filter processing using a square 3 x 3 kernel 118

Figure 38. Removing shot noise with the Median Filter algorithm 120

Figure 39. Median Filter dialog box..... 122

Figure 40. Examples of (A) a segmented image and (B) the results of applying the Mode algorithm to the image 125

Figure 41. Mode Filter dialog box.....127

Figure 42. Image processing using the Nonlinear Noise Reduction algorithm 131

Figure 43. Nonlinear Noise Reduction dialog box..... 132

Figure 44. Examples of Nonmaximum Suppression processing..... 138

Figure 45. Nonmaximum Suppression dialog box 140

Figure 46. Diffusion constant for 143

Figure 47. Examples of regularized isotropic (nonlinear) diffusion..... 144

Figure 48. Regularized Isotropic Diffusion dialog box..... 145

Figure 49. Slice Averaging algorithm processing 148

Figure 50. Slice Averaging dialog box 149

Figure 51. Unsharp Mask processing 152

Figure 52. Unsharp Mask dialog box 153

Figure 53. Fuzzy C-means algorithm processing..... 160

Figure 54. Fuzzy C-Means dialog box for a single image.....161

Figure 55. Fuzzy C-Means dialog box for multispectral images..... 163

Figure 56. 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..... 166

Figure 57. Examples of no clipping, 45-percent clipping, and 75-percent clipping.... 169

Figure 58. The Regional Adaptive Histogram Equalization dialog box 172

Figure 59. Sample of kernel created by algorithm.....176

Figure 60. 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 177

Figure 61. The effect of thresholding on images 180

Figure 62. A reference image and histogram with no clipping and with 75-percent clipping 181

Figure 63. A reference image and histogram with 45-percent clipping 182

Figure 64. Local Adaptive Histogram Equalization dialog box 184

Figure 65. 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). ... 187

Figure 66. The input image and its histogram p(f) 188

Figure 67. The specified image and its histogram pd(g) 188

Figure 68. The result image and its histogram 188

Figure 69. Histogram Matching dialog box 190

Figure 70. Two input images, output 2D histogram image and its lookup table .. 192

Figure 71. The Histogram Two Dimensional dialog box for grayscale images 193

Figure 72. The Histogram Two Dimensional dialog box for color images 195

Figure 73. Histogram summary in the Output window 197

Figure 74. The Histogram summary dialog box options 198

Figure 75. Mask algorithm processing 199

Figure 76. Mask Image dialog box 201

Figure 77. Orthogonal Regression Colocalization dialog boxes for (A) color and (B) grayscale images 213

Figure 78. 2D Histogram window 216

Figure 79. Colocalization Histogram window showing a histogram for a color image in which red and green color localization was done 218

Figure 80. The effect on the image when dragging the mouse-movable point on the histogram (part 1) 219

Figure 81. The effect on the image when dragging the mouse-movable point on the histogram (part 2) 220

Figure 82. Load Bleached ROI message 224

Figure 83. Fluorescence Resonance Energy Transfer (FRET) dialog box 226

Figure 84. Example of (A) prebleached image and (B) postbleached image after FRET algorithm was applied 227

Figure 85. FRET data in the Data page of the Output window 229

Figure 86. Fluorescence Recovery after Photobleaching dialog box 245

Figure 87. An image with a required photobleached VOI in red and a whole organ VOI in green 247

Figure 88. Graph windows: Photobleaching Recovery Curve, Uncorrected Photobleaching Recovery Curve, and Uncorrected Whole Organ windows 249

Figure 89. Print dialog box 251

Figure 90. Modify Graph Features command on the Views menu in a graph window 252

Figure 91. Modify Graph dialog box showing the Graph page 253

Figure 92. Pick Background Color dialog box 254

Figure 93. Legend page in the Modify Graph dialog box 256

Figure 94. Functions page in the Modify Graph dialog box 257

Figure 95. Fitted Functions page in the Modify Graph window 258

Figure 96. The original color image (A), the intermediate grayscale image (B), and the final estimate color image after 10 iterations (C). 262

Figure 97. The original image of Jupiter in grayscale (A), the estimate image after five iterations (B), and the final estimate after 10 iterations (C). 262

Figure 98. The Maximum Likelihood Iterated Blind Deconvolution dialog box 264

Figure 99. A status pop-up window. 265

Figure 100. The PSF warning. 266

Figure 101. 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. 267

Figure 102. 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 270

Figure 103. Calculating the center of mass (COM) 273

Figure 104. levelEight optimization 276

Figure 105. Trilinear interpolation 279

Figure 106. B-spline interpolation 280

Figure 107. 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 282

Figure 108. T y=sinc(a), where a=px, xÎ [-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 210 = 1024. 284

Figure 109. The input image (A), reference image (B) and result image (C). Registration was performed using the default algorithm settings. 293

Figure 110. The Optimized Automatic Registration dialog box options 294

Figure 111. Landmark—Least Squares processing 300

Figure 112. Least Squares Registration dialog box 302

Figure 113. Example of Registration: Landmark—TPSpline algorithm 304

Figure 114. Thin Plate Spline Registration dialog box 306

Figure 115. Registration: Manual 2D Series window 307

Figure 116. The Load dialog box 310

Figure 117. The Blended page of the Manual 2D Series window showing the adjusted image moved down and to the right 315

Figure 118. Manual 2D Series window showing center of rotation 317

Figure 119. Manual 2D Series window showing the Blended page 318

Figure 120. Manual 2D Series window showing the Dual page 321

Figure 121. Manual 2D Series window for 2.5D images 324

Figure 122. The Time Series Registration dialog box 332

Figure 123. Reslice—Isotropic Voxels algorithm applied to an image 335

Figure 124. Reslice dialog box 336

Figure 125. 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. 352

Figure 126. 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. 353

Figure 127. The choice of 2D/3D or 2.5D B-spline registration is made by selecting the appropriate menu item from the Algorithms: Registration submenu 354

Figure 128. B-Spline Single - Pass Automatic Registration 2D/3D dialog box..... 355

Figure 129. B-Spline Two-Pass Automatic Registration 2D/3D dialog box..... 357

Figure 130. B-Spline Single-Pass Automatic Registration 2.5D dialog box 359

Figure 131. B-Spline Automatic Two-Pass Registration 2.5D dialog box..... 361

Figure 132. Flow chart describing the N3 method.....371

Figure 133. The N3 method showing the equations 372

Figure 134. The effect of N3 on an image: (A) original image and (B) image after N3 correction 373

Figure 135. The effects of different settings for the Inhomogeneity N3 algorithm on an image 374

Figure 136. N3 Inhomogeneity Correction dialog box..... 378

Figure 137. The indexing convention for vertices and edges. Here, edge indexes are shown in blue and vertex indexes are shown in magenta..... 379

Figure 138. Vertex 3 is inside the volume 380

Figure 139. How 256 cases can be reduced to 15 cases..... 381

Figure 140. The Extract Surface dialog box options 384

Figure 141. The Resample dialog.....385

Figure 142. The view of the surface extracted using the VOI Region option 386

Figure 143. The view of the surface extracted using the Mask Image option.....387

Figure 144. The intensity level is shown on the main MIPAV toolbar 388

Figure 145. Enter the intensity level value here.....389

Figure 146. The view of the surface extracted using the Intensity Level option ...390

Figure 147. The Resample dialog box options.....392

Figure 148. The Load dialog box 400

Figure 149. The Blended tab and the image slider..... 401

Figure 150. The Blended tab options 401

Figure 151. The Blended tab: activating moving and rotation modes 404

Figure 152. Rotating the adjusted image.....406

Figure 153. The Dual tab options 407

Figure 154. 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.....409

Figure 155. The Window Region box is filled with the adjusted image..... 412

Figure 156. The adjusted image and its Lookup table 413

Figure 157. The adjusted image which appears in the Blended tab and also in the Window Region. 414

Figure 158. 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..... 416

Figure 159. Registering the 3D image. 418

Figure 160. The Threshold dialog box..... 421

Figure 161. Original image before subsampling.....423

Figure 162. Subsample dialog box.....424

Figure 163. An image subsampled by 2, by 4, and then by 8.....425

List of Tables

Table 1. MIPAV algorithms and their effects on images 32

Table 2. Possible responses of the SUSAN filter 129

Table 3. edgeTable 381

Preface

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," lists the file formats supported by MIPAV and 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, “Connecting to SRB BIRN” explains how to get connected to SRB.
- Chapter 7, “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 8, “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 9, “Analyzing Images,” discusses how to calculate statistics for VOIs and masks and how to generate intensity profiles, or graphs, for images.
- Chapter 10, “Changing Image Datasets Using MIPAV Utilities,” explains how to use the utilities included in the software.
- Chapter 11, “Using Scripts (Macros) in MIPAV,” describes how to develop scripts, which you can use to customize the program.
- Chapter 12, “Developing Plugin Programs,” explains how to incorporate plug-in programs into MIPAV.
- Chapter 13, “MIPAV Troubleshooting” explains how to reach MIPAV tech. support.
- 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,” provides examples of files 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. Also explains how to use the debug mode.

- 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 . . .
Italics	Names of books, guides, or manuals as references New terms or emphasis Names of executable files
Bold	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
Hyperlink	An internet link (position the cursor on this word and click the left mouse button)*

This convention . . .	Stands for . . .
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.

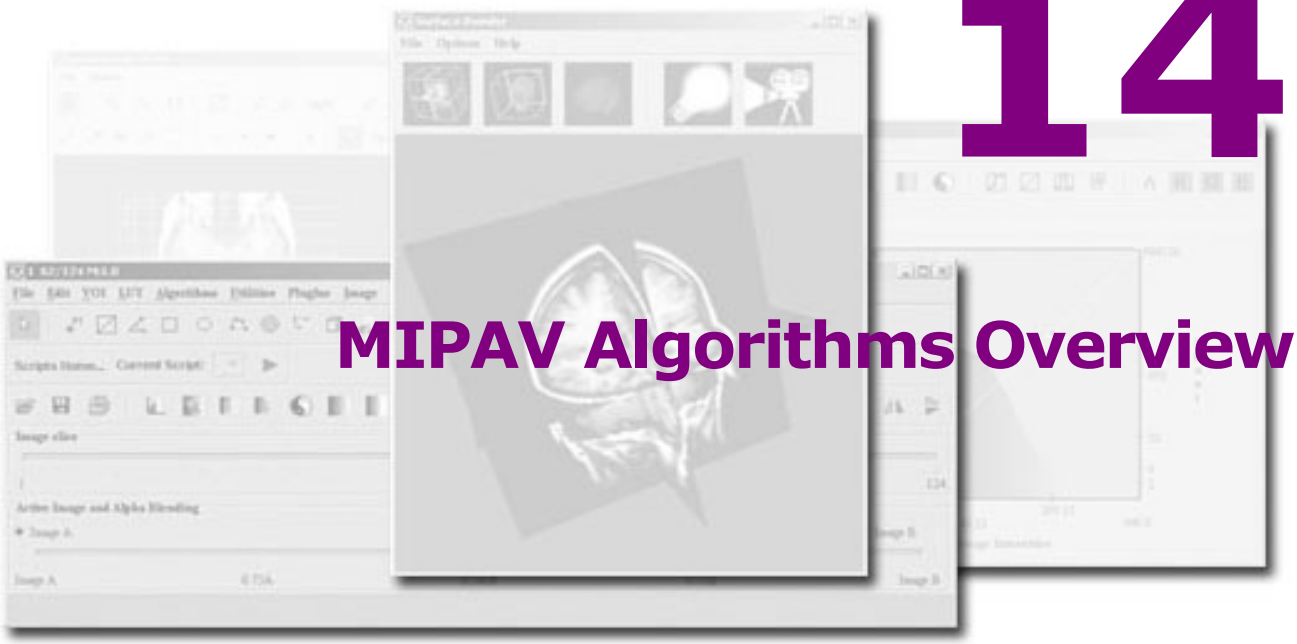
Definition: The definitions of specific words or phrases of note appear in "definition" paragraphs.

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.



14 MIPAV Algorithms Overview

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.

15 Using MIPAV Algorithms

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.

Table 1 lists both the current and planned MIPAV algorithms.

This table also explains what effect the algorithms have on images.

Table 1. MIPAV algorithms and their effects on images

Algorithm	Effect	Image types
Edge Detection: Zero X Laplacian	TBD	TBD
Edge Detection Zero X Non-max Suppress	TBD	TBD
Extract Surface: Adaptive Climbing	TBD	TBD
Extract Surface: Tetrahedron	TBD	TBD
"Extract Surface (Marching Cubes)"	Extracts surface information from a 3D array of values.	2D and 3D MRI images

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

Algorithm	Effect	Image types
Extract Brain Surface	Extracts the surface of the brain from a T1-weighted MRI.	3D MRI images
Extract Brain: Extract Brain Surface (BSE)		3D MRI images
Fast Fourier Transformation (FFT)	Processes images by filtering in the frequency domain.	• Gray-scale 2D and 3D images • Conjugate, symmetric, and complex data (frequency filter or inverse FFT only)
Filters (Frequency)	TBD	TBD
Filters (Spatial): Adaptive Path Smooth		
Filters (Spatial): Adaptive Noise Reduction	Reduces noise without blurring the edges.	• Color images • Black-and-white images
Filters (Spatial): Anisotropic Diffusion	Blurs image except across boundaries.	Grayscale images
Filters (Spatial): CoherenceEnhancingDiffusion	Useful for filtering relatively thin, linear structures such as blood vessels, elongated cells, and muscle fibers	• All data types except complex • 2D, 2.5D, and 3D images
Filters (Spatial): Gaussian Blur	Blurs image. This algorithm produces a similar result as applying a low-pass or smoothing filter.	• All image data types except Complex • 2D, 2.5D, 3D, and 4D images
Filters (Spatial): Gradient Magnitude	Generates a strong response at the edges of objects.	• All image data types except Complex • 2D, 2.5D, 3D, and 4D images
Filters (Spatial): Laplacian	Detects edges in an image by identifying meaningful discontinuities in gray level or color.	• All image data types except Complex • 2D, 2.5D, and 3D images
Filters (Spatial): Local Normalization	TBD	TBD
Filters (Spatial): Mean	Provides a simple way of reducing noise either in an entire image or in a delineated VOI in the image.	• All image data types except Complex • 2D, 2.5D, and 3D images

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

Algorithm	Effect	Image types
Filters (Spatial): Median	Removes shot noise by replacing a target pixel's value with the median value of neighboring pixels.	• Gray-scale 2D, 2.5D, and 3D images • Color 2D, 2.5D, and 3D images
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.	• Black-and-white 2D, 2.5D, and 3D images • Edge processing only applicable to black-and-white 2D images
"Filters (Spatial): Regularized Isotropic (Nonlinear) Diffusion"	Regularized isotropic nonlinear diffusion. Diffusion filtering, which models the diffusion process, is an iterative approach of spatial filtering.	• All data types except complex • 2D, 2.5D, and 3D images
Filters (Spatial): Slice Averaging	Reduces image noise.	• All image data types • All 2D, 2.5D, and 3D images
Filters (Spatial): Unsharp Mask	Produces a sharpened version of the image or a VOI of the image.	• All image data types, except complex and RGB images • 2D, 2.5D, 3D, and 4D images
Filters (Wavelet): Thresholding	TBD	TBD
Fuzzy C-Means: Multispectral	Performs both hard and soft segmentation on multiple images.	• 2D and 3D datasets • RGB datasets
Fuzzy C-Means: Single Channel	Performs both hard and soft segmentation on a single image.	• 2D and 3D datasets • RGB datasets
Histogram Equalization: Neighborhood Adaptive	Enhances the contrast in an image by reevaluating the gray-scale, or intensity, value of each pixel based on a region of nearby pixels.	• Color image • Black-and-white images
Histogram Equalization: Regional Adaptive	Enhances the contrast in an image.	• Color images • Black-and-white images • Whole images, not VOIs
Image Calculator	TBD	TBD
Image Math	TBD	TBD

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

Algorithm	Effect	Image types
In homogeneity N3 Correction	TBD	TBD
Levelset	TBD	TBD
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
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
Microscopy: FRAP	Determines an association rate, dissociation rate, and diffusion transfer coefficient in images.	Color and black-and-white 3D images
Microscopy: FRET		
Morphological: Dilate	TBD	TBD
Morphological: Erode	TBD	TBD
Morphological: Ultimate Erode	TBD	TBD
Morphological: Open	TBD	TBD
Morphological: Close	TBD	TBD
Morphological: ID Objects	TBD	TBD
Morphological: Distance Map	TBD	TBD
Morphological: Bg (Background) Distance Map	TBD	TBD
Morphological: Delete Objects	TBD	TBD
Morphological: Particle Analysis	TBD	TBD
Morphological: Find Edges	TBD	TBD
Noise	TBD	TBD
Plot Surface	TBD	TBD
Principal Component	TBD	TBD

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

Algorithm	Effect	Image types
Quantify Using Mask	TBD	TBD
Randomize Slice Order	TBD	TBD
Registration: AFNI-Shear	TBD	TBD
Registration: AIR Linear	TBD	TBD
Registration: AIR Nonlinear	TBD	TBD
Registration: Optimized Automatic Registration	TBD	TBD
Registration: Optimized Automatic Registration 2.5D	TBD	TBD
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: 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	Allows you to register 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
Registration: Patient Position (DICOM)	TBD	TBD
Registration: VOI Landmark	TBD	TBD
Reslice—Isotropic Voxels	Applies image resampling.	All image data types except RGB and complex
Shading Correction: Inhomogeneity N3 Correction	Corrects for shading artifacts often seen in MRI.	2D and 3D MRI images

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

Algorithm	Effect	Image types
Stereo Depth	TBD	TBD
Subtract VOI Background	TBD	TBD
Threshold	TBD	TBD
Transform	TBD	TBD
Transform Nonlinear	TBD	TBD
Watershed	TBD	TBD

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_{back} = histogram index with the greatest count
- Median threshold: t_{med} = median pixel value with the initial ellipsoid that are greater than t_{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_{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.

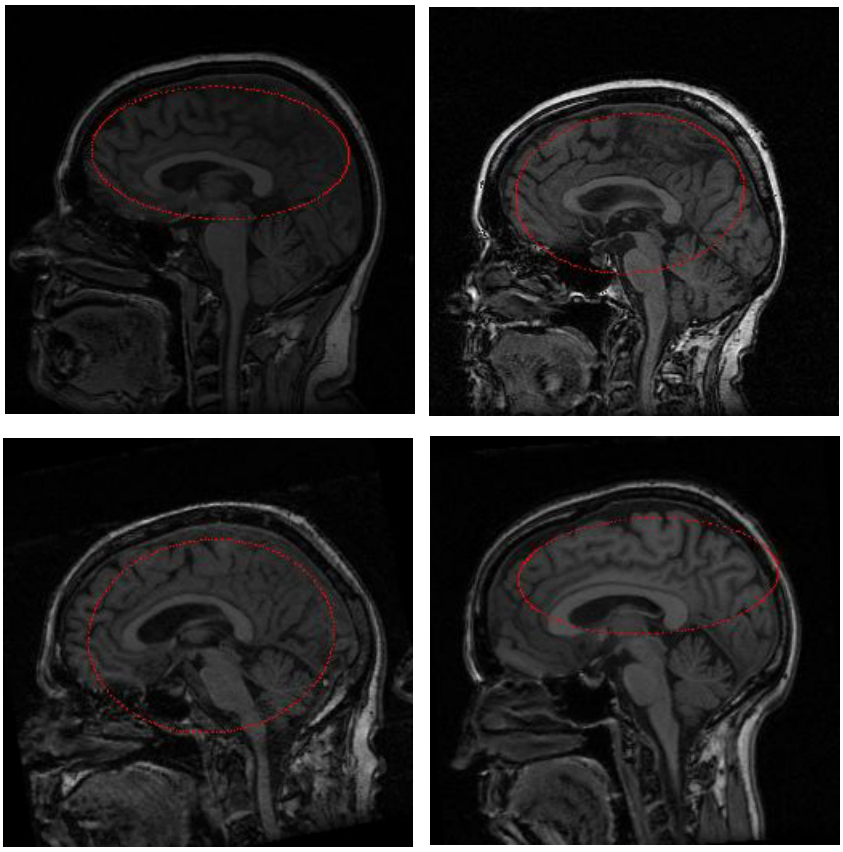


Figure 1. The initial ellipsoids intersected with the middle slices

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 (±1, 0, 0), (0,±1, 0), and (0, 0,±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.

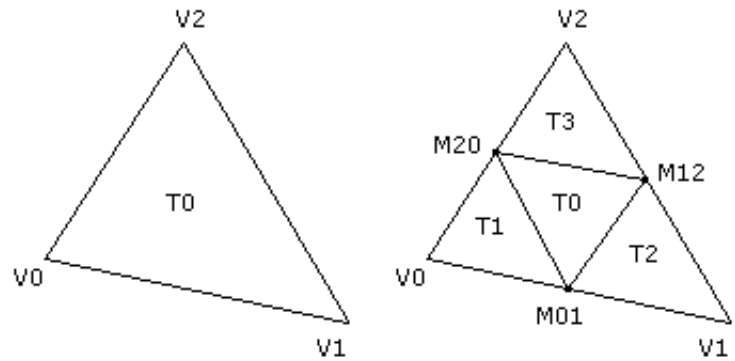


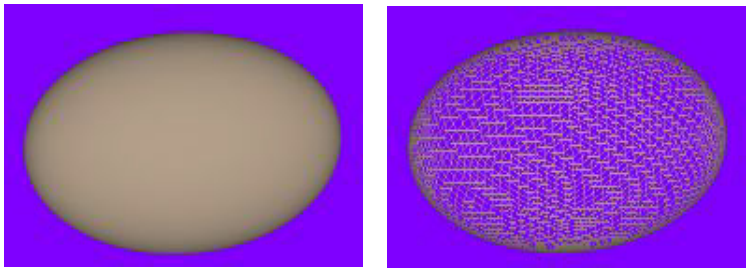
Figure 2. Subdivision of a triangle

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.



(A) Solid ellipsoid (B) Wireframe ellipsoid

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:

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

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 47) 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 κ 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,* σ, with a default value of 0.1 and use.

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

Moreover, σ 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.

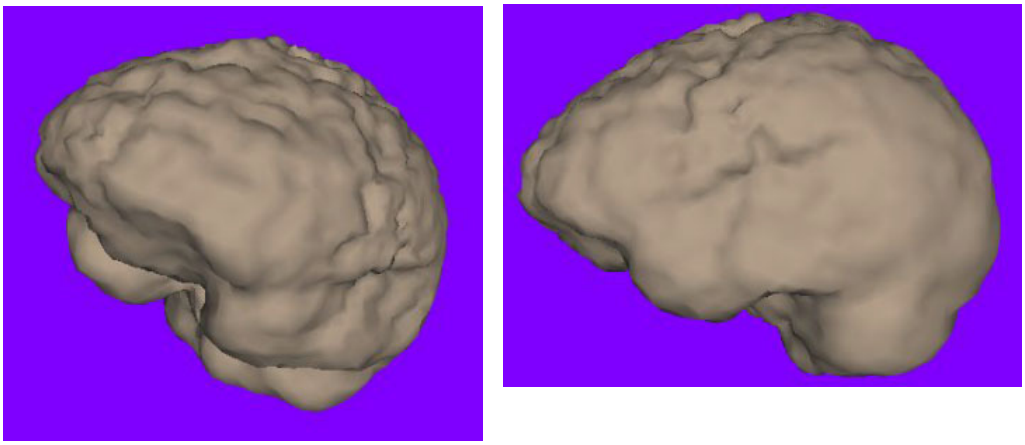


Figure 4. Two views of a brain surface mesh

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.

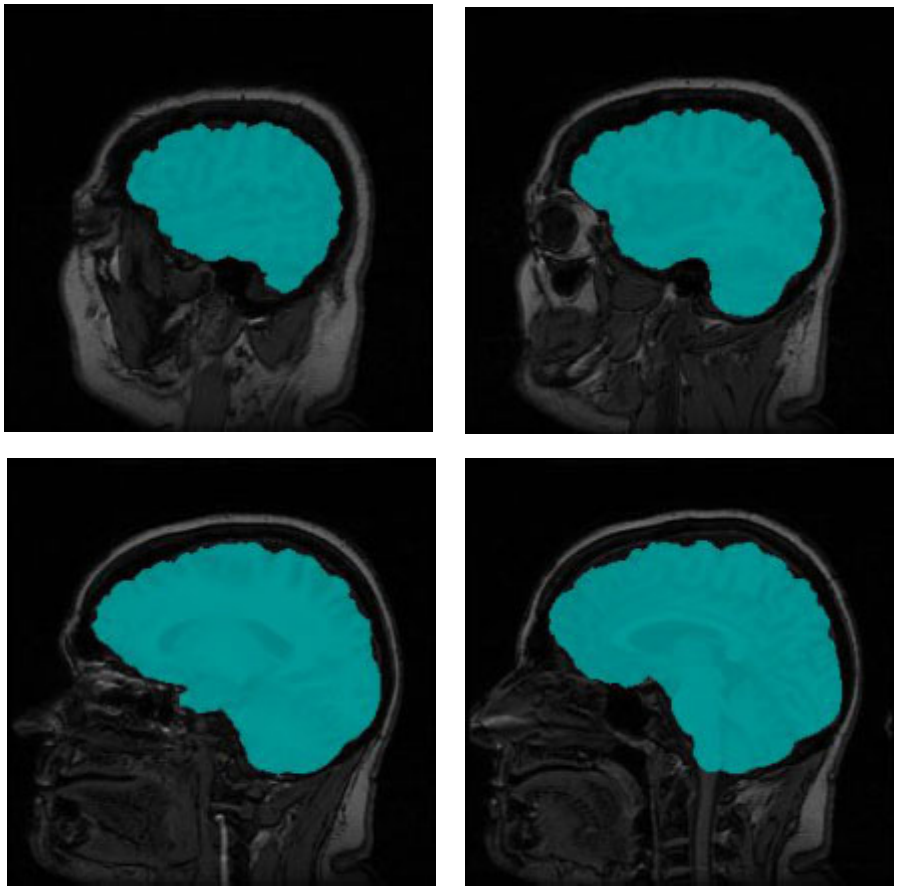


Figure 5. MRI slices with colored brain voxels

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

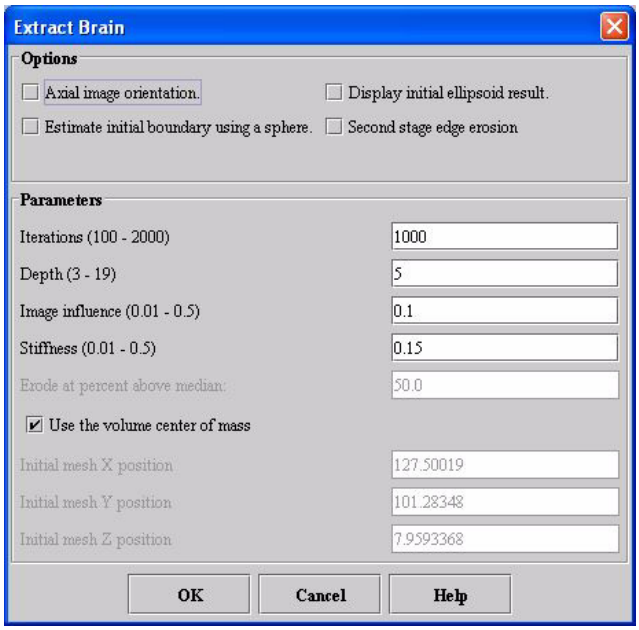
Refer to the following reference for more information about this algorithm:

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.



Axial image orientation	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.
Estimate initial boundary using a sphere	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.
Display initial ellipsoid result	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.
Second stage edge erosion	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.
Iterations	Specifies the number of times to run the algorithm when evolving the ellipsoid into the brain surface. The default is 1000.
Depth	Specifies the maximum depth inside the surface of the brain to use in sampling intensities. The default value is 5.
Image influence	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

Stiffness	Controls the stiffness of the mesh against motion in the surface normal direction. The default value is 0.15.
Erode at percent above median	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.
Use the volume center of mass	Contructs the mesh using the volume center of mass. When you select this check box, you *cannot* specify the Initial *X, Y,* or *Z* mesh positions.
Initial mesh *X* position	Constructs the mesh using the *X* 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.
Initial mesh *Y* position	Constructs the mesh using the *Y* 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.
Initial mesh *Z* position	Constructs the mesh using the *Z* 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.
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.

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 50;
- Step 2, Detecting edges in the image, refer to page 50;
- Step 3, Performing morphological erosions and brain isolation, see page 51;
- Step 4, Performing surface cleanup and image masking, refer to page 51.

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 7A shows the original image, and Figure 7B 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 7C).

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 7D). 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 7E).

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 7F).

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 7.

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 7 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.

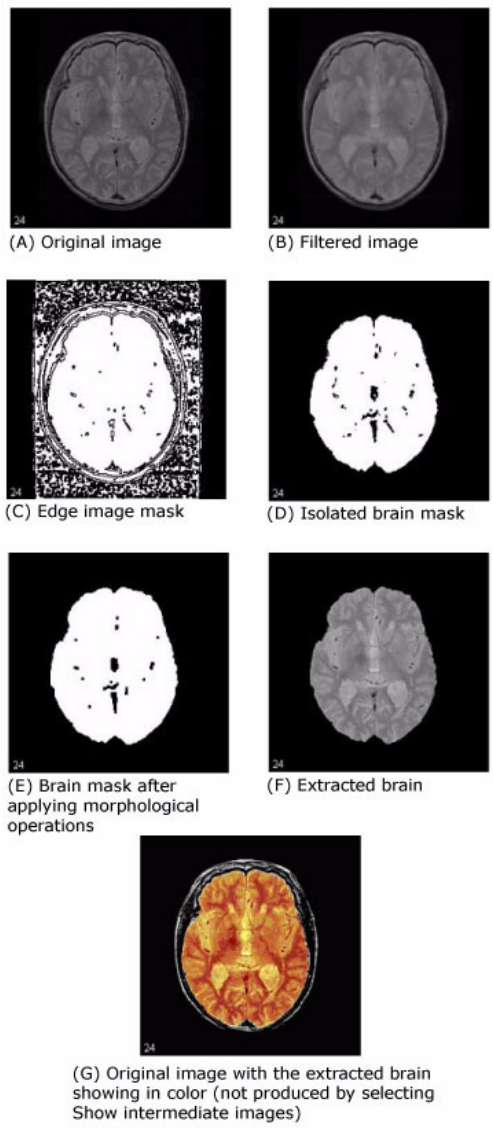


Figure 7. Examples of Extract Brain Surface (BSE) image processing

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 8 on page 54).
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.

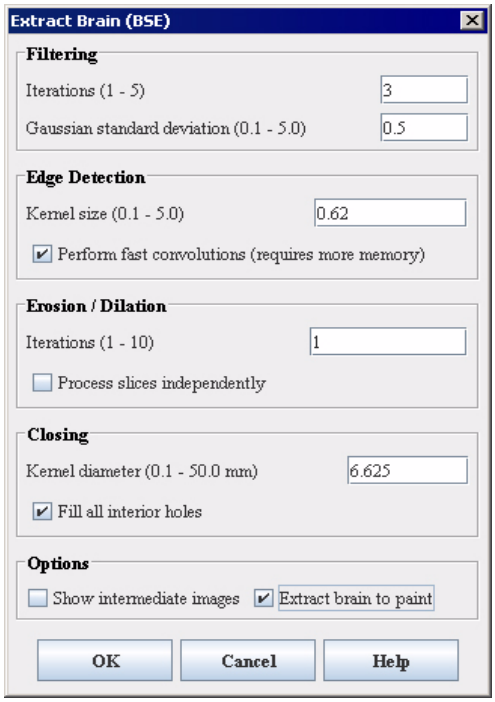
Filtering		
Iterations (1-5)	Specifies the number of regularized isotropic (nonlinear) diffusion filter passes to apply to the image. This parameter is used to find anatomical boundaries seperating the brain from the skull and tissues. For images with a lot of noise, increasing this parameter will smoth noisy regions while maintaining image boundaries.	
Gaussian standard deviation (0.1-5.0)	Specifies the standard deviation of the Gaussian filter used to regularize the image. A higher standard deviation gives preferebce to high-contrast edges for each voxel in the region.	
Edge Detection		
Kernel size (0.1-5.0)	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 kernal size will result in more edges.	
Perform fast convolutions (requires more memory)	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 8. Extract Brain Surface (BSE) algorithm dialog box

Erosion/Dilation	
Iterations (1-10)	Specifies the number of: • Erosions that should be applied to the edge image before the brain is isolated from the rest of the volume; • Dilations to perform afterward. 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.
Process slices independently	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.
Closing	
Kernel diameter (in mm) (0.1-50.0)	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.
Fill all interior holes	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.
Options	
Show intermediate images	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. For an image named *ImageName*, the debugging images displayed would include: • The filtered image (named *ImageName_filter*) • The edge image (named *ImageName_edge*) • The eroded edge image (named *ImageName_erode_brain*) • The isolated brain mask after erosion and dilation (named *ImageName_erode_brain_dilate*) • The brain mask after closing (named *ImageName_close*) • The closing image is shown before any interior mask holes are filled
Extract brain to paint	Paints the extracted brain onto the current image. See also Figure 9.

Figure 8. Extract Brain Surface (BSE) algorithm dialog box (continued)

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.

Figure 8. Extract Brain Surface (BSE) algorithm dialog box (continued)

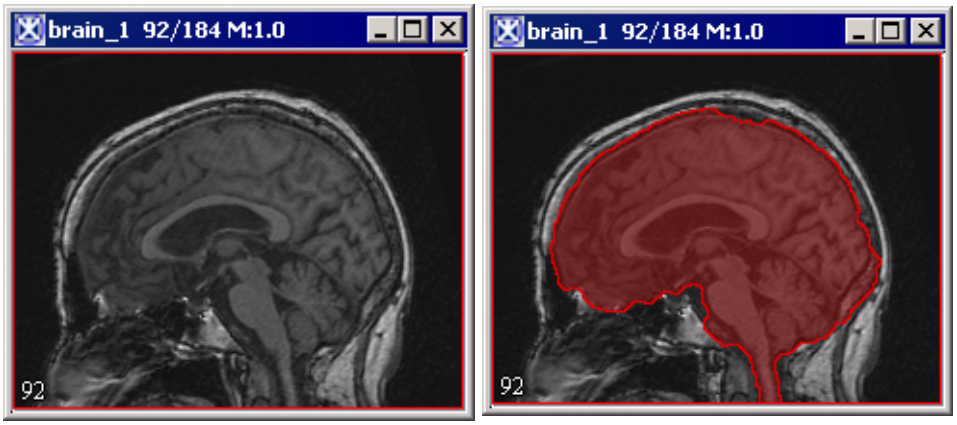


Figure 9. The Extract Brain to Paint option: on your left is the original image and on your right is the result image with the brain extracted to paint.

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 59 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 57;
2. Buffer the brain, refer to page 58;
3. Define the face, refer to page 58;
4. Extract the face, refer to page 58;
5. Smooth the face, refer to page 58.

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 59.

For more information concerning the mechanics of this portion of the algorithm, see references on page 59.

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 59.

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.

4 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.

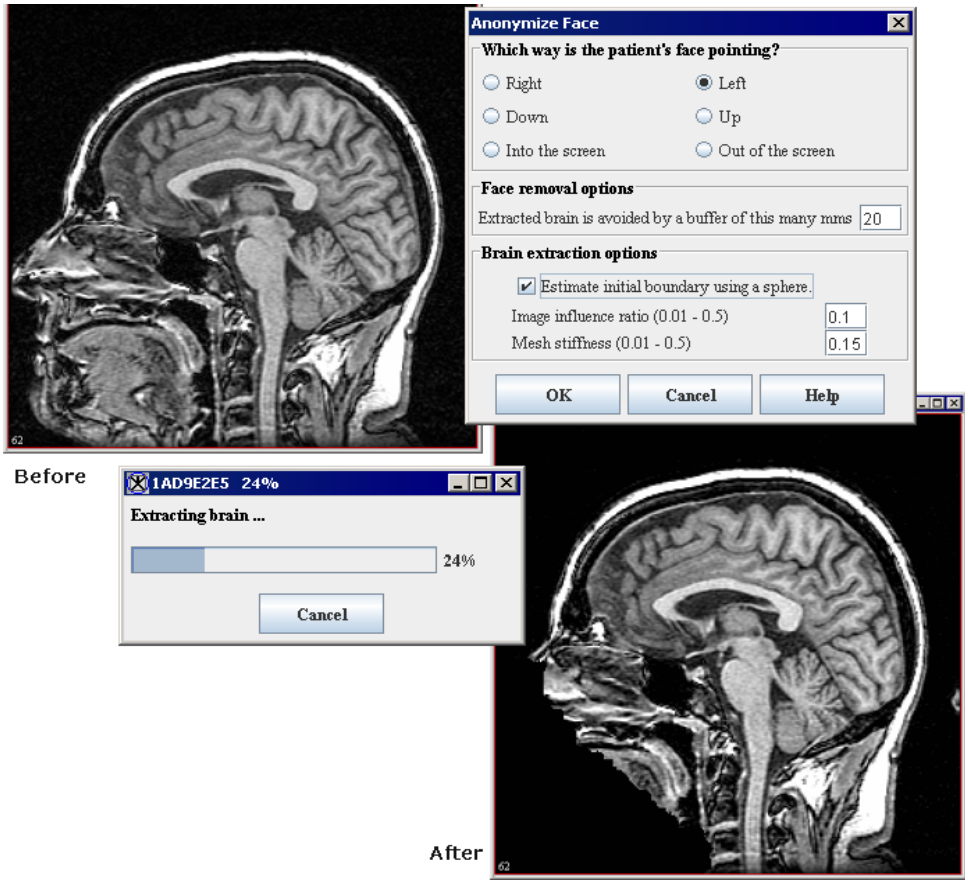


Figure 10. An example of the Face Anonymizer (BET) image processing. The image has been processed using default parameters

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.

Which way is the patient's face pointing?	Specifies the orientation of the image. MIPAV extrapolates this value from the image file information. Confirm that the orientation given is correct: **Right** – sagittal image with x-axis oriented posterior to anterior; **Left** – sagittal image with X-axis oriented anterior to posterior; **Down** – axial image with Y-axis oriented posterior to anterior; **Up** – axial image with y-axis oriented anterior to posterior; **Into the screen** – coronal image with Z-axis oriented posterior to anterior; **Out of the screen** – coronal image with Z-axis oriented anterior to posterior.
Face Removal Options	
Brain buffer	Guarantees that the extracted brain will be avoided by the specified number of millimeters. Refer to "Step 2: Buffer the brain" for more information.
Brain Extraction Options	
Initial shape	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.
Image influence ratio	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.
Mesh stiffness	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.
OK	Confirms the dialog settings and runs the algorithm.
Cancel	Disregards any changes that you made in this dialog box and closes this dialog box.
Help	Displays online help for this dialog box.

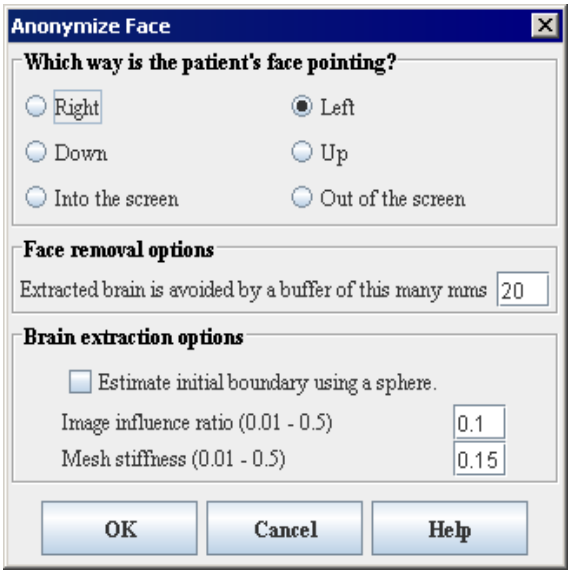


Figure 11. Face Anonymizer dialog box options

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 . . .	To . . .
Algorithms > Filters (frequency)	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.
Algorithms > FFT	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 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 12 shows the application of the FFT algorithm to a 2D image.

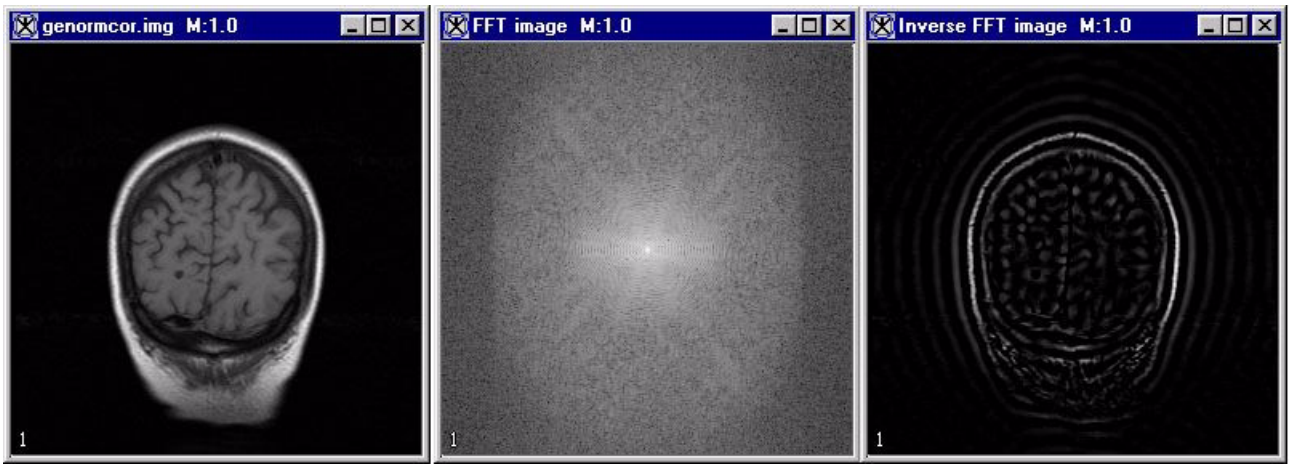


Figure 12. FFT algorithm processing

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:

$$X(\omega_1,\omega_2) = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} x(n_1,n_2) e^{-j\omega_1 n_1} e^{-j\omega_2 n_2}$$

$$x(n_1,n_2) = \frac{1}{(2\pi)^2} \int_{\omega_1=-\pi}^{\pi} \int_{\omega_2=-\pi}^{\pi} X(\omega_1,\omega_2) e^{-j\omega_1 n_2} e^{-j\omega_2 n_2} d\omega_1 d\omega_2$$

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

$$coefficient = \frac{1.0}{1.0 + \left(\frac{distsq}{(f1)}\right)^{2 \cdot butterworthorder}}$$

High-pass filter

$$coefficient = \frac{1.0}{1.0 + \left(\frac{(f1)}{distsq}\right)^{2 \cdot butterworthorder}}$$

Band-pass filter

$$width = f2 - f1, centersq = f1 \cdot f2$$

$$num = (distsq - centersq)^{2.0 butterworthorder}$$

$$coeff = \frac{num}{(num + (distsq - centersq)^{2 \cdot butterworthorder})}$$

Band-stop filter

$$width = f2 - f1, centersq = f1 \cdot f2$$

$$num = (\sqrt{distsq} \cdot width)^{2 \cdot butterworthorder}$$

$$coeff = \frac{num}{\left(num + (\sqrt{distsq} \cdot width)^{2 \cdot butterworthorder}\right)}$$

Gaussian low- and high-pass filters

The following equations are for the Gaussian low- and high-pass filters.

Gaussian low-pass (2D) filter

$$coeffientforgaussianhighpass = 1 - gaussianlowpass$$

Gaussian low-pass (3D) filter

$$coefficient = e^{-(x-xcenter)^2/(2f^2xcenter^2)} \cdot e^{-(y-ycenter)^2/(2f1^2ycenter^2)} \cdot e^{-(z-z\text{center})^2/(2f1^2z\text{center}^2)}$$

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 13).
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 log10 [1 + square root (real part**2 + imaginary part**2)] is shown. If the Display log magnitude is clear, square root (real part**2 + 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 13) 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.

Process each slice independently (2.5D)	Filters each slice of the dataset independently of adjacent slices.
Display log magnitude	Displays 1 plus the magnitude of the FFT.
Allow unequal FFT dimensions	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.
Windowed finite impulse response	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.
Crop image to save memory	Saves memory by assigning the edge pixels of the image a value of zero.
Convolution kernel diameter-odd	Limits the diameter of the convolution kernel to the size that you specify.
Gaussian filter	Applies the Gaussian filter to the image. The Gaussian filters only low-pass and high-pass filters are available.
Butterworth filter	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.

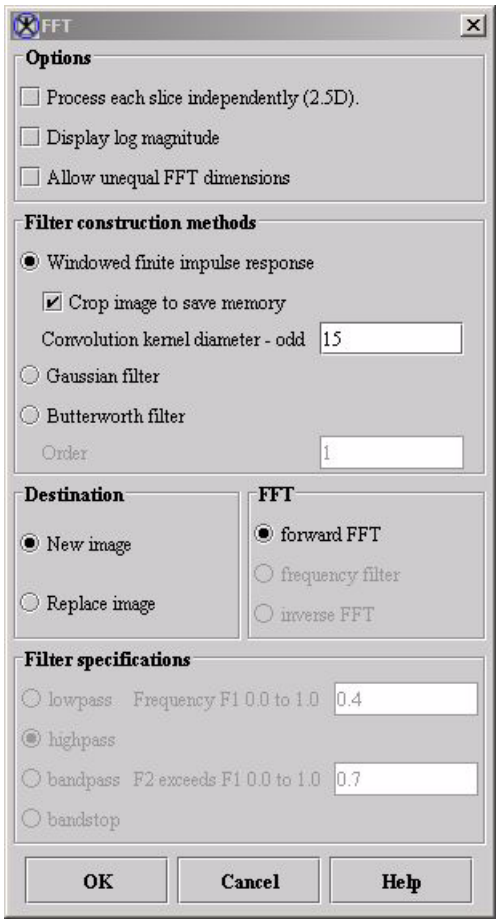


Figure 13. FFT dialog box

Order	Displays the order of the Butterworth filter.
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.
Forward FFT	Converts spatial image to a complex fourier transform image.
Frequency filter	Enhances or attenuates selected frequency components by multiplying the fourier transform image by the low-pass, high-pass, band-pass, or band-stop filters.
Lowpass	Suppresses frequencies above a specified threshold. Frequencies below the threshold are transmitted.
Highpass	Suppresses frequencies below a specified threshold. Frequencies above the threshold are transmitted.
Bandpass	Suppresses all frequencies except those in a specified band.
Bandstop	Suppresses a band of frequencies from being transmitted through the filter. Frequencies above and below the band are transmitted.
OK	Applies the FFT 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.
Help	Displays online help for this dialog box.

Figure 13. 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(n1,n2) real <- > X(k1,k2) = X*(-k1,-k2) for 2D and x(n1,n2,n3) real <-> X(k1,k2,k3) = X*(-k1,-k2,-k3). 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 13) 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 (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 >= 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 = 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 key < maxDistance:

$$M = (int)(Rd(S+512))$$

where M = maxDistance, Rd = radius, and S = 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 $(int)\left(\frac{i-1}{2}\right)$.

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.

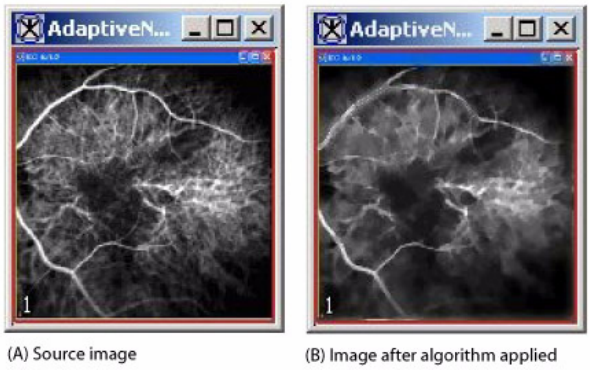


Figure 14. Adaptive Noise Reduction processing

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 *C*b at halv*e*d 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

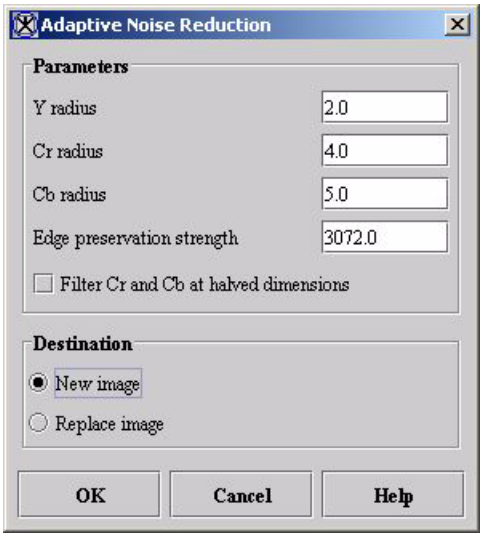
To run this algorithm, complete the following steps:

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.



Y radius	Specifies the local square used for *Y* space filtering, which is approximately $(2\text{YRadius}+1)(2\text{YRadius}+1)$ in size. As the radius increases, the algorithm removes more noise, but loses more detail. The default value is 2.0.
Cr radius	Specifies the local square used for *Cr* space filtering (only available for color), which is approximately in size. $(2\text{CrRadius}+1)(2\text{CrRadius}+1)$ As the radius increases, the algorithm removes more noise, but loses more detail. The default value is 4.0.
Cb radius	Specifies the local square used for *Cb* space filtering, which is approximately $(2\text{CbRadius}+1)(2\text{CbRadius}+1)$ in size. As the radius increases, the algorithm removes more noise but loses more detail. The default value is 5.0.
Edge preservation strength	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 $\left(\frac{12.0}{255.0}\right)(IMx - IMn)$ where *IMx* is image maximum and *IMn* is image minimum.
Filter Cr and Cb at halved dimensions	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.
New image	Shows the results of the algorithm in a new image window.
Replace image	Replaces the original source image with the results of the algorithm.
OK	Applies the algorithm according to the specifications in this dialog box.
Cancel	Disregards any changes you made in this dialog box and closes the dialog box.
Help	Displays online help for this dialog box.

Figure 15. 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 *range* = (int)(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 >= 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 $_{(int)}\frac{(i-1)}{2}$.

(A) Image before algorithm applied

(B) Image after algorithm applied

Figure 16. 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 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 shrunken 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 17) 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 18).

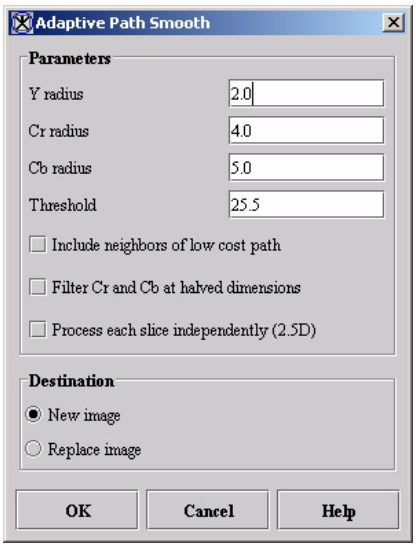
3 Complete the information in the dialog box.

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.

Radius	The local square used is approximately (2 x *Radius* + 1) times (2 x *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.
Threshold	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 x (image maximum - image minimum).
Include neighbors of low cost path	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 *not* selected.
Process each slice independently	Filters each slice of the dataset independently of adjacent slices. The default value is *not* selected. (This check box only appears for 3D images.)
New image	Shows the results of the algorithm in a new image window (default choice).
Replace image	Replaces the current active image with the new image produced by the algorithm.
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 this dialog box.
Help	Displays online help for this dialog box.

Figure 17. Adaptive Path Smooth dialog box for grayscale images



Y radius	The local square used for *Y* space is approximately (2 x *YRadius* + 1) times (2 x *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.
Cr radius	The local square used for *Cr* space filtering is approximately (2 x *CrR* + 1) x (2 x *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.
Cb radius	The local square used for *Cb* space filtering is approximately (2 x *CbR* + 1) x (2 x *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.
Threshold	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 x (max (red max, green max, blue max) - min(red min, green min, blue min)).
Include neighbors of low cost path	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 *not* selected.
Filter Cr and Cb at halved dimensions	When this option is selected, the algorithm runs faster but loses some detail. The default value is *not* selected.
Process each slide independently	Filters each slice of the dataset independently of adjacent slices. The default value is *not* selected. (This check box only appears for 3D images.)
New image	Shows the results of the algorithm in a new image window (default choice).
Replace image	Replaces the current active image with the new image produced by the algorithm.
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 this dialog box.
Help	Displays online help for this dialog box.

Figure 18. 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.

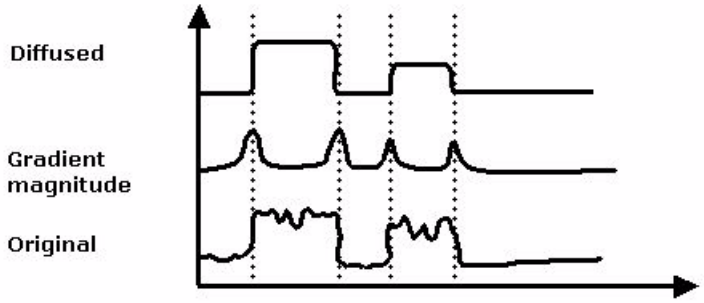


Figure 19. A 1D example of anisotropic diffusion

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:

$$I_t = div(c(\vec{x},t)\nabla I) = c(\vec{x},t)\nabla^2 I + \nabla c \bullet \nabla I$$

$$I(\vec{x},0) = I_0(\vec{x})$$

$$c(\vec{x},t) = e^{-\left(\frac{\|\nabla I\|}{k}\right)^2} \quad \text{or}$$

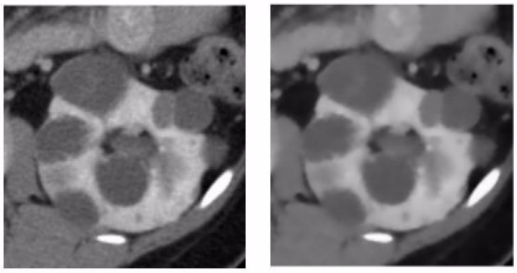
$$c(\vec{x},t) = \frac{1}{1 + \left(\frac{\|\nabla I\|}{k}\right)^2}$$

where $c(\vec{x},t)$ is the variable conductance term. Typically, a function of the gradient magnitude $\|\nabla I\|$, if $c(\vec{x},t)$ 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 20 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.



(A) Image before Anisotropic Diffusion algorithm applied

(B) Image after Anisotropic Diffusion algorithm applied

Figure 20. An image before and after applying the Anisotropic Diffusion algorithm

SPECIAL NOTES

Refer to the section on Filters (Spatial): Nonlinear Noise Reduction.

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 21) 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.

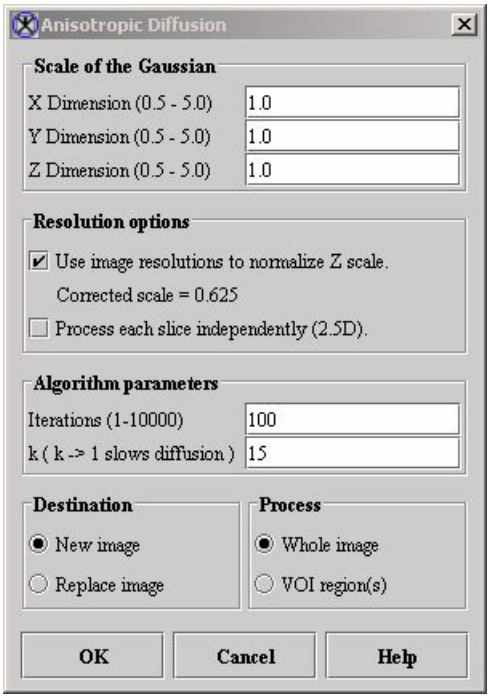
5. Click OK. The algorithm begins to run.

   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.

X dimension (0.5 - 5.0)*	Indicates the standard deviation (SD) of Gaussian in the *X* direction.
Y dimension (0.5 - 5.0)*	Indicates the SD of Gaussian in the *Y* direction.
Z dimension (0.5 - 5.0)*	Indicates the SD of Gaussian in the *Z* direction.
Use image resolutions to normalize Z scale	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.
Process each slice independently (2.5D)	Blurs each slice of the dataset independently.
Iterations (1-1000)	Specifies the number of times the process is run.
k (k > 1 slows diffusion)	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.
New image	Shows the results 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 regions	Applies the algorithm to the volumes (regions) delineated by the VOIs.
OK	Applies the algorithm according to the specifications in this dialog box.
Cancel	Disregards any changes you made in this dialog box and closes the dialog box.
Help	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 21. 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 96). 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.

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

$$\partial_t I = div(D\nabla I)$$

where

$\partial_t I$ =The time derivative of the image

div=The divergence operator

D =The diffusion tensor (which may be a scalar value)

∇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:

$$\partial_t I = div(D(|\nabla I|^2)\nabla I)$$

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:

$$\partial_t I = div\left(D\left(\left|\nabla I_\sigma\right|^2\right)\nabla I_\sigma\right)$$

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 $\nu_1, \nu_2, \ldots, \nu_n$ of the diffusivity tensor so that $\nu_1 || \nabla I_\sigma$ and $\nu_2 \perp \nabla I_\sigma, \ldots, V_n \perp \nabla I_\sigma$.

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_ρ is a Gaussian kernel of standard deviation ρ, 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 $v_1 || \nabla I_\sigma$ and $v_2 \perp \nabla I_\sigma, \ldots, V_n \perp \nabla I_\sigma$.

Figure 23 shows the results of applying the coherence-enhancing anisotropic diffusion filter to an example MR knee image.

Figure 22. 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.

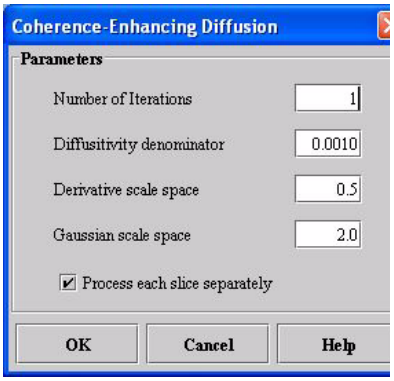
Weickert, Joachim. "Nonlinear Diffusion Filtering," in *Handbook of Computer Vision and Applications,* Volume 2, eds. Bernd Jahne, Horst Haussecker, and Peter Geissler. (Academic Press, April 1999), 423–450.

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 23).



Number of iterations	Specifies the number of iterations, or number of times, to apply the algorithm to the image.
Diffusitivity denominator	Specifies a factor that controls the diffusion elongation.
Derivative scale space	Specifies the standard deviation of the Gaussian kernel that is used for regularizing the derivative operations.
Gaussian scale space	Specifies the standard deviation of the Gaussian filter applied to the individual components of the diffusivity tensor.
Process each slice separately	Applies the algorithm to each slice individually. By default, this option is selected.
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.
Help	Displays online help for this dialog box.

Figure 23. 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.

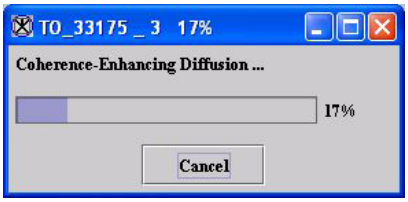


Figure 24. Status message that appear when the Coherence-Enhancing Diffusion algorithm is running

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:

$$G(\vec{x}, \sigma) = \frac{1}{(2\pi\sigma)^{n/2}} e - (\vec{x} \cdot \vec{x})/(2\sigma^2)$$

EQUATION 1

where σ (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 $I \cdot \Re n \rightarrow \Re$ 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:nx $\Re + \rightarrow \Re$ 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.

$$G(x,\sigma) = \frac{1}{2\pi\sigma^{2^{\frac{1}{2}}}} e^{-\frac{x^2}{2\sigma^2}}$$

$$1n0.5 \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} = 1n\left(\frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} e^{-\frac{(x^2)}{(2\sigma^2)}}\right)$$

$$1n0.5 + 1n\left(\frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}}\right) = 1n\left(\frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}}\right) + 1n\left(e^{-\frac{(x^2)}{(2\sigma^2)}}\right)$$

$$-0.693 = -\frac{x^2}{2\sigma^2}$$

$$1.18\sigma = x$$

Figure 26 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.

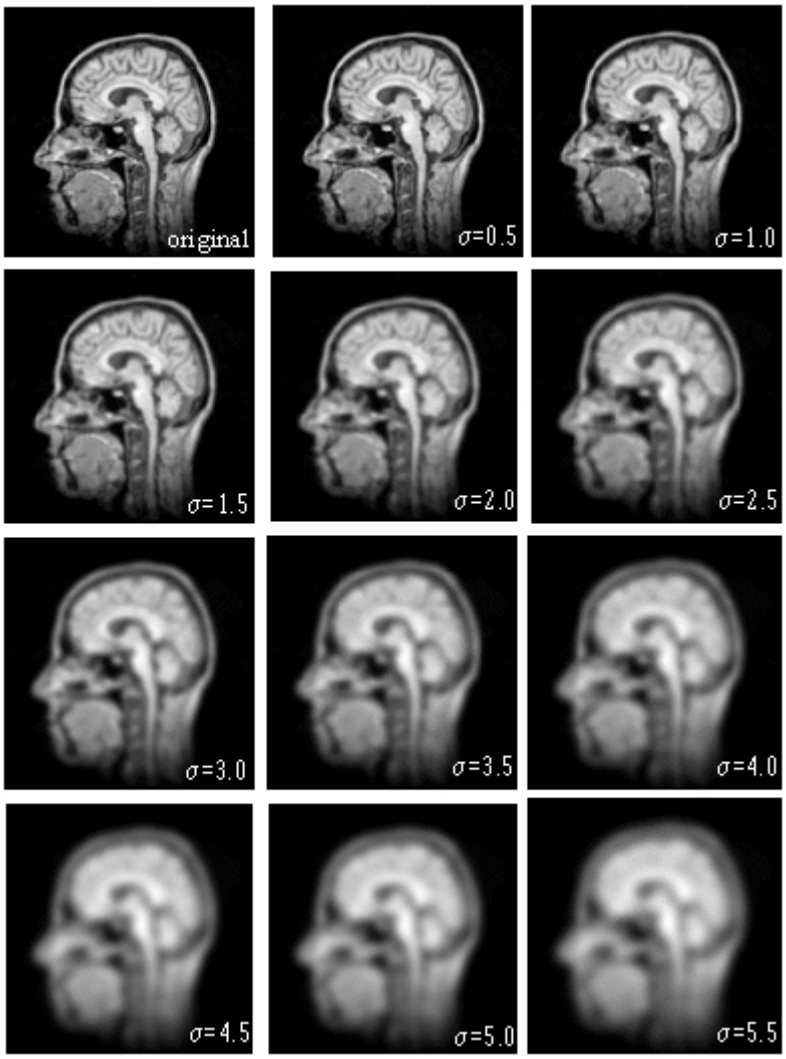


Figure 25. Gaussian Blur algorithm processing

Figure 25 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 σ from 0.5 to 5.5 in steps of 0.5 blurs the other images (from left to right and top to bottom).

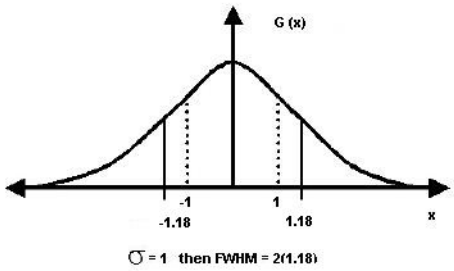


Figure 26. Gaussian function

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 25).

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

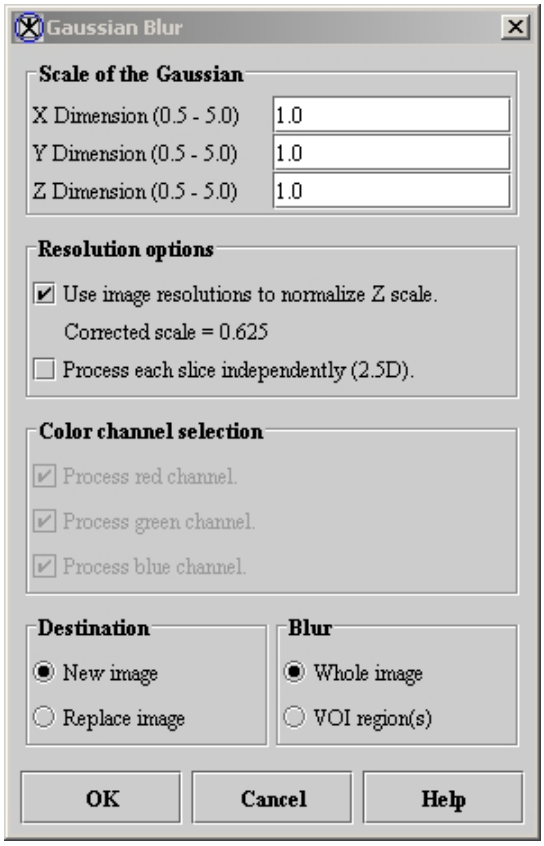
To run this algorithm, complete the following steps:

1. Select Algorithms > Filter > Gaussian Blur. The Gaussian Blur dialog box opens (Figure 27).
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.

X Dimension	Indicates the standard deviation (SD) of Gaussian in the *X* direction.
Y Dimension	Indicates the SD of Gaussian in the *Y* direction.
Z Dimension	Indicates the SD of Gaussian in the *Z* direction.
Use image resolutions to normalize Z scale	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.
Process each slice independently (2.5D)	Blurs each slice of the dataset independently of adjacent slices.
Process red channel	Applies the algorithm to the red channel only.
Process green channel	Applies the algorithm to the green channel only.
Process blue channel	Applies the algorithm to the blue channel only.



New image	Shows the results 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 volumes (regions) delineated by the VOIs.
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.
Help	Displays online help for this dialog box.

Figure 27. 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 (*Gx*, *Gy*, and *Gz* [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 28 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.

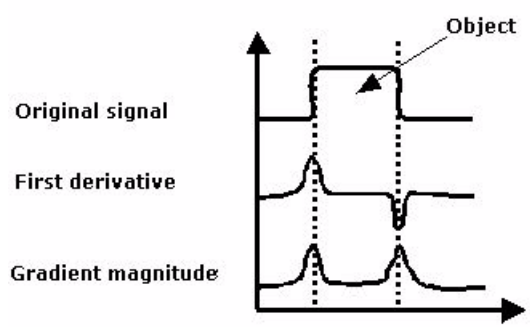


Figure 28. Edge detection using derivative operators

At location (*x*,*y*) of a 2D image, the gradient vector is:

$$\nabla(x,y) = \begin{bmatrix} I_x \\ I_y \end{bmatrix} = \begin{bmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{bmatrix}$$

and points in the direction of the maximum rate of change of the image at that point.

The gradient magnitude is defined as:

$$mag(\nabla I) = \left[I_x^2 + I_y^2 \right]^{1/2}$$

and is the maximum rate of increase of $I(x,y)$ in the direction of ∇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:

$$I_x(x,y,\sigma) = \frac{\partial}{\partial x} G(x,y,\sigma) \otimes I(x,y)$$

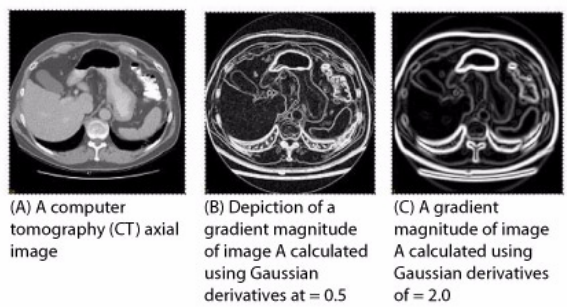


Figure 29. Gradient Magnitude algorithm processing

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.

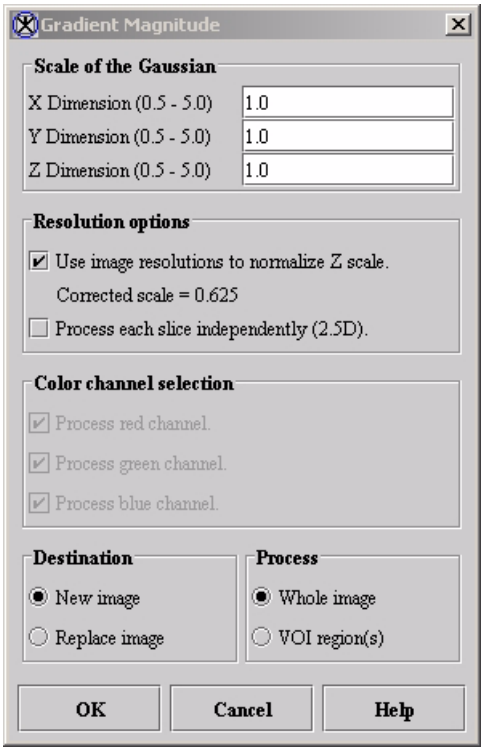
Applying the Gradient Magnitude algorithm

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 results appear in a new window or replace the image to which the algorithm was applied.



X Dimension	Indicates the standard deviation (SD) of Gaussian in the *X* direction.
Y Dimension	Indicates the SD of Gaussian in the *Y* direction.
Z Dimension	Indicates the SD of Gaussian in the *Z* direction.
Use image resolutions to normalize Z scale	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.
Process each slice independently (2.5D)	Blurs each slice of the dataset independently.
Process red channel	Applies algorithm to the red channel only.
Process green channel	Applies algorithm to the green channel only.
Process blue channel	Applies algorithm to the blue channel only.
New image	Shows the results 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 volumes (regions) delineated by the VOIs.
OK	Applies the Gradient Magnitude 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.
Help	Displays online help for this dialog box.

Figure 30. Gradient Magnitude dialog box

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 (*Gxx*, *Gyy*, and *Gzz* [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.

Background

Figure 31 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.

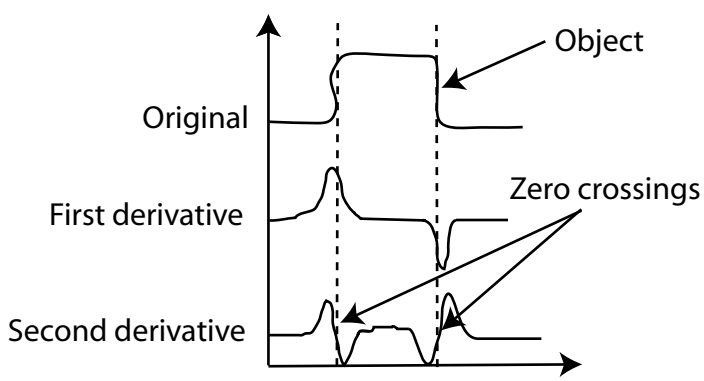


Figure 31. Edge detection using laplacian operators

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:

$$\nabla^2 I(x,y) = \left(\frac{\partial^2 G}{\partial x^2} + \frac{\partial^2 G}{\partial y^2} \right) \otimes I(x,y)$$

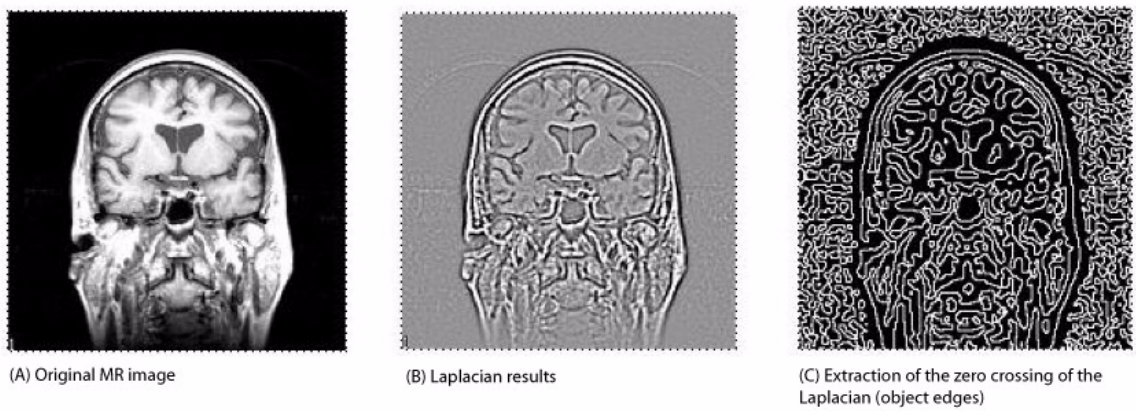


Figure 32. (A) Original MR image; (B) laplacian results; and (C) extraction of the zero crossing of the laplacian (object edges)

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 31).

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).

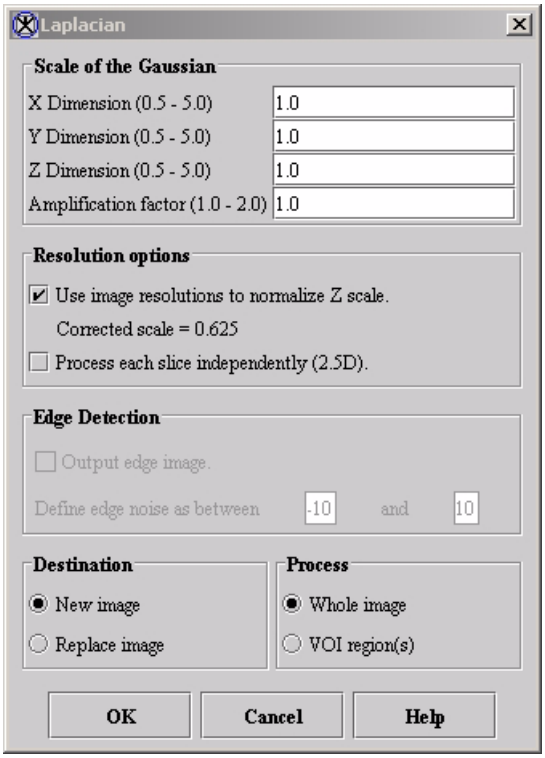
Applying the Laplacian algorithm

To run this algorithm, complete the following steps:

1. Select Algorithms > Filter > Laplacian. The Laplacian dialog box opens (Figure 33).
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.



X Dimension	Specifies the standard deviation (SD) of Gaussian in the *X* direction.
Y Dimension	Specifies the SD of Gaussian in the *Y* direction.
Z Dimension	Specifies the SD of Gaussian in the *Z* direction.
Amplification factor	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.
Use image resolutions to normalize Z scale	Normalizes the Gaussian to compensate for the difference if the voxel resolution is less than the voxel resolution inplane.
Process each edge independently (2.5D)	Blurs each slice of the dataset independently.
Output edge image	Produces a binary image that represents the edges defined by the 0 crossings of the laplacian
Threshold edge noise between <_> and <_>	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.
New image	Shows the results 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 volumes (regions) delineated by the VOIs.
OK	Applies the laplacian 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.
Help	Displays online help for this dialog box.

Figure 33. Laplacian dialog box

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 34 and Figure 35), 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 35), 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 35 for examples of slice and volume averaging.

Note that the term *average* does not always represent the mean. If 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.

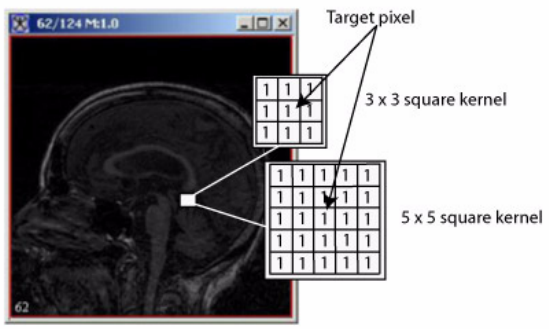


Figure 34. Slice averaging

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 x7, 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

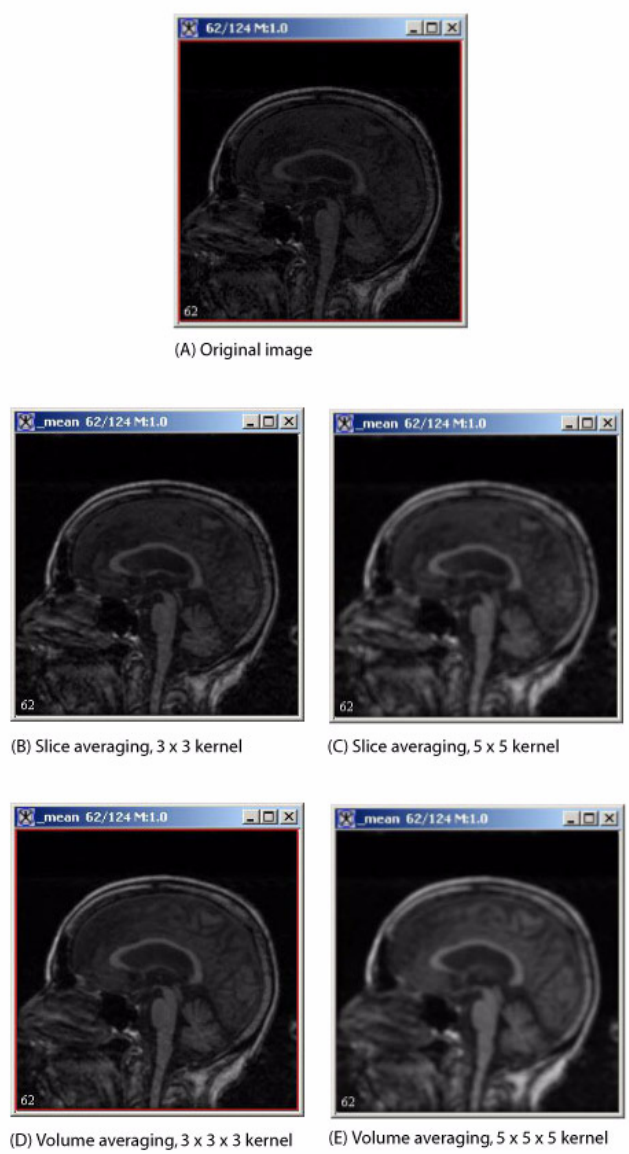


Figure 35. Image processing using the Mean algorithm

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 36) 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.

Kernel size	Specifies the size of the *k* x *k* neighborhood that extends in all directions by (*k* - 1)/2 from the center pixel.
Red channel	Applies the algorithm to the red channel.
Green channel	Applies the algorithm to the green channel.
Blue channel	Applies the algorithm to the blue channel.
New image	Shows the results of the algorithm in a new image window.
Replace image	Replaces the current active image with the newly calculated mean image.
Whole image	Applies the algorithm to the whole image.
VOI regions	Applies the algorithm to the volumes (regions) delineated by the VOIs.
Apply slice kernel	Applies the kernel to only the specific slice in the dataset
Apply volume kernel	Applies the kernel to all of the slices—that is, the entire volume—in the dataset.
OK	Applies the mean 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.
Help	Displays online help for this dialog box.

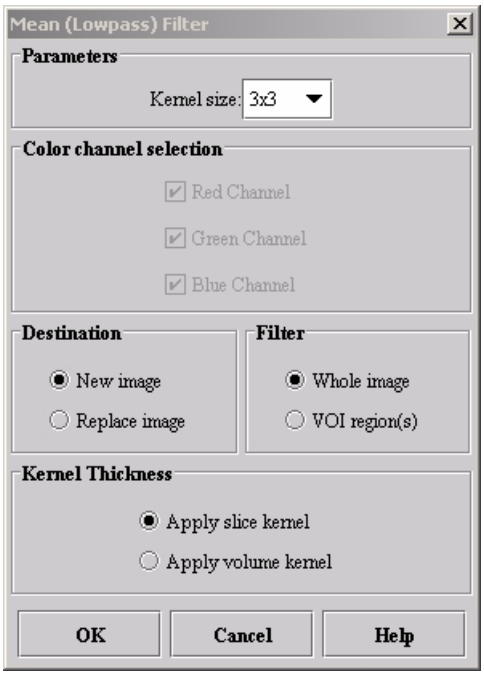


Figure 36. Mean Filter dialog box

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.

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 37). 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.

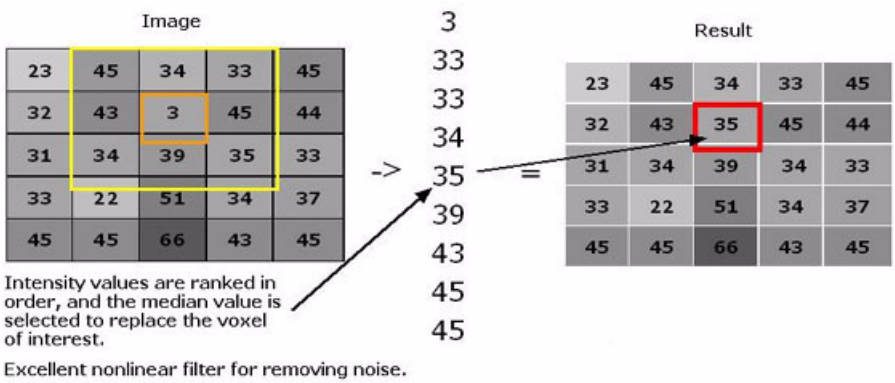


Figure 37. An example of median filter processing using a square 3 x 3 kernel

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 38 illustrates how the median filter algorithm is used to remove shot noise.

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.

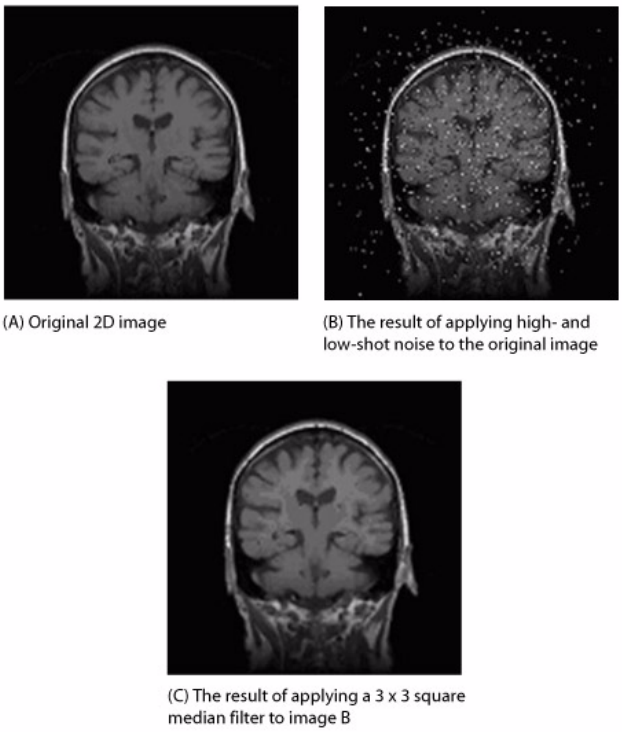


Figure 38. Removing shot noise with the Median Filter algorithm

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.

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.

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 39) 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.

Number of iterations	Indicates the number of times to filter the image with a median filter of the selected deviation, kernel size, and kernel shape.
Maximum standard deviation	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.
Kernel size	A k x k region selects a neighborhood that extends in all directions by $(k-1)/2$, placing the target pixel in the center of the kernel.
Kernel shape	Selects which pixels in the kernel size box are included as a part of the median's neighborhood.

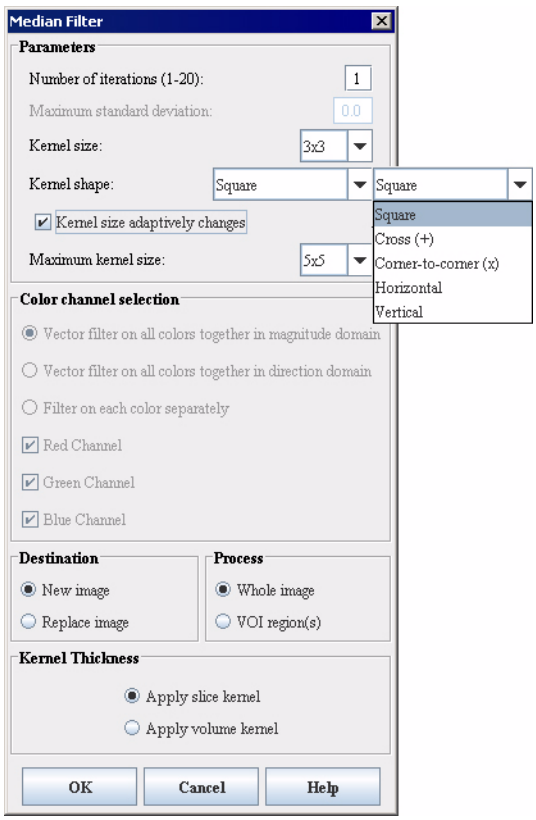


Figure 39. Median Filter dialog box

Kernel size adaptively changes	This checkbox indicates whether median filter should be run via adaptive filtering. Adaptive filtering has 3 main purposes: 1. To remove salt-and-pepper (impulse) noise; 2. To prove smoothing of other noise that may not be impulsive; 3. And to reduce distortion, such as excessive thinning or thickening of object borders.
Maximum Kernel Size	This is the maximum kernel size that the window size can get increased to under the algorithm.
Red channel	Filters the red channel (for use with RGB images).
Green channel	Filters the green channel (for use with RGB images).
Blue channel	Filters the blue change (for use with RGB images).
New image	Indicates where the results of the algorithm appear. If you select this option, the results appear in a new image window.
Replace image	Indicates where the results of the algorithm appear. If you select this option, the results replace the current active image.
Whole image	Applies the algorithm to the whole image.
VOI region(s)	Applies the algorithm to the volumes (regions) delineated by the VOIs.
Apply slice kernel	Filters each slice of the image independently of other slices, finding neighboring pixels from the slice.
Apply volume kernel	Gathers pixel values from the slice as well as from neighboring slices.
OK	Applies the Median 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.
Help	Displays online help for this dialog box.

Figure 39. 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 algorithms 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 40 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.

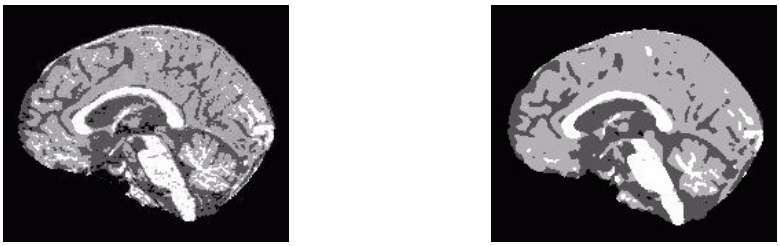


Figure 40. Examples of (A) a segmented image and (B) the results of applying the Mode algorithm to the image

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

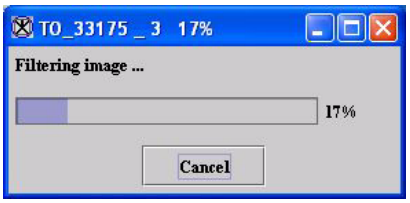
This algorithm was developed by members of the MIPAV team. Mode filtering is similar to median filtering, and to our knowledge there is not a significant amount of literature about the topic.

Applying the Mode algorithm

To run this algorithm, complete the following steps:

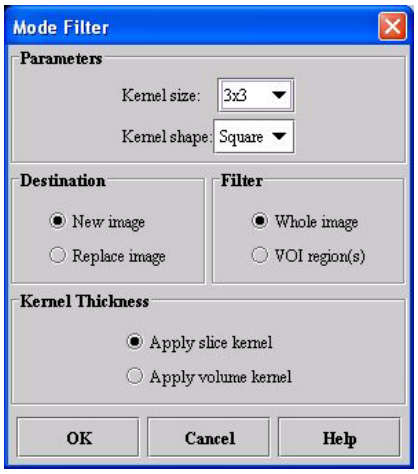
1. Open an image.
2. Select Algorithms > Filters (Spatial) > Mode. The Mode Filter dialog box (Figure 41) opens.
3. Complete the fields in the dialog box.
4. Click OK when complete. The algorithm begins to run.

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.



Kernel size	Specifies the size of the local neighborhood when performing mode filtering. A *k* x *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 x 3, 5 x 5, 7 x 7, and so on.
Kernel shape	Selects which pixels in the kernel size box are included as a part of the mode's neighborhood.
New image	Indicates where the results of the algorithm appear. If you select this option, the results appear in a new image window.
Replace image	Indicates where the results of the algorithm appear. If you select this option, the results replace the current active image.
Whole image	Applies the algorithm to the whole image.
VOI region(s)	Applies the algorithm to the volumes (regions) delineated by the VOIs.
Apply slice kernel	Filters each slice of the image independently of other slices, finding neighboring pixels from the slice.
Apply volume kernel	Gathers pixel values from the slice as well as from neighboring slices.

Figure 41. Mode Filter dialog box

OK	Applies the Mode filtering 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.
Help	Displays online help for this dialog box.

Figure 41. Mode Filter dialog box (continued)

Filters (Spatial): Nonlinear Noise Reduction

This algorithm provides a port of the SUSAN (Smallest Univalue Segment Assimilating Nucleus) low-level image processing program (see "References" on page 131.)

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.

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 *univalue segment assimilating nucleus* (USAN), that has the same or similar brightness as the nucleus. The SUSAN filter takes an average of the pixels in the USAN and then uses a Gaussian in both the brightness and spatial domains.

Table 2 explains the possible responses that the SUSAN filter provides if it cannot determine the neighborhood and based on whether you selected the option of using the median.

Table 2. 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.
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 42 for an example of image processing using the Nonlinear Noise Reduction algorithm.

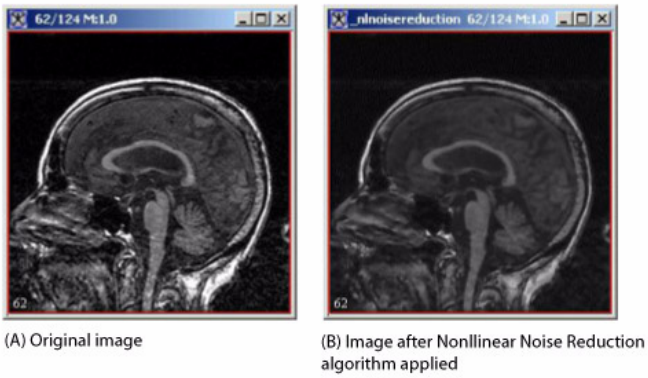


Figure 42. Image processing using the Nonlinear Noise Reduction algorithm

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.

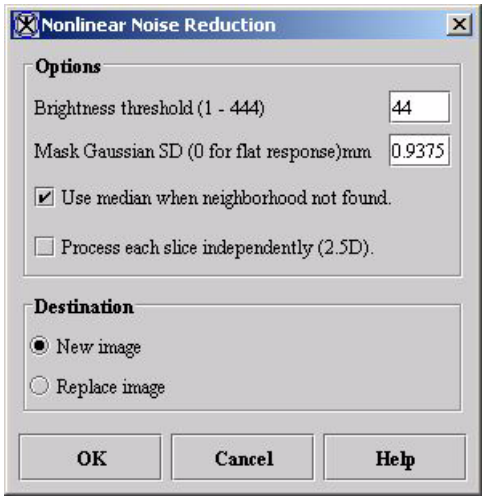
Applying the Nonlinear Noise Reduction algorithm

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 43) 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.



Brightness threshold	Blurs edges of contrast smaller than this threshold. Reducing the brightness threshold gives less smoothing. The default value is 0.1 * (image maximum - image minimum).
Mask Gaussian SD (0 for flat response) mm	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.
Use median when neighborhood not found	Refer to Table 2 on page 129 for an explanation. By default, this check box is selected.
Process each slice independently	Smooths each slice of the dataset independently (applies only to 3D images).
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.

Figure 43. Nonlinear Noise Reduction dialog box

Cancel	Disregards any changes that you made in the dialog box and closes the dialog box.
Help	Displays online help for this dialog box.

Figure 43. 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.

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.

$$I_u = \sin\alpha I_x - \cos\alpha I_y$$
$$I_v = \cos\alpha I_x + \sin\alpha I_y$$

$$\cos\alpha = \frac{I_x}{\sqrt{I_x^2 + I_y^2}} \text{ evaluated at } P_i$$

$$\sin\alpha = \frac{I_y}{\sqrt{I_x^2 + I_y^2}} \text{ evaluated at } P_i$$

$$I_u = \frac{(I_x I_y - I_x I_y)}{\sqrt{I_x^2 + I_y^2}} = 0$$

$$I_v = \frac{\left(I_x^2 + I_y^2\right)}{\sqrt{I_x^2 + I_y^2}} = \sqrt{I_x^2 + I_y^2}$$

$$I_v^2 = I_x^2 + I_y^2$$

$$I_v^2 I_{vv} = I_x^2 I_{xx} + 2 I_x I_y I_{xy} + I_y^2 I_{yy}$$

$$I_v^3 I_{vvv} = I_x^3 I_{xxx} + 3 I_x^2 I_y I_{xxy} + 3 I_x I_y^2 I_{xyy} + I_y^3 I_{yyy}$$

$$I_{vv} = (\cos\alpha I_x + \sin\alpha I_y)(\cos\alpha I_x + \sin\alpha I_y)$$

$$= (\cos\alpha)^2 I_{xx} + 2\cos\alpha\sin\alpha I_{xy} + (\sin\alpha)^2 I_{yy}$$

$$= \frac{\left(I_x^2 I_{xx} + 2I_x I_y I_{xy} + I_y^2 I_{yy}\right)}{\left(I_x^2 + I_y^2\right)}$$

$$I_{vvv} = \left((\cos\alpha)^2 I_{xx} + 2\cos\alpha\sin\alpha I_{xy} + (\sin\alpha)^2 I_{yy}\right)\cos\alpha I_x + \sin\alpha I_y = (\cos\alpha)^3 I_{xxx}$$

$$+ 3(\cos\alpha)^2\sin\alpha I_{xxy} + 3\cos\alpha(\sin\alpha)^2 I_{xyy} + (\sin\alpha)^3 I_{yyy}$$

$$= \frac{I_x^3 I_{xxx} + 3I_x^2 I_y I_{xxy} + 3I_x I_y^2 I_{xyy} + I_y^3 I_{yyy}}{\left(I_x^2 + I_y^2\right)^{1.5}}$$

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:

$$I_{vv} = 0$$
$$I_{vvv} < 0$$

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:

$$I_v^2 I_{vv} = I_x^2 I_{xx} + 2 I_x I_y I_{xy} + I_y^2 I_{yy}$$

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/(image maximum – 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 >= low threshold and <= 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.

$$I_v = \frac{\left(I_x^2 + I_y{}^2 + I_z{}^2\right)}{\sqrt{I_x^2 + I_y^2 + I_z^2}} = \sqrt{I_x{}^2 + I_y{}^2 + I^2{}_z}$$

$$I_v^2 = I_x^2 + I_y{}^2 + I_z{}^2$$

The nonmaximum suppression is:

$$\cos\alpha = \frac{I_x}{\sqrt{I_x^2 + I_y^2}} \text{ evaluated at } P_i$$

$$\sin\alpha = \frac{I_y}{\sqrt{I_x^2 + I_y^2}} \text{ evaluated at } P_i$$

$$\cos\beta = \frac{I_z}{\sqrt{I_x^2 + I_y^2 + I_z^2}} \text{ evaluated at } P_i$$

$$\sin\beta = \frac{\sqrt{I_x^2 + I_y^2}}{\sqrt{I_x^2 + I_y^2 + I_z^2}} \text{ evaluated at } P_i$$

$$I_v^3 I_{vvv} = I_x^3 I_{xxx} + 3I_x^2 I_y I_{xxy} + 3I_x^2 I_z I_{xxz} + 3I_x I_y^2 I_{xyy} + 6I_x I_y I_z I_{xyz} + 3I_x I_z^2 I_{xzz} + I_y^3 I_{yyy} + 3I_y^2 I_z I_{yyz} + 3I_y I_z^2 I_{yzz} + I_z^3 I_{zzz}$$

$$I_v^2 I_{vv} = I_x^2 I_{xx} + 2I_x I_y I_{xy} + 2I_x I_z + I_y^2 I_{yy} + 2I_y I_z I_{yz} + I_z^2 I_{zz}$$

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.

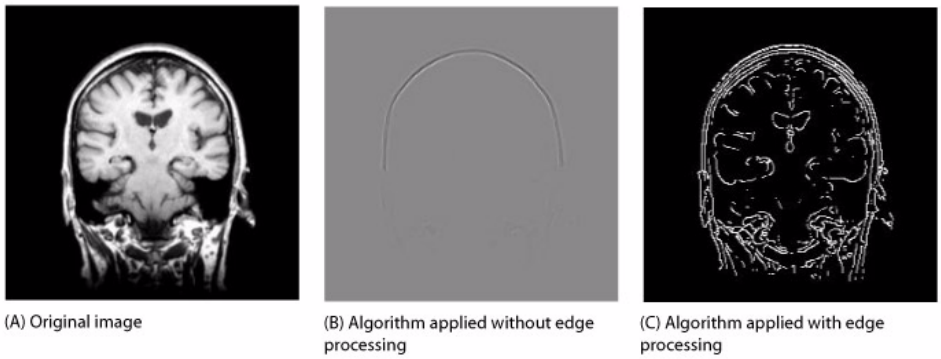


Figure 44. Examples of Nonmaximum Suppression processing

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

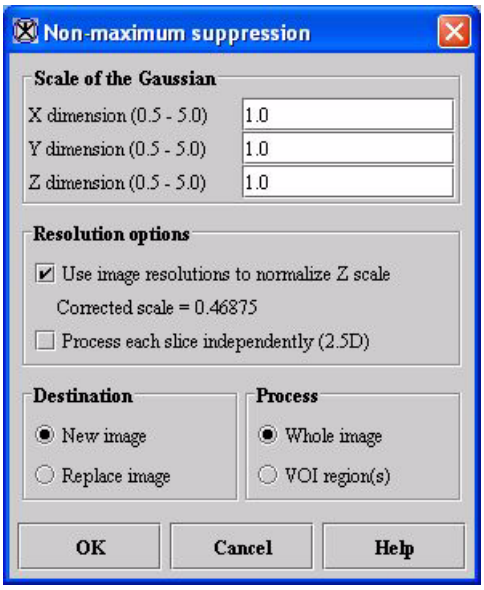
To run this algorithm, complete the following steps:

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 45) 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.



X dimension	Indicates the scale of the Gaussian in the *X* direction (the default value is 1.0).
Y dimension	Indicates the scale of the Gaussian in the *Y* direction (the default value is 1.0).
Z dimension	Indicates the scale of the Gaussian in the *Z* direction (for 3D images only). The default value is 1.0.
Use image resolutions to normalize *Z* scale	Normalizes the *Z* 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). If enabled, then $Z = Z\left(\frac{XRs}{ZRs}\right)$ where *Z* = scale Z, *XRs* = image X resolution, and *ZRs* = image Z resolution).
Process each slice independently	Calculates nonmaximum suppression for each slice of the dataset independently (for 3D images only the default value is enabled).
New image	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.
Replace image	Replaces the current active image with the results of the algorithm.
Whole image	Applies the algorithm to the whole image (default choice).
VOI region(s)	Applies the algorithm inside VOIs. Outside VOIs, the pixel values are unchanged.
OK	Applies the algorithm according to the specifications in this dialog box.
Cancel	Disregards any changes that you made iin the dialog box and closes this dialog box.
Help	Displays online help for this dialog box.

Figure 45. 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 (see "References" on page 144). 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.

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

$$\partial_t I = div(D \nabla I)$$

where

∂_t =The time derivative of the image

div=The divergence operator

D=The diffusion tensor (which may be a scalar value)

∇ =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:

$$\partial_t I = div(D(|\nabla I|^2)\nabla I)$$

One useful diffusivity function is:

$$D(|\nabla I|^2) = \frac{1}{1 + (|\nabla I|^2 / \lambda^2)}$$

In this equation λ is called a diffusion constant, which defines the point of inflection of the diffusivity function, which is shown for $\lambda = 0.15$ in Figure 46.

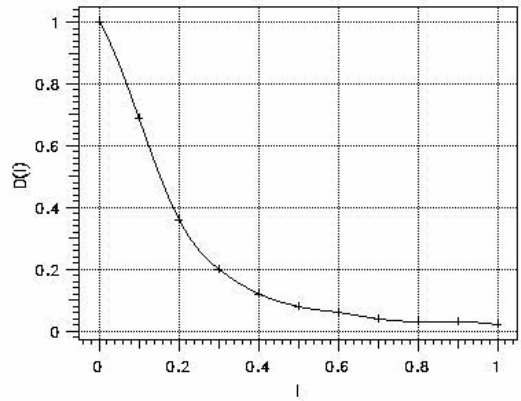


Figure 46. Diffusion constant for $\lambda = 0.15$

Figure 46 shows that, for a given value of the diffusion constant λ, 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:

$$\partial_t I = div\left(D\left(\left|\nabla I_\sigma\right|^2\right)\nabla I_\sigma\right)$$

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 47 show the results of applying the *regularized isotropic (nonlinear) diffusion* filter to an example MR brain image.

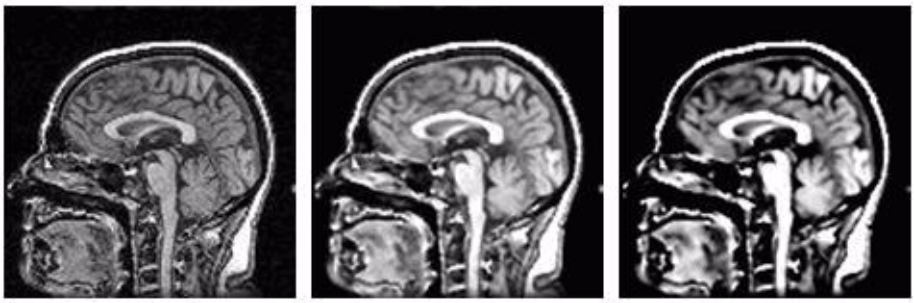


Figure 47. Examples of regularized isotropic (nonlinear) diffusion

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.

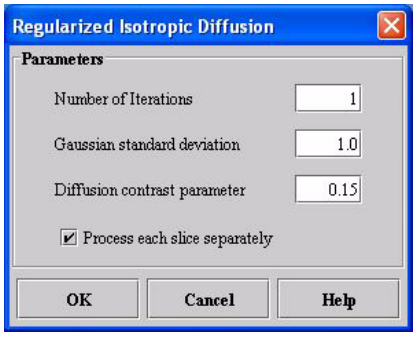
Weickert, Joachim. *Anisotropic Diffusion in Image Processing* (Stuttgart, Germany: Teubner, 1998).

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 48).
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.



Number of iterations	Specifies the number of iterations, or number of times, to apply the algorithm to the image.
Gaussian standard deviation	Specifies the standard deviation of the Gaussian filter used to regularize the image.
Diffusion contrast parameter	Specifies the inflection point in the diffusivity function, which dictates the shape of the function.
Process each slice separately	Applies the algorithm to each slice individually. By default, this option is selected.
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.
Help	Displays online help for this dialog box.

Figure 48. 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.

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:

$$A(x, y) = \frac{\sum_{s=1}^{s=LSN} value(x, y, s)}{LSN}$$

If you select 3, 5, or 7 as the number of slices averaged:

$$A(x, y, z) = \frac{\sum_{s=lowest}^{s=highest} value(x, y, s)}{(highest - lowest + 1)}$$

where

A = slice average

LSN = last slice number

$lowest$ = maximum (1, z - offset)

$highest$ = minimum (last slice number, z + offset)

$offset$ = (number of slices averaged -1)/2

The preceeding 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.

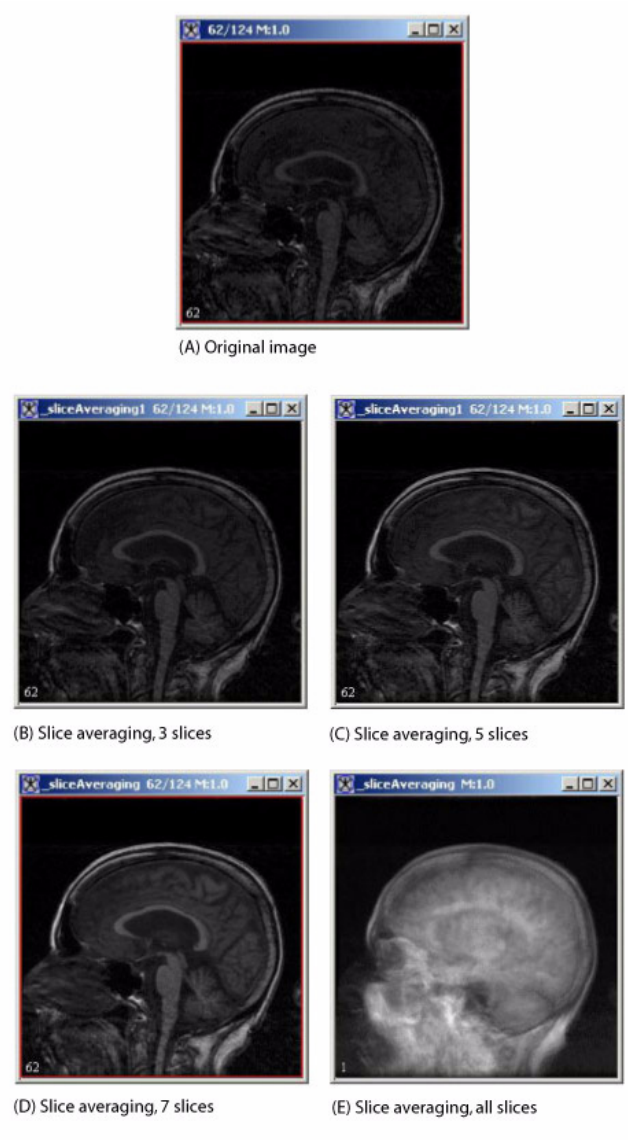


Figure 49. Slice Averaging algorithm processing

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 50) 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.

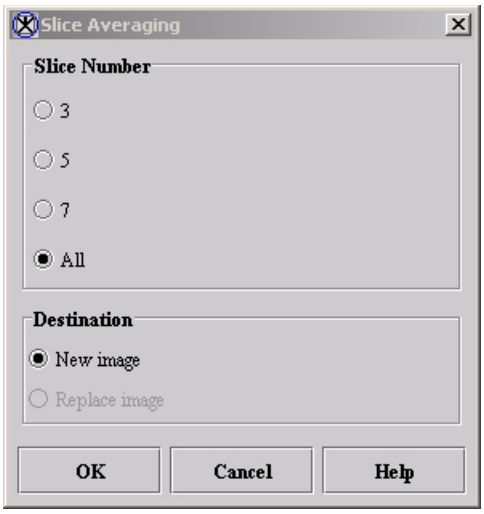


Figure 50. Slice Averaging dialog box

Help	Displays online help for this dialog box.

Figure 50. Slice Averaging dialog box

3. Complete the information in the dialog box.
4. Click OK.

   The algorithms 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.

Background

The process of unsharp masking is defined by:

High pass = original – low pass

To control the amount of filtering, a weighting factor (k) is added to the function:

High pass = original – (k * blurred image) where $0 < k < 1$

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 51).

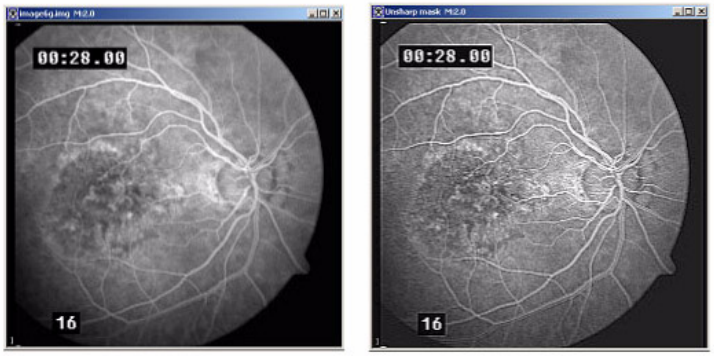
Image types

You can apply this algorithm to all image data types, except complex and RGB images, and to 2D, 2.5D, 3D, and 4D images.

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 52).



(A) Original image

(B) Image after Unsharp Mask algorithm is applied

Figure 51. 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.

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 52) 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.

X Dimension	Specifies the standard deviation (SD) of Gaussian in the *X* direction.
Y Dimension	Specifies the SD of Gaussian in the *Y* direction.
Z Dimension	Specifies the SD of Gaussian in the *Z* direction.
Use image resolutions to normalize Z scale	Normalizes the Gaussian to compensate for the difference if the voxel resolution is less between slides than the voxel resolution in-plaine. This option is selected by default.
Process each slice independently (2.5D)	Filters each slice of the dataset independently of adjacent slices.
Image - (k<1) * blurred image	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.
New image	Shows the results of the algorithm in a new image window.
Replace image	Replaces the current active image with the results of the algorithm.

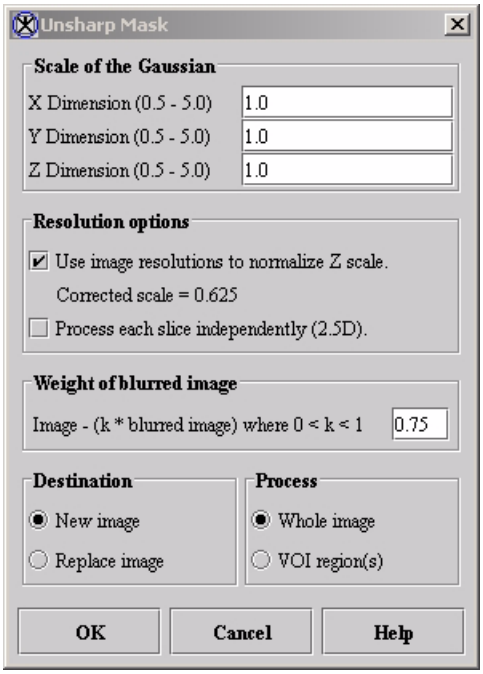


Figure 52. Unsharp Mask dialog box

Whole image	Applies the algorithm to the whole image.
VOI region(s)	Applies the algorithm to the volumes (regions) delineated by the VOIs.
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.
Help	Displays online help for this dialog box.

Figure 52. Unsharp Mask dialog box

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:

(ujk**q) * ((yj - vk)**2)

nClass = number of classes

where

ujk is the membership value at pixel location *j* for class *k* such that the sum over *k* from *k* = 1 to *k* = *n*Class for *ujk* = 1.

q is a weighing exponent on each membership value and determines the amount of "fuzziness" of the resulting segmentation.

q., which must be greater than 1, is typically set to 2.

yj is the observed single channel or multispectral image intensity at location *j*.

vk is the centroid of class *k*.

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 Algorithm > Fuzzy C-means > Single channel in the MIPAV window):

numerator = (yj - vk)**(-2/(q-1))
denominator = Sum over l from l = 1 to l = nClass of (yj-FL)**(-2/(q-1))
ujk = numerator/denominator for all pixels j and all classes k

In the multispectral case (select Algorithm > Fuzzy C-means > Multispectral in the MIPAV window):

numerator = (Sum over spectra s of (yjs - vks)**2)**(-1/(q-1))
denominator = Sum over l from l = 1 to l = nClass of
(Sum over spectra s of (yjs -vls)**2)**(-1/(q-1))

The minimization process also computes the centroids using current estimates of the membership functions.

In the single channel case:

numerator = sum over all pixels j of (ujk**q) * yj
denominator = sum over all pixels j of (ujk**q)
vk = numerator/denominator for all classes k

In the multispectral case:

numerator = sum over all pixels j of (ujk**q) * yjs
denominator = sum over all pixels j of (ujk**q)
vks = numerator/denominator for 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 55), 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 x 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.

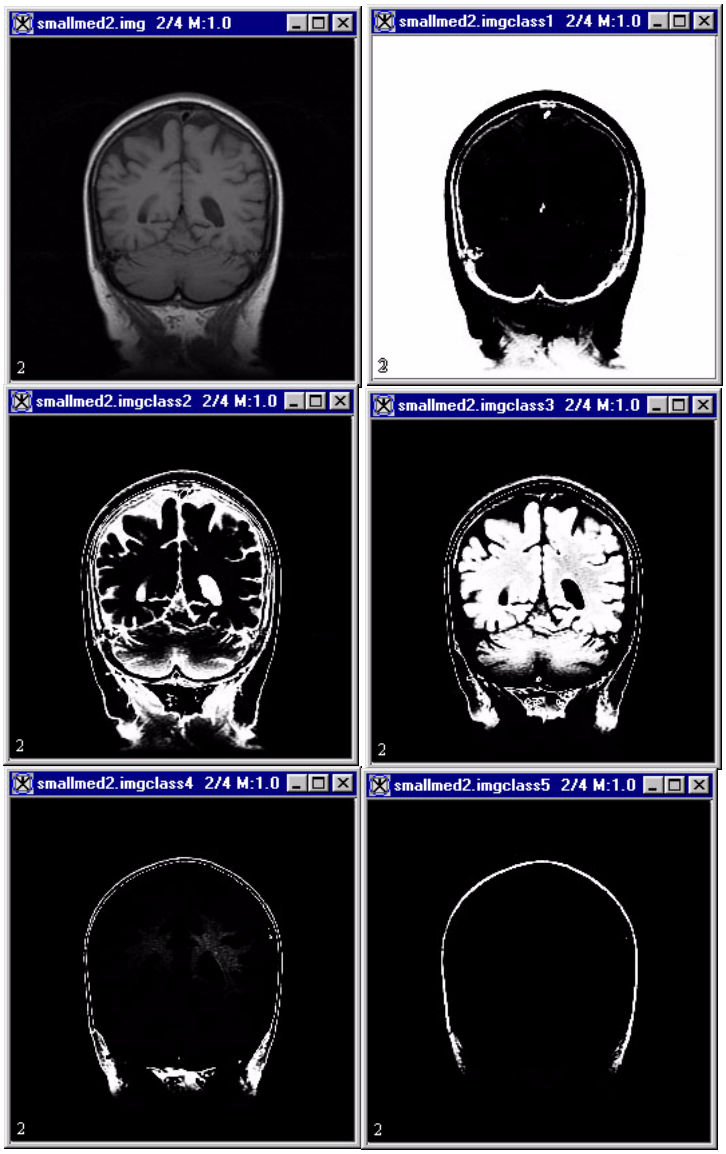


Figure 53. Fuzzy C-means algorithm processing

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 54) 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.

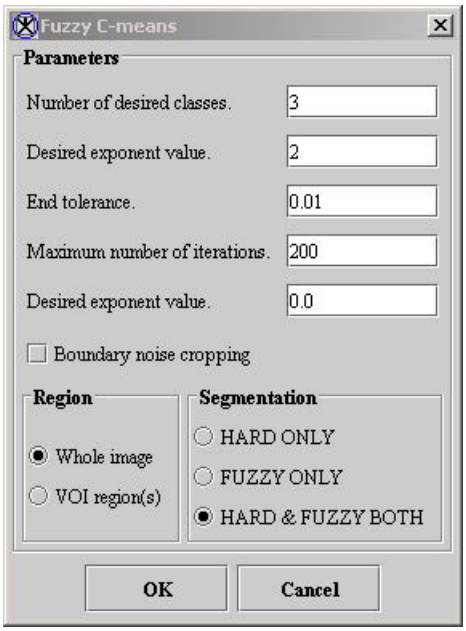


Figure 54. Fuzzy C-Means dialog box for a single image

Maximum number of iterations	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.
Signal treshold	Determines the treshold. By default, the exponent value is 0.0.
Background cropping	Finds the smallest bounding box in the image or delineated VOI outside of which all image pixel values are below the image threshold.
Whole image	Applies the algorithm to the whole image.
VOI region(s)	Applies the algorithm only to the delineated VOIs.
HARD ONLY	Performs only hard segmentation on the image
FUZZY ONLY	Performs only fuzzy segmentation on the image.
HARD & FUZZY BOTH	Performs both types of segmentation—hard and soft—on the image.
OK	Performs the Fuzzy C-Means algorithm on the image based on your choices in this dialog box.
Cancel	Disregards any changes that you made in this dialog box and closes the dialog box.
Help	Displays online help for this algorithm.

Figure 54. 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 55) 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.

Number of desired classes	Indicates the number of groups into which the algorithm divides the dataset structures.
Desired exponent value	Determines the amount of "fuzziness" in the resulting segmentation.
Boundary noise cropping	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.
Signal threshold	Appears only when you select Algorithm > Fuzzy C-means > Single Channel. By default, MIPAV assigns the image minimum value and uses only pixels whose values equal or exceed the theshold used in the centroid calculation. 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.

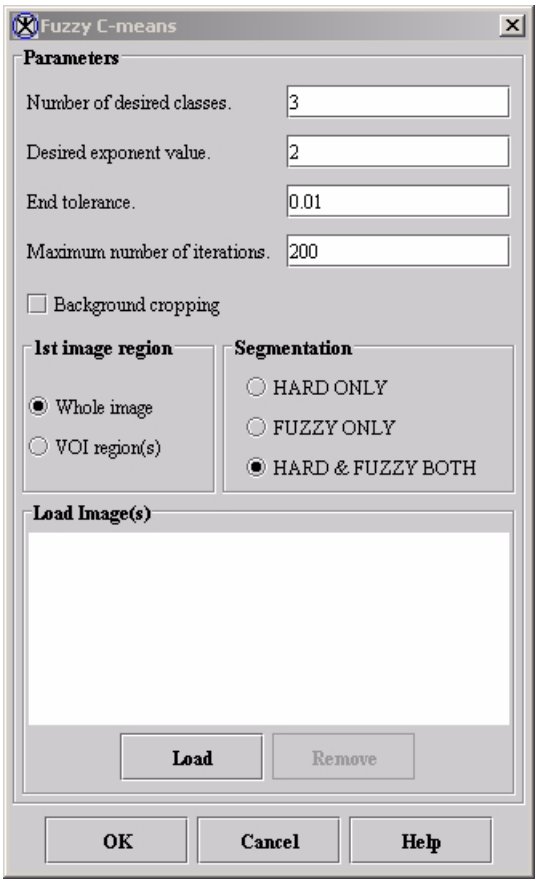


Figure 55. Fuzzy C-Means dialog box for multispectral images

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.
Maximum number of iterations	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.
Background noise cropping	Finds the smallest bounding box in the image or delineated VOI outside of which all image pixel values are below the image threshold.
Whole image	Applies the algorithm to the whole image.
VOI region(s)	Applies the algorithm only to the delineated VOIs.
HARD ONLY	Performs only hard segmentation on the image
FUZZY ONLY	Performs only fuzzy segmentation on the image.
HARD & FUZZY BOTH	Performs both types of segmentation—hard and soft—on the image.
Load	Loads the images that you select.
Remove	Removes the images that you select.
OK	Performs the FuzzyC-Means algorithm on the selected images based on your choices in this dialog box.
Cancel	Disregards any changes that you made in this dialog box and closes the dialog box.
Help	Displays online help for this algorithm.

Figure 55. Fuzzy C-Means dialog box for multispectral images (continued)

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.

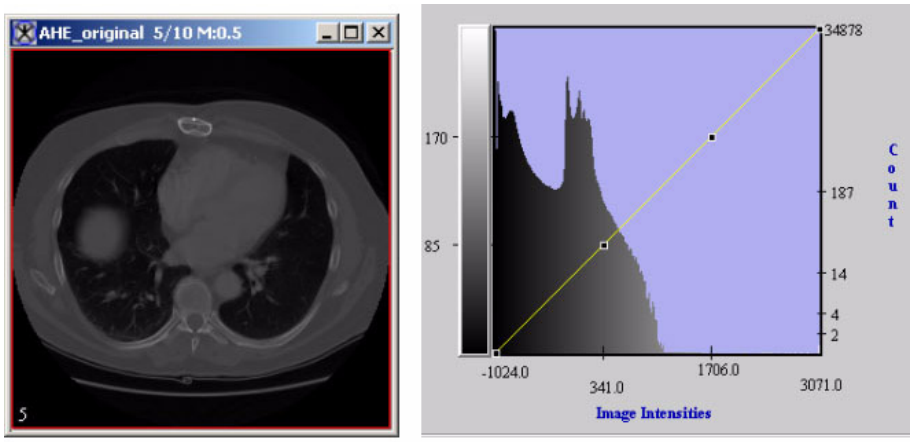
Although, depending on image type, there are some minor variations in how this algorithm proceeds, generally the algorithm first calculates the *bin width* of each region. It then processes each region by judging each pixel's location within this new histogram. Regardless of pixel value, the algorithm 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:

$$k = \sum_{i=0}^{j} \frac{N_i}{T}$$

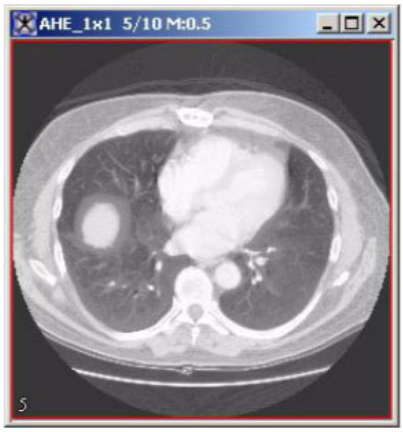
where N_i is the intensity of the ith pixel, and T is the total number of pixels in the region.

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 56).

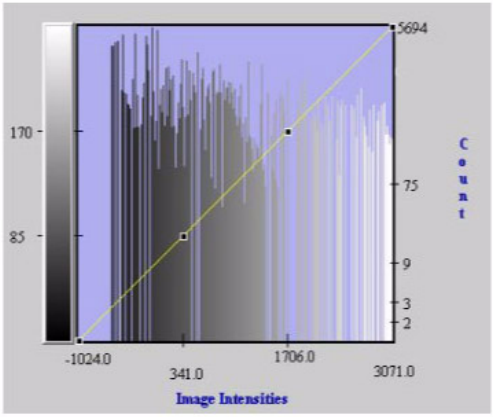


(A) Original, or source, image (B) Histogram for source image

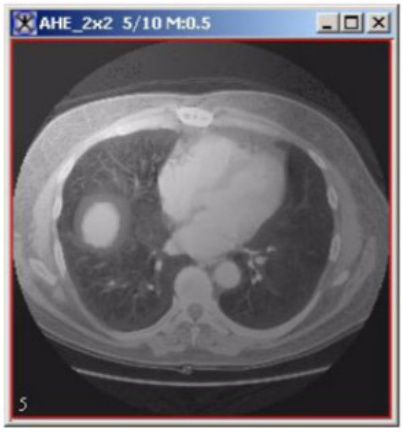
Figure 56. Source image and histogram and reference images and histograms for reference images that were separated into 1 x 1, 2 x 2, and 3 x 3 regions



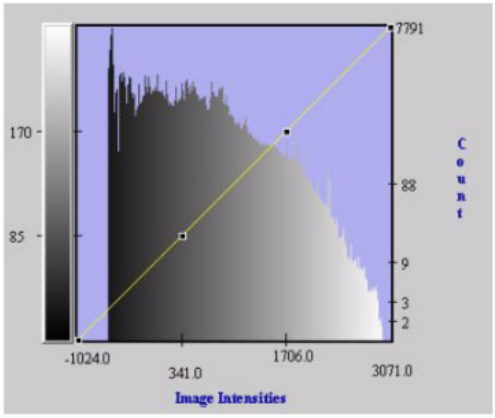
(C) Reference image that was separated into 1 x 1 regions



(D) Histogram for reference image that was separated into 1 x 1 regions

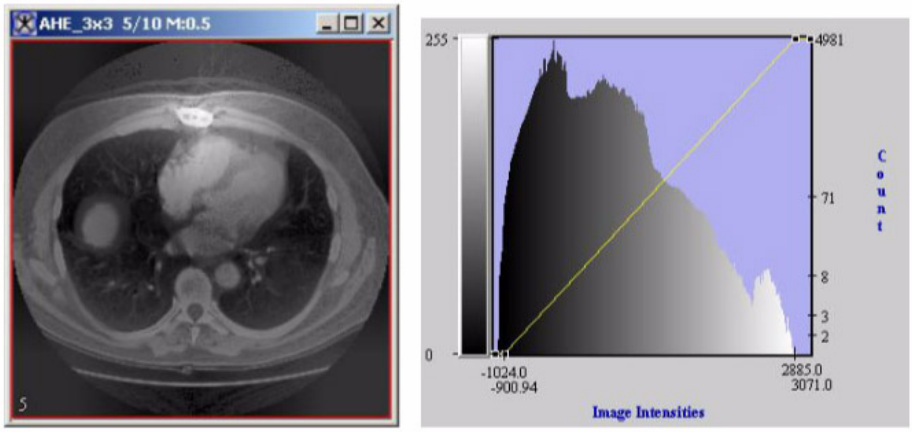


(E) Reference image that was separated into 2 x 2 regions



(F) Histogram for reference image that was separated into 2 x 2 regions

Figure 56. 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)



(G) Reference image that was separated into 3 x 3 regions

(H) Histogram for reference image that was separated into 3 x 3 regions

Figure 56. 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.

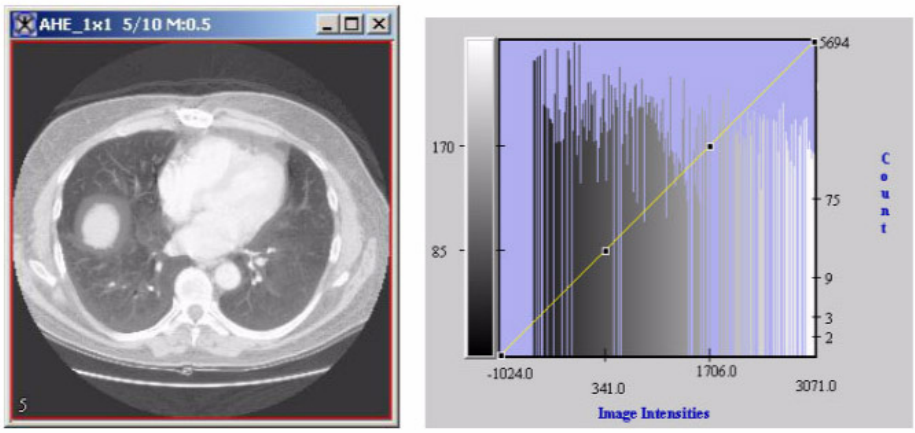
The histogram equalization process can enforce certain characteristics about the image called *clipping*. As the algorithm processes the image, it counts out the number of pixels of a certain intensity. Clipping cuts that counting short; it stores the excess pixels of a particular intensity and redistributes 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 57 shows examples of the source image from Figure 56 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

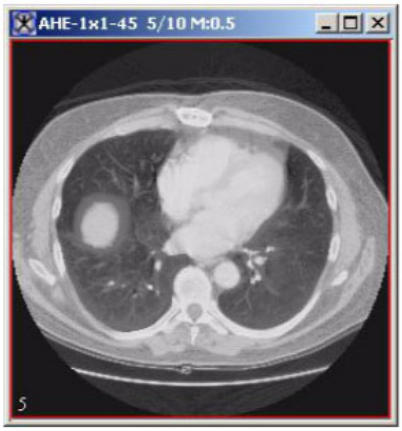
This algorithm only works on the whole image, not on volumes of interest (VOIs).



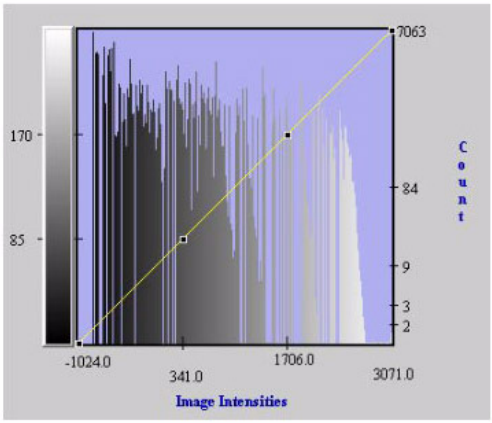
(A) Reference image that was separated into 1 x 1 regions with no clipping

(B) Histogram for reference image that was separated into 1 x 1 regions with no clipping

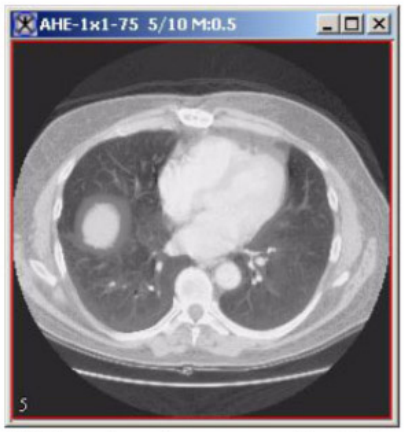
Figure 57. Examples of no clipping, 45-percent clipping, and 75-percent clipping



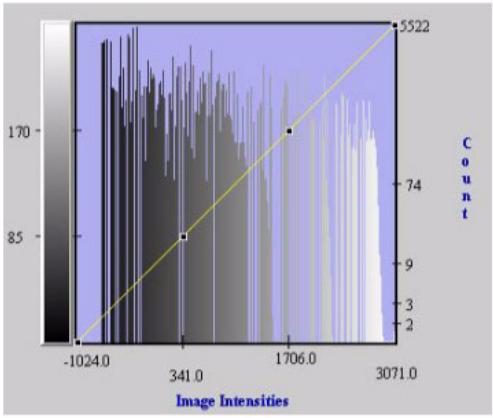
(C) Reference image that was separated into 1 x 1 regions with 45-percent clipping



(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 57. 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 58) opens.
4. Complete the information in the dialog box.
5. Click OK. The algorithm begins to run.

Divisions of width	Indicates the number of times the algorithm should divide the image in width to make a region. 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.
Divisions of height	Indicates the number of times the algorithm should divide the image in height to make a region. If you select 3, the algorithm divides the image in height by 3 yielding three areas: a top, middle, and bottom part.
Activate clamping	Allows you to specify the maximum number of pixels with a certain pixel bin. Selecting this check box makes Fraction of most frequent pixel intensity active.

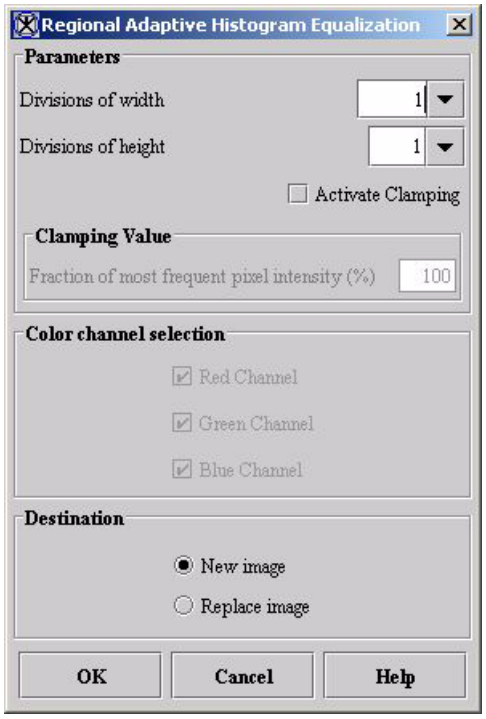


Figure 58. The Regional Adaptive Histogram Equalization dialog box

Fraction of most frequent pixel intensity (%)	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 oversatured. To use this field, first select Activate clamping.
Red channel	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).
Green channel	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).
Blue channel	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).
New image	Shows the results of the algorithm in a new image window.
Replace image	Replaces the source image with the results of the algorithm.
OK	Applies the algorithm according to the specifications in this dialog box.
Cancel	Disregards any changes you made in this dialog box and closes the dialog box.
Help	Displays online help for this dialog box.

Figure 58. 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 59. 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.

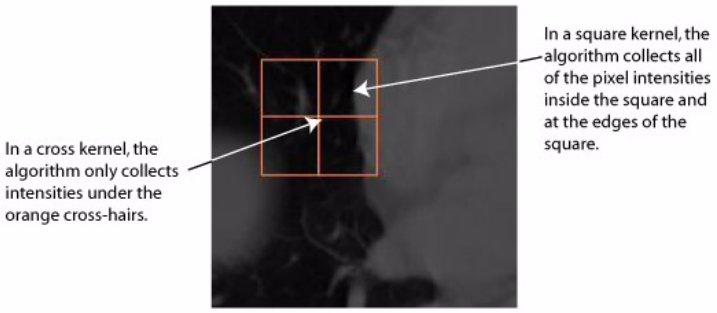


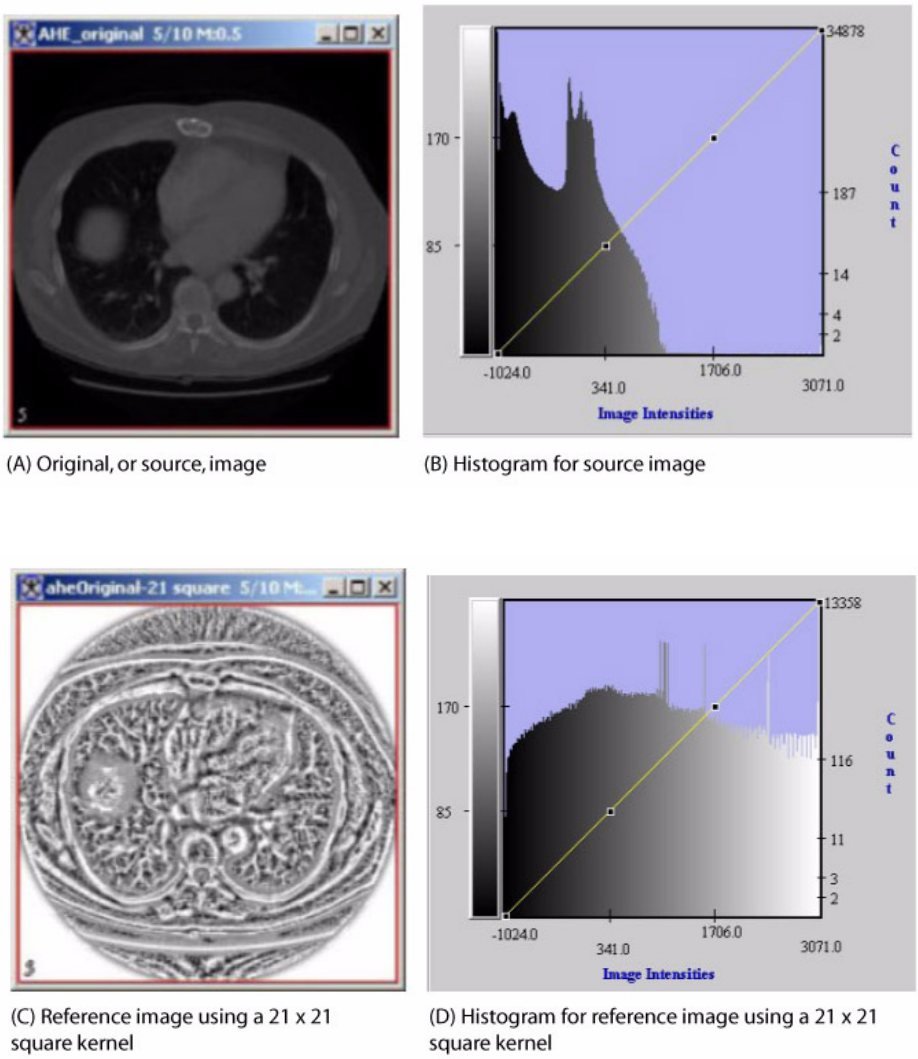
Figure 59. Sample of kernel created by algorithm

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 60 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.



(A) Original, or source, image

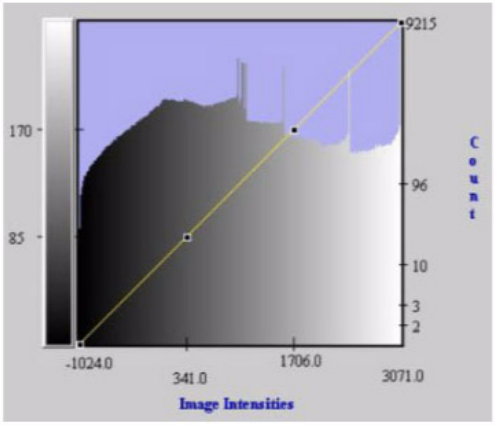
(B) Histogram for source image

(C) Reference image using a 21 x 21 square kernel

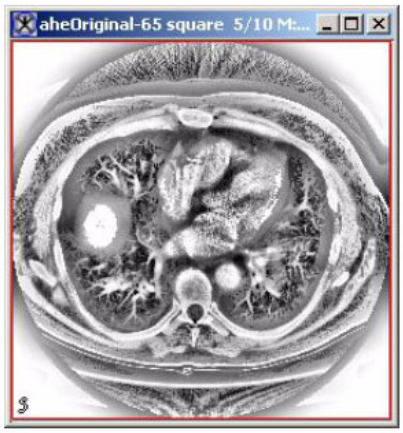
(D) Histogram for reference image using a 21 x 21 square kernel

Figure 60. 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

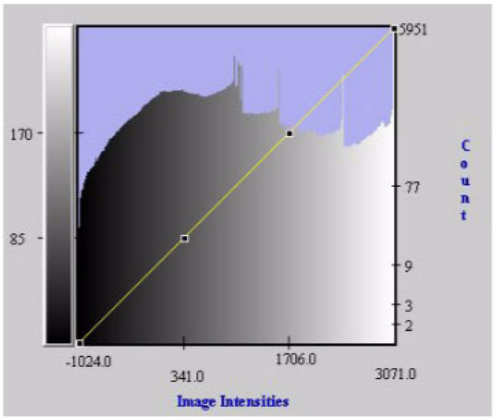
(E) Reference image using a 45 x 45 square kernel



(F) Histogram for reference image using a 45 x 45 square kernel



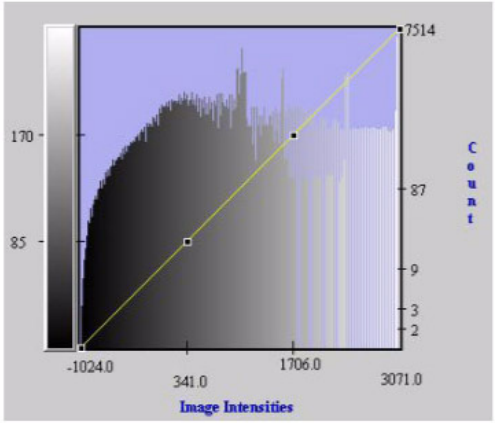
(G) Reference image using a 65 x 65 square kernel



(H) Reference image using a 65 x 65 square kernel

Figure 60. 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

(I) Reference image using a 65 x 65 cross kernel

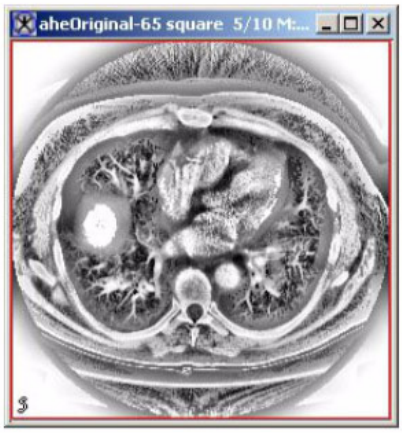


(J) Histogram for reference image using a 65 x 65 cross kernel

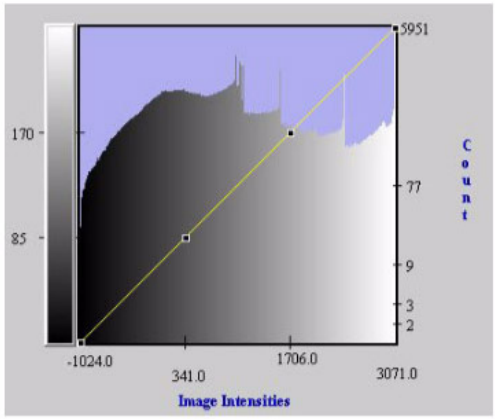
Figure 60. A source image and its histogram (A, B) compared to reference images using a 21 x 21 (C, D), 45 x 45 (E,F), or 65 x 65 (G,H) square kernels and (continued)65 x 65 (I,J) cross kernels

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 threshholding feature to prevent processing on a pixel when its intensity is too low.

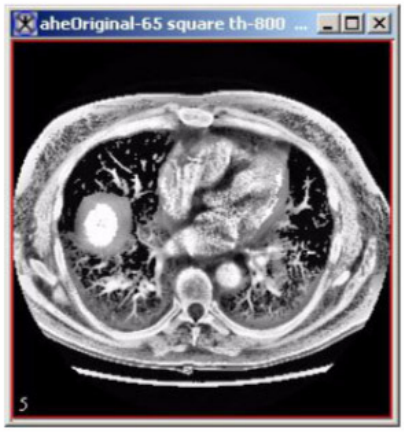
The histogram equalization process can enforce certain characteristics about the image called *clipping*. As the algorithm processes the image, it counts out the number of pixels of a certain intensity. *Clipping* cuts that counting short; it stores the excess pixels of a particular intensity and redistributes 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.



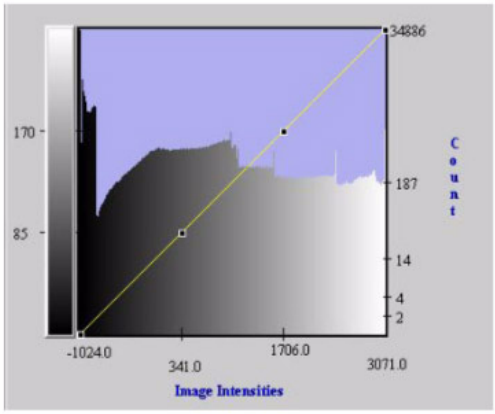
(A) Reference image using 65 x 65 square kernel with no threshold



(B) Histogram for image using 65 x65 square kernel with no threshold

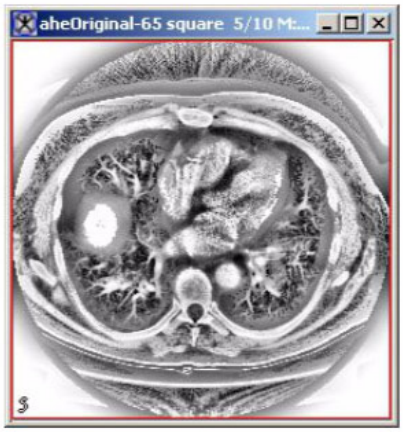


(C) Reference image using 65 x 65 square kernel with threshold set at -800

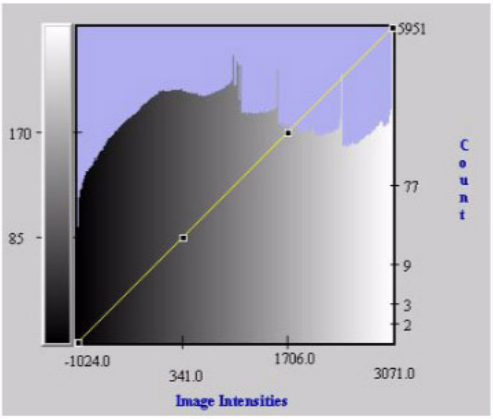


(D) Histogram for image using 65 x 65 square kernel with threshold set at -800

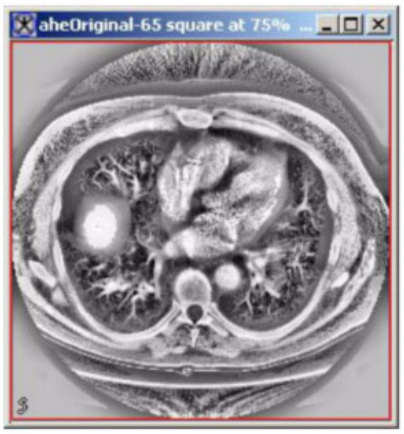
Figure 61. The effect of thresholding on images



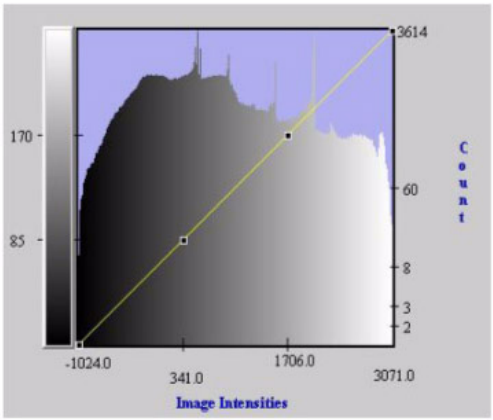
(A) Reference image using 65 x 65 square kernel, no clipping



(B) Histogram for reference image using 65 x 65 square kernel, no clipping

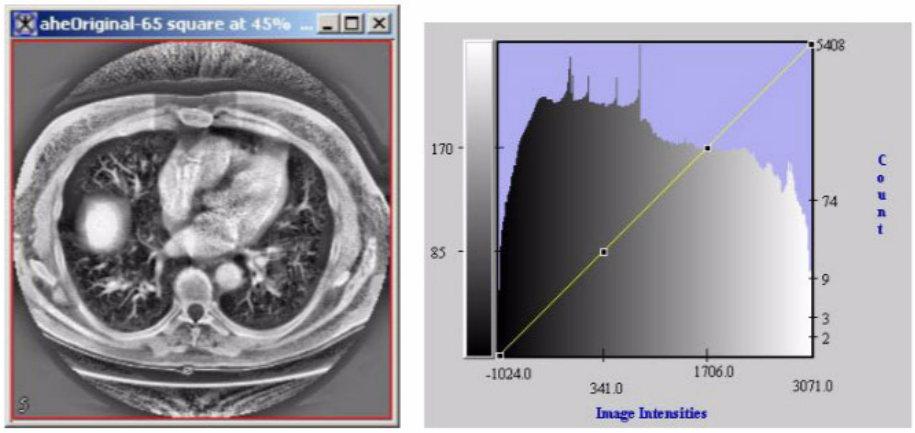


(C) Reference image using 65 x 65 square kernel with 75-percent clipping



(D) Histogram for reference image using 65 x 65 square kernel with 75-percent clipping

Figure 62. A reference image and histogram with no clipping and with 75-percent clipping



(A) Reference image using 65 x 65 square kernel with 45-percent clipping

(B) Histogram for reference mage using 65 x 65 square kernel with 45-percent clipping

Figure 63. 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, Swedon: 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 64) 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.

Kernel size	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.
Kernel shape	Specifies the shape of the kernel the algorithm should use. Select either: • Square—To evaluate all pixels on any column or row away from the central pixel to the perimeters of the kernel • Cross—To choose from only pixels in the same column or row to the perimeter of the kernel as neighbors of the pixel.
Use as scale maximum	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.
Use threshold	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. 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.
Use clamping	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.
Red channel	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.

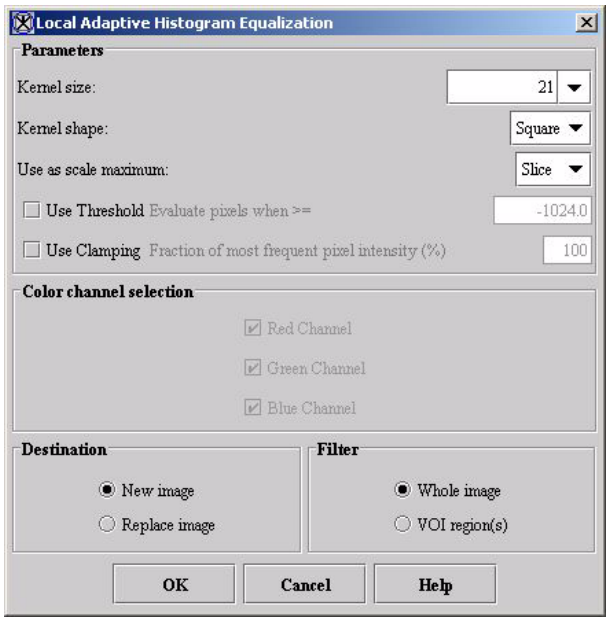


Figure 64. Local Adaptive Histogram Equalization dialog box

Green channel	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.
Blue channel	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.
New image	Shows the results of the algorithm in a new image window.
Replace image	Replaces the source image with the results of the algorithm.
Whole image	Applies the algorithm on the entire image.
VOI regions	Applies the algorithm to only the selected VOI.
OK	Applies the algorithm according to the specifications in this dialog box.
Cancel	Disregards any changes you made in this dialog box and closes the dialog box.
Help	Displays online help for this dialog box.

Figure 64. 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 pd(g) are generated. Here, the desired histogram of the specified image is denoted as pd(g). Cumulative histograms – P(f) and Pd(g) are then generated using the following equations:

EQUATION 2

$$P(f) = \sum_{k=0}^{f} p(k)$$

EQUATION 3

$$P_d(g) = \sum_{k=0}^{g} p_d(k)$$

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 Pd(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.

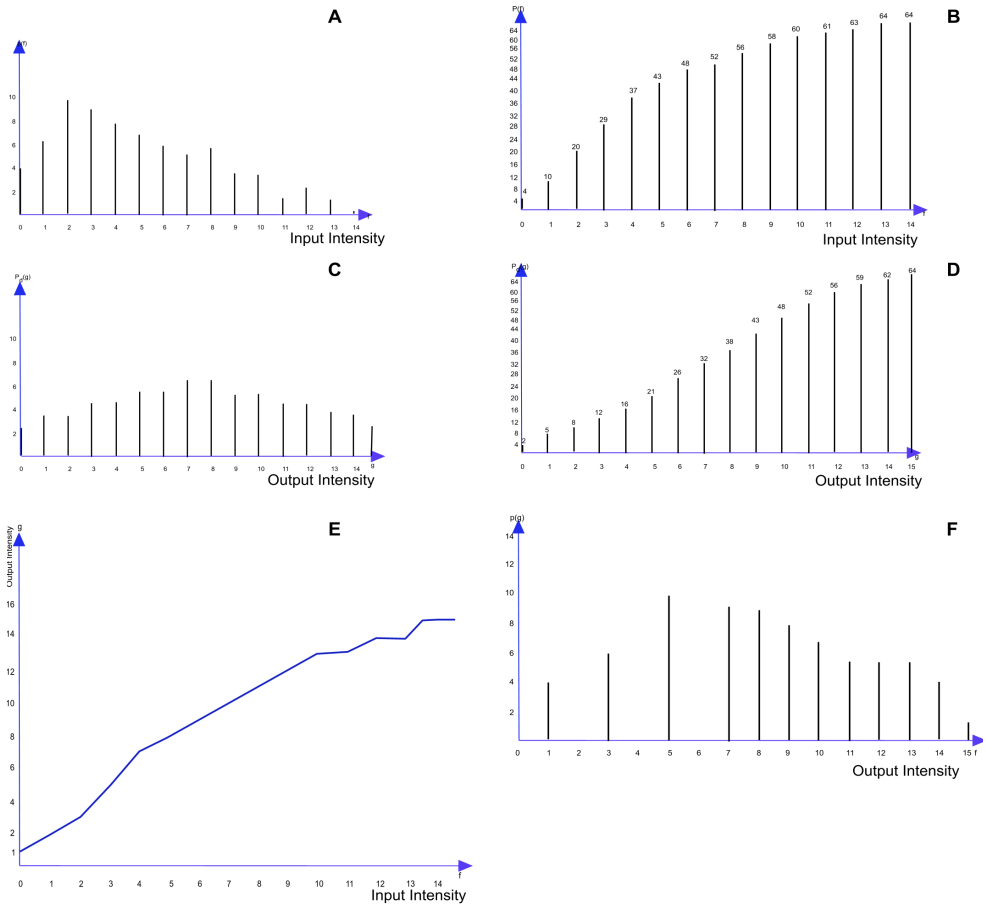


Figure 65. Histograms and cumulative histograms:
(A) histogram of 8x8 pixel image;
(B) cumulative histogram derived from (A);
(C) desired histogram of the same 8x8 pixel image;
(D) cumulative histogram derived from (C);
(E) gray scale transformation function that approximately transforms the histogram (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).

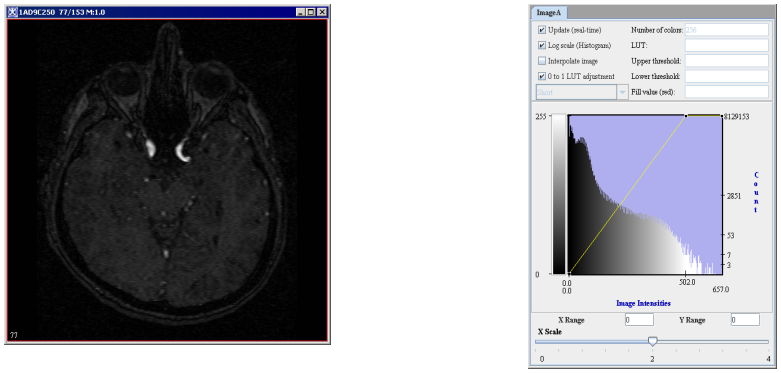


Figure 66. The input image and its histogram p(f)

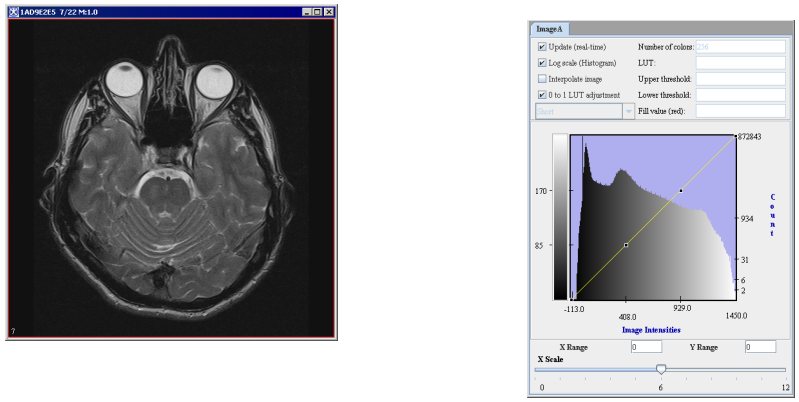


Figure 67. The specified image and its histogram pd(g)

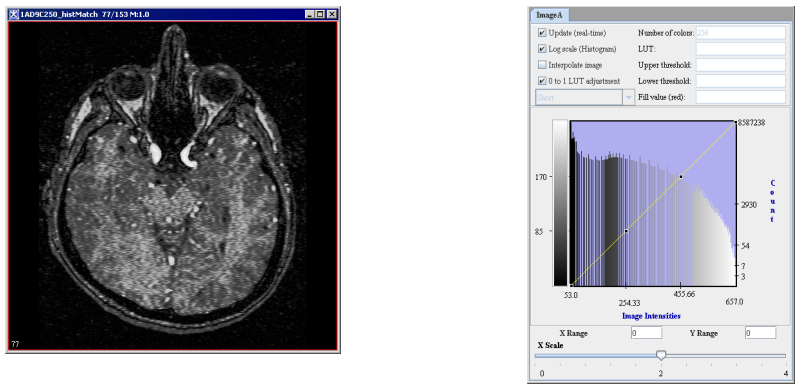


Figure 68. The result image and its histogram

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.

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

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.

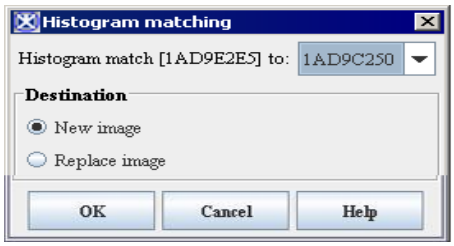
Histogram match <File name> to	Select the specified image $p_d(g)$ from the list box 
New image	Shows the results of the algorithm in a new image window
Replace image	Shows the results of the algorithm in the same image window
OK	Applies the algorithm according to the specifications in this dialog box.
Cancel	Disregards any changes that you made in the dialog box and closes this dialog box.
Help	Displays online help for this dialog box.

Figure 69. Histogram Matching dialog box

2D Histogram

The 2D Histogram algorithm takes as 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 double or float 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 72 on page 195 shows the two input images, the output 2D histogram image and its lookup table.

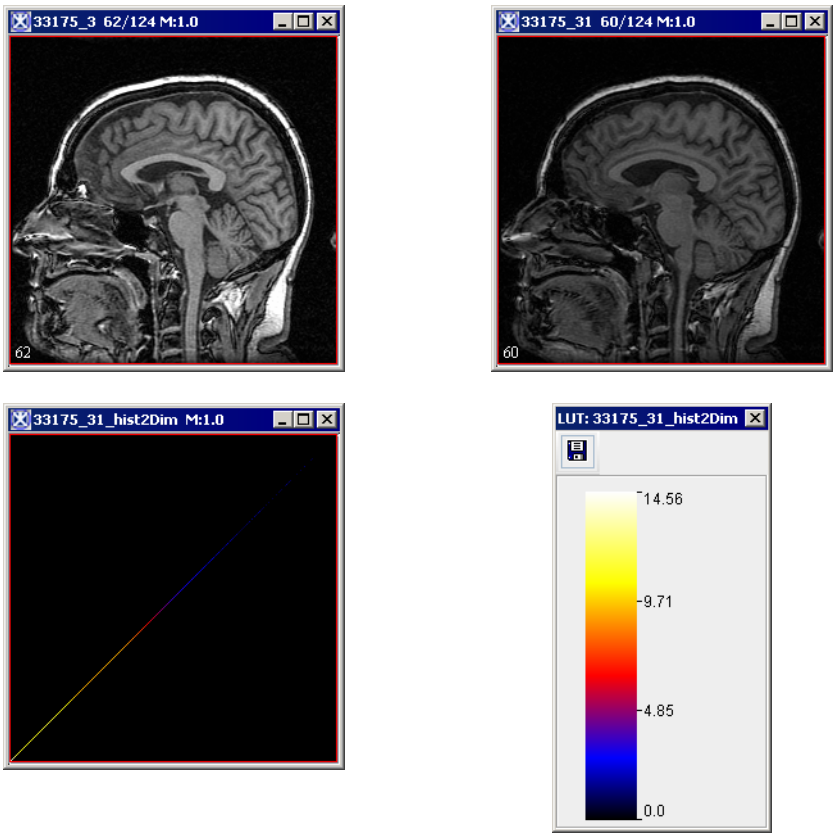


Figure 70. Two input images, output 2D histogram image and its lookup table

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 71 on page 193 for details.

Histogram 2D with <File name> to:	Select the base image from the list box.
Options	
Linearly rescale 2-nd image	Use this checkbox if you want to rescale the 2-nd image. By default, the box is checked.
Image 1 bin format	Enter the bin number for the image 1. The default value is 256.
Image 2 bin format	Enter the bin number for the image 2. The default value is 256.
Calculate the log of the result image	Check if you want to calculate the log of the result image for better visualization. By default, the box is checked.
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.

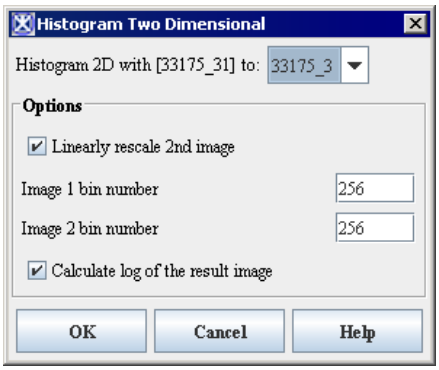


Figure 71. The Histogram Two Dimensional dialog box for grayscale images

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 72 on page 195.

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.

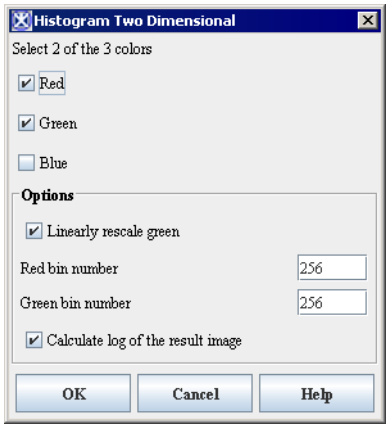
Select 2 of 3 colors	Use the checkbox to select any two colors from red, green, and blue. Note that, only two colors can be selected.	
	Options	
Linearly rescale	Use this checkbox if you want to rescale the 2-nd color. By default, the box is checked.	
Color 1 bin number	Enter the bin number for color 1. The default value is 256.	
Color 2 bin number	Enter the bin number for color 2. The default value is 256.	
Calculate the log of the result image	Check if you want to calculate the log of the result image for better visualization. By default, the box is checked.	
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.	

Figure 72. The Histogram Two Dimensional dialog box for color images

Creating a 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 73).

The histogram summary table includes the following four columns, see Figure 73:

- **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, $\text{area} = \text{count} \times x \text{ resolution} \times y \text{ resolution}$. For 3D images, $\text{volume} = \text{count} \times x \text{ resolution} \times y \text{ resolution} \times z \text{ resolution}$.

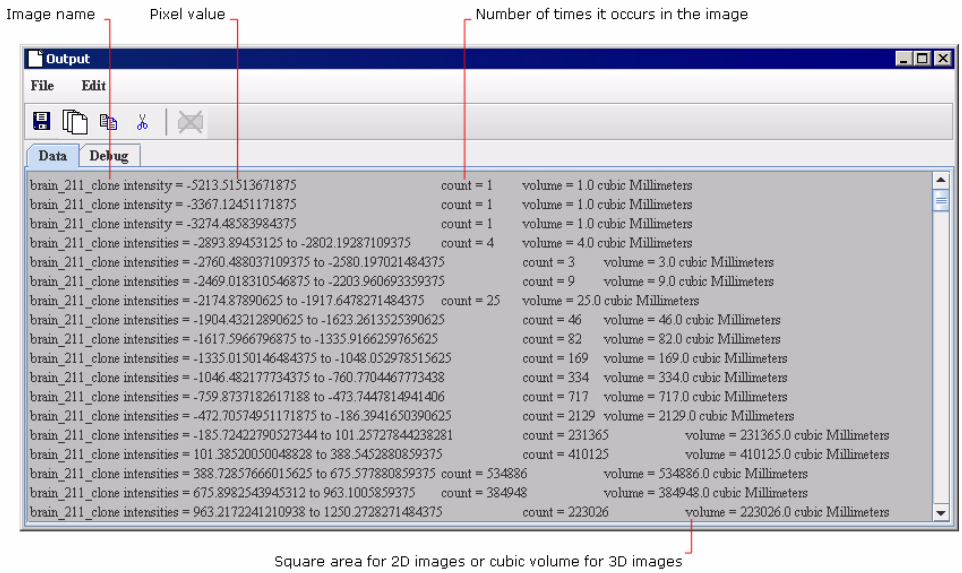


Figure 73. Histogram summary in the Output window

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.

Number of bins	Specify the number of bins here. The default value is 256.
Whole image	Check this option if you want create histogram for the whole image.
VOI regions	Check this option if you want create histogram only for selected VOI regions.
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.

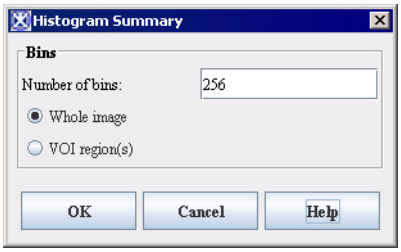


Figure 74. The Histogram summary dialog box options

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. 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 75).

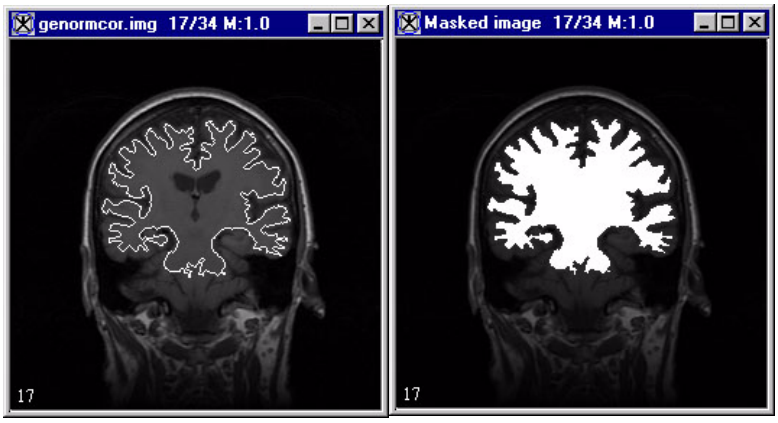


Figure 75. Mask algorithm processing

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 76) appears.

Value used to fill VOI(s)	Indicates the intensity value that replaces the existing value inside or outside the region bound by the VOI contours. 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. To replace the intensity value outside the VOI region, simply type a value in this text box.
Interior/exterior fill	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).
New image	If selected, indicates that the results of the algorithm appear in a new image window.
Replace image	If selected, indicates that the results of the algorithm replace the current active image.
OK	Performs the Mask algorithm on the selected images based on your choices in this dialog box.
Cancel	Disregards changes you made in this dialog box and closes the dialog box.
Help	Displays online help for the algorithm.

Figure 76. 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 so that 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 77 on page 213) 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

$$L = \frac{\sum_i ((R_i - \bar{R}) \times (G_i - \bar{G}))}{\sqrt{\sum\left((R_i - \bar{R})^2\right) \times \sum\left((G_i - \bar{G})^2\right)}}$$

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 anticolocalized 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, background1= 1.0 and 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 background1 and background2. However, in the second iteration, points are excluded if they are below either lastPositive1 or lastPositive2. The (lastPositive1, 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*).

$$\text{slope of the line} = \frac{\text{direction } y}{\text{direction } x} = \frac{Vy - Vx + \sqrt{(Vy-Vx)^2 \times 4Cxy^2}}{2Cxy}$$

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:

$$MSE = \text{variance } x \times (\cos\theta)^2 + \text{covariance } xy \times \sin 2\theta + \text{variance } y \times (\sin\theta)^2$$
$$= (\text{variance } x + 2\text{slope} \times \text{covariance } xy + \text{slope}^2 \times \text{variance } y)/(1 + \text{slope}^2)$$

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 $\text{red} \geq \text{threshold}$ and $\text{green} \geq \text{secondThreshold}$.

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 >= threshold is . . .	
Percent	**Selected**	**Not selected (default)**
Red colocalization	$100 \times \frac{R}{R_T}$ where R = Sum of the red values in the upper right rectangular region R_T = Sum of the red values >= red threshold in the image or selected VOI	$100 \times \frac{R}{R_I}$ where R = Sum of the red values in the upper right rectangular region R_I = Sum of the red values in the image or the selected VOI
Green colocalization	$100 \times \frac{G}{G_T}$ where G = Sum of the green values in the upper right rectangular region G_T = Sum of the green values >= green threshold in the image or selected VOI	$100 \times \frac{G}{G_I}$ where G = Sum of the green values in the upper right rectangular region G_I = Sum of the green values in the image or the selected VOI

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 Dissanaike 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.

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

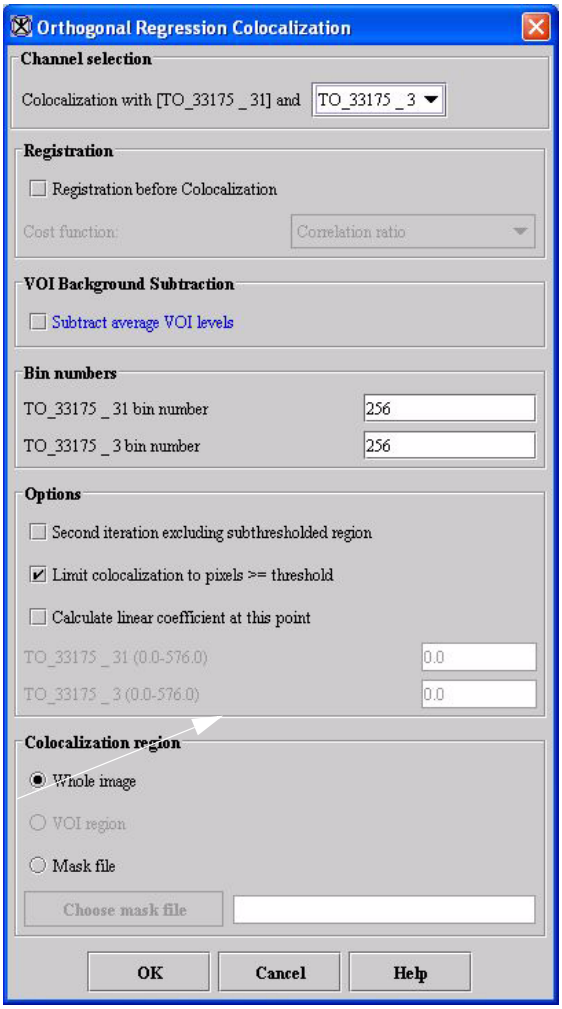
To run this algorithm, complete the following steps:

1. Open either one color or two black-and-white images.
2. Select Algorithms > Microscopy > Colocalization > Orthogonal Regression.

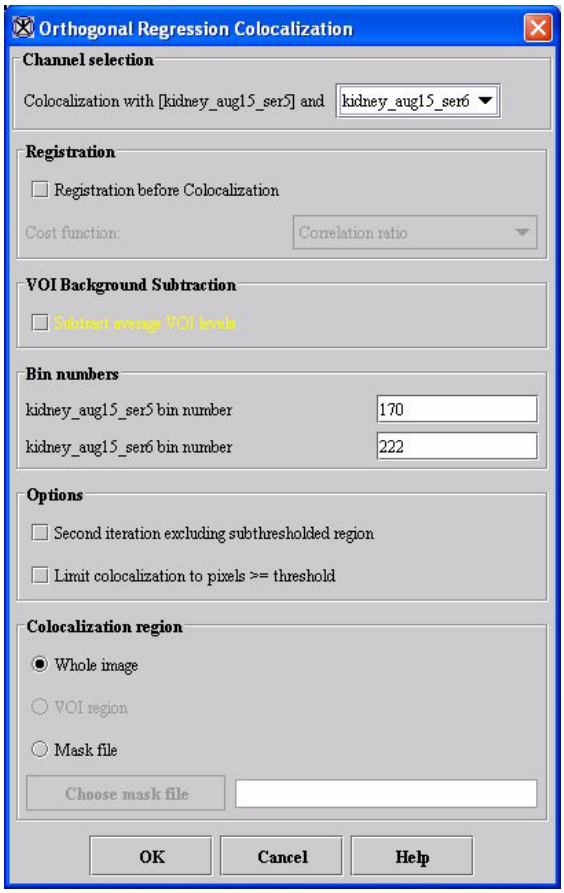
   Depending on whether the image is grayscale or color, the Orthogonal Regression dialog box for grayscale images (Figure 1) or the Orthogonal Regression box for color images (Figure 2) opens.
3. Complete the information in the dialog box.
4. Click OK.

   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 77. Orthogonal Regression Colocalization dialog boxes for (A) color and (B) grayscale images

Channel selection	**Color images:** Specifies the colors to use in color images. • 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. • 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. **Grayscale images:** Specifies the second grayscale image to be used. Select one from the list box.
Registration before colocalization	Performs registration on an entire image before performing colocalization. When you select Registration before colocalization, the Cost function box becomes available. By default, registration is not selected.
Cost function	Specifies the cost function to be used during registration. You can specify correlation ratio (default), least squares, normalized cross correlation, or normalized mutual information. To make this box available, you must first select Registration before colocalization.
VOI background subtraction	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 >= zero. The default is no subtraction.
Bin numbers	Specifies the number of histogram bins to be used in the 2D histogram. The default is the range of the data or 256, whichever is less, for nonfloating point images and 256 for floating point images.
Options	**Second iteration excluding subthresholded region—**Excludes the L-shape subthreshold region determined from the first iteration from the regression line calculations. By default, this check box is clear.
	Limit colocalization to pixels >= threshold—Uses, in red colocalization, only pixels with red values >= red threshold in the denominator sum and, in green colocalization, only pixels with green values >= green threshold in the denominator sum. The default value is not selected.

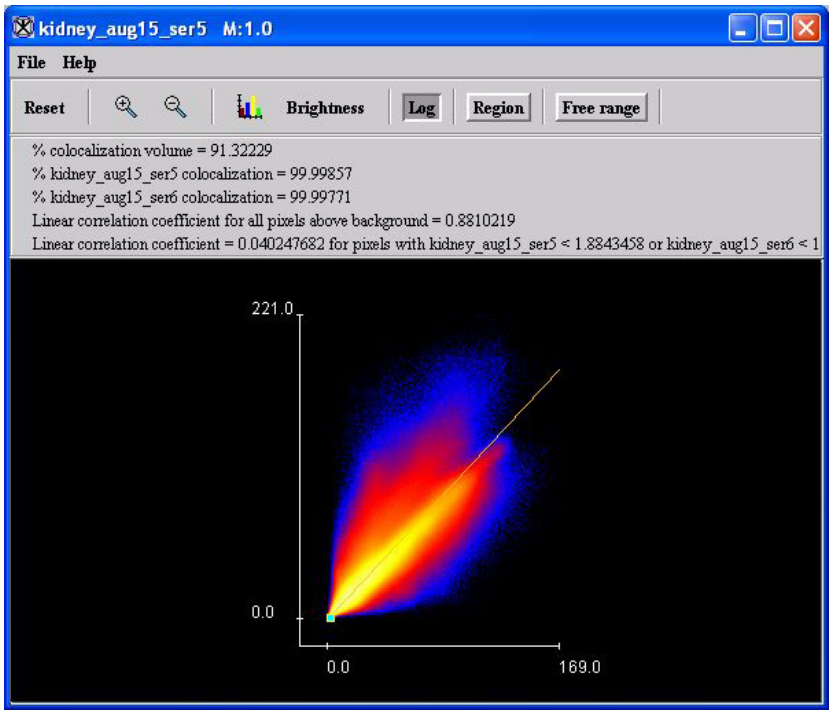
Figure 77. Orthogonal Regression Colocalization dialog boxes for (A) color and (B) grayscale images (continued)

Colocalization region	**Whole image—**Makes calculations on the whole image. By default, this option is selected.
	VOI region—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.
	Mask file—Makes calculations on only the pixels in the mask. Selecting this option enables the Choose mask file button.
	Choose mask file—Allows you to select a mask file.
Free range	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.
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.
Help	Displays online help for this dialog box.

Figure 77. Orthogonal Regression Colocalization dialog boxes for (A) color and (B) grayscale images (continued)

COLOCALIZATION HISTOGRAM WINDOW

The Colocalization Histogram window (Figure 78) contains a 2D histogram with a correlation line and a mouse-movable point on the correlation line. As the point is



File	**Close Colocalization**—Closes the colocalization window.
Help	**About Colocalization**—Displays online help for this window.
Reset	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.
Magnify image 2.0X	Magnifies the histogram by 2.
Magnify image 0.5X	Magnify the histogram by one-half.

Figure 78. 2D Histogram window

Displays Lookup Table (LUT)	Displays the Lookup Table (LUT) dialog box, which allows you to specify the LUT for this histogram.
Brightness	Displays the Brightness/Contrast dialog box, which allows you to adjust the brightness and contrast in the histogram.
Log	Toggles the histogram between showing counts and $\log_{10}$ (1+ histogram counts). The default is no log function.
Region	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 78. 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 >= threshold and green >= 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 >= threshold is not selected, the percent red colocalization (refer to Figure 79) 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.

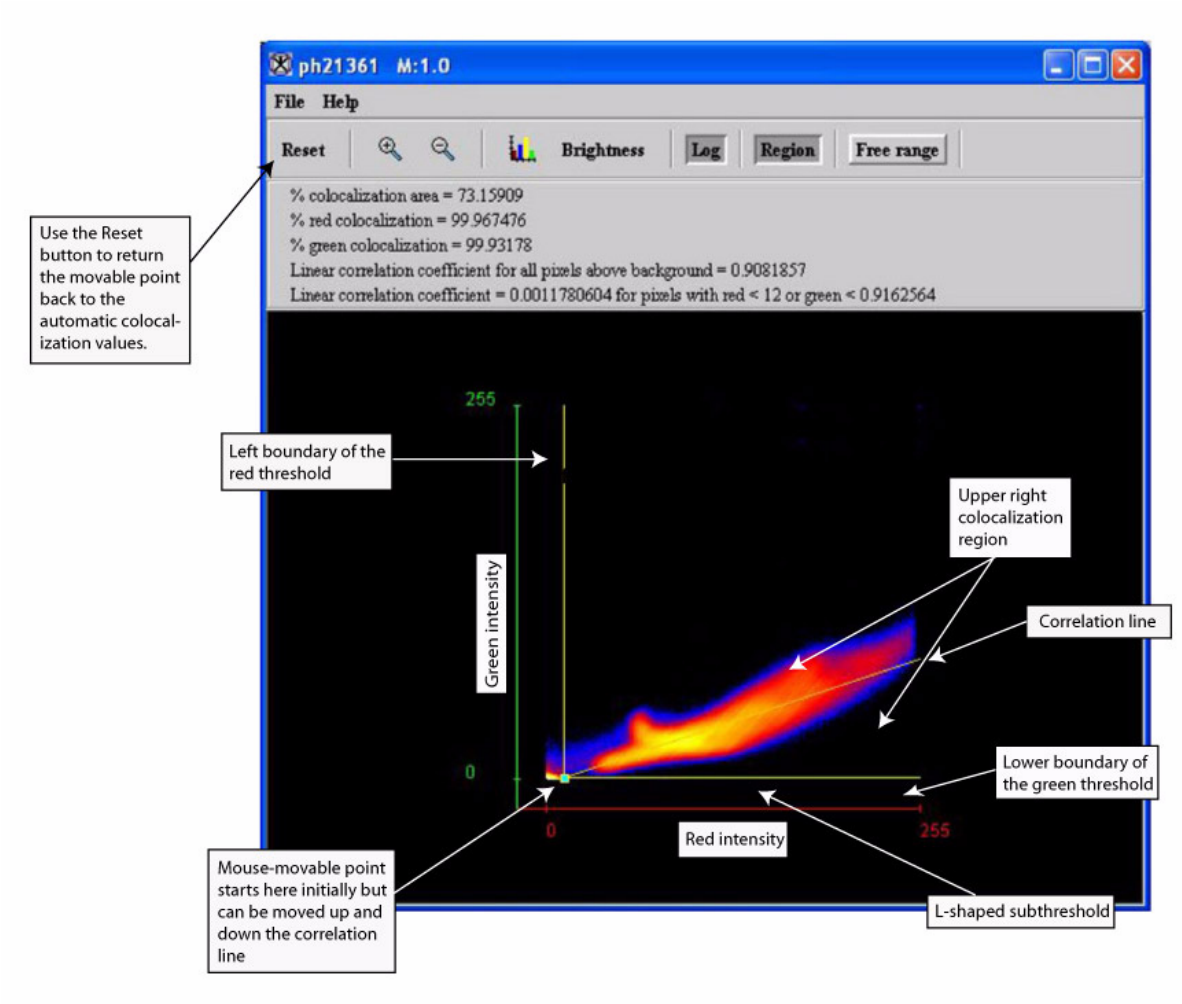
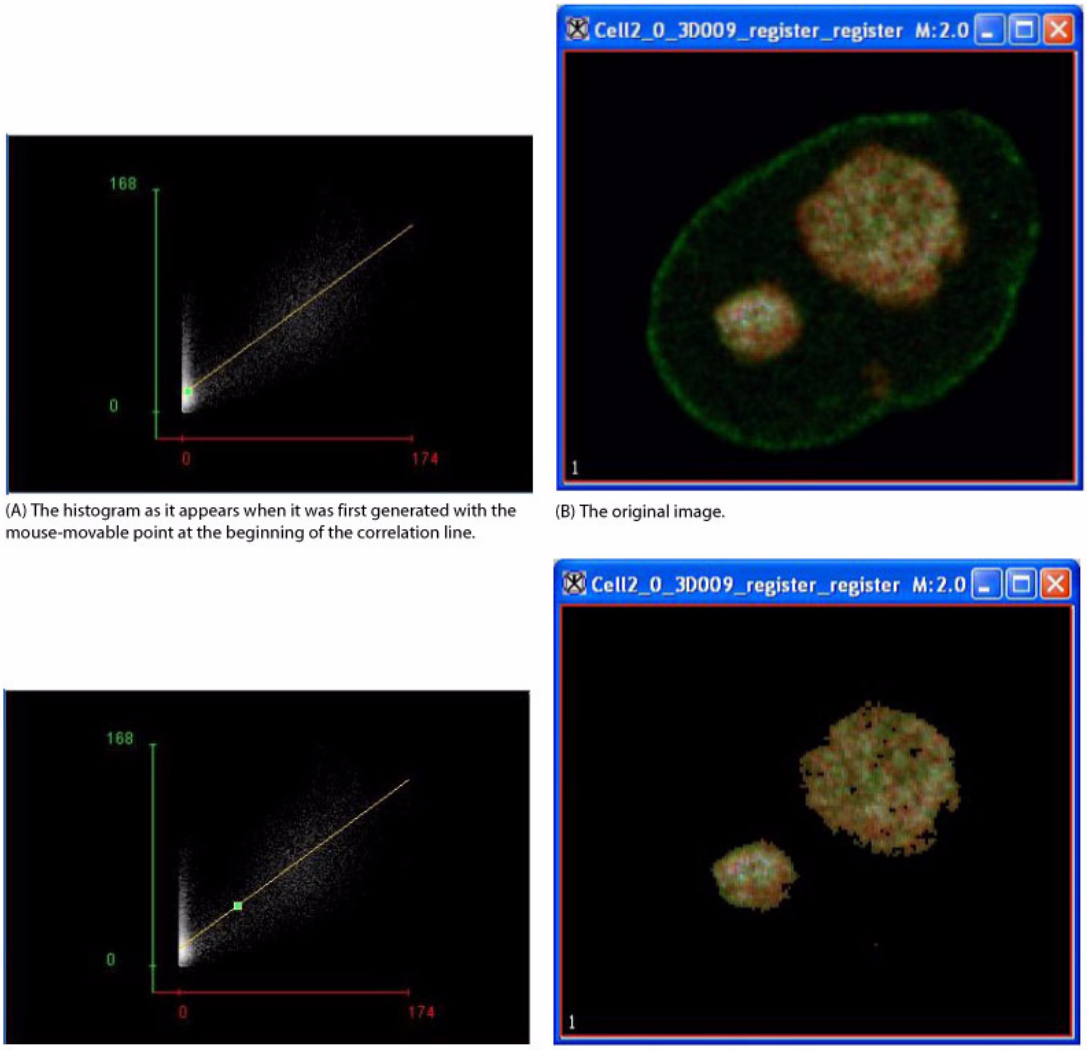


Figure 79. Colocalization Histogram window showing a histogram for a color image in which red and green color localization was done

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 >= 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 >= 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 >= the green threshold in the image or selected VOI.



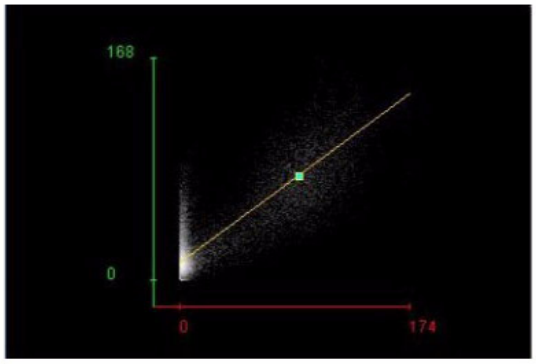
(A) The histogram as it appears when it was first generated with the mouse-movable point at the beginning of the correlation line.

(B) The original image.

(C) The histogram when you move the point up and down the correlation line. Notice on the right the changes to the image as you move the point.

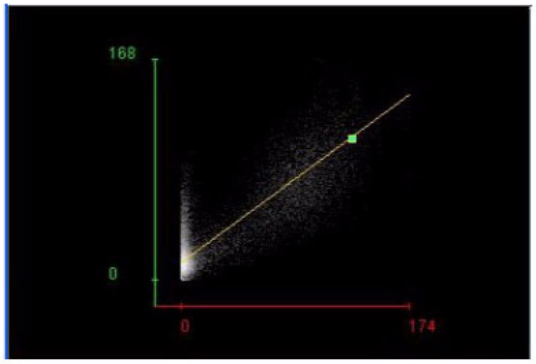
(D) The image for the histogram in C.

Figure 80. The effect on the image when dragging the mouse-movable point on the histogram (part 1)

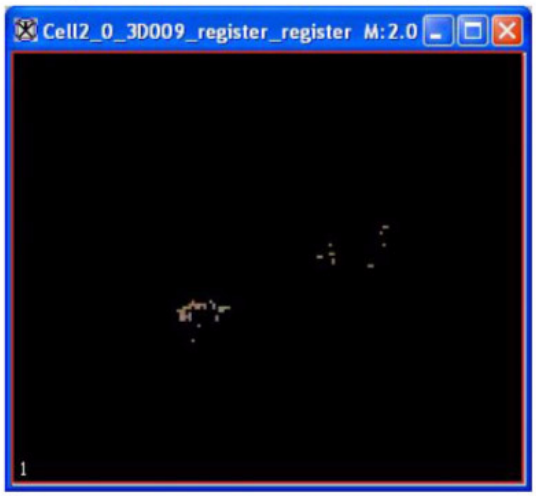


(A) Histogram after moving the point to the middle of the correlation line

(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 81. The effect on the image when dragging the mouse-movable point on the histogram (part 2)

Microscopy: Fluorescence Resonance Energy Transfer (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:

$$E = \frac{1}{\left[1 + \left(\frac{r}{Ro}\right)^6\right]}$$

EQUATION 4

where Ro, the Forster critical distance, is the distance at which $E = 0.5$.

We also have

$$E = 1 - \frac{FDA}{FD}$$

EQUATION 5

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 5 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 $2R_0$. 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.

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 prebleached image to the postbleached 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* prebleached image rather than on the original prebleached image.

To run the algorithm, you must first delineate a VOI on the postbleached 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 postbleached 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 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 $b2$ and $s2$ 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:

$$L = \left(\frac{(s2 - b2)}{(s1 - b1)}\right) \times P + \frac{(b2 \times s1 - b1 \times s2)}{(s1 - b1)}$$

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 82) appears.

Figure 82. 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 83) 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.

Prebleached image	Specifies the image *before* acceptor photobleaching.
Postbleached image	An image *after* acceptor photobleaching.
Add required donor fluorescence VOI	Adds a mandatory VOI to the postbleached image.
Add optional background VOI	Adds an optional background VOI on the postbleached image.
Add optional signal normalization VOI	Requires that a background VOI be used. Adds an optional signal normalization VOI on the postbleached image.
Color selection (only appears for color images)	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.
Registration before FRET‘	Registers the prebleached image to the postbleached image. The default is no registration. Selecting this check box enables the Cost function list.
Cost function	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.
Create registration image	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.
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.

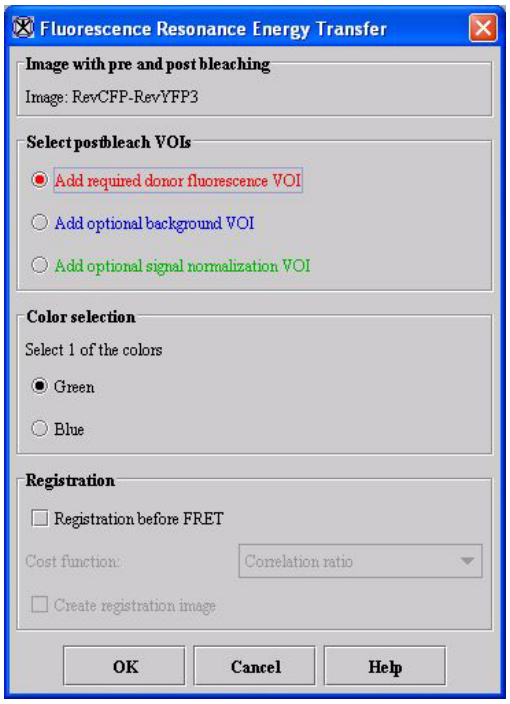
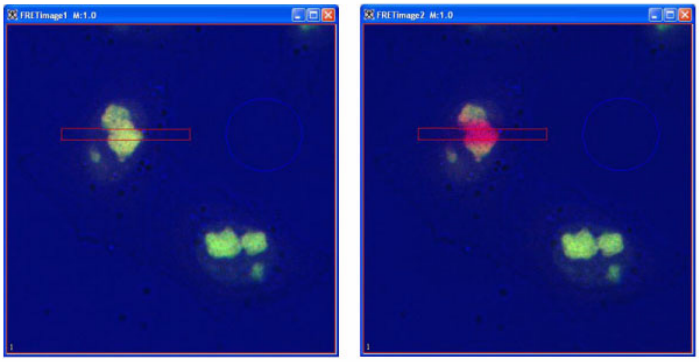


Figure 83. Fluorescence Resonance Energy Transfer (FRET) dialog box

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



(A) Prebleached image (B) Postbleached image

Figure 84. 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 85).

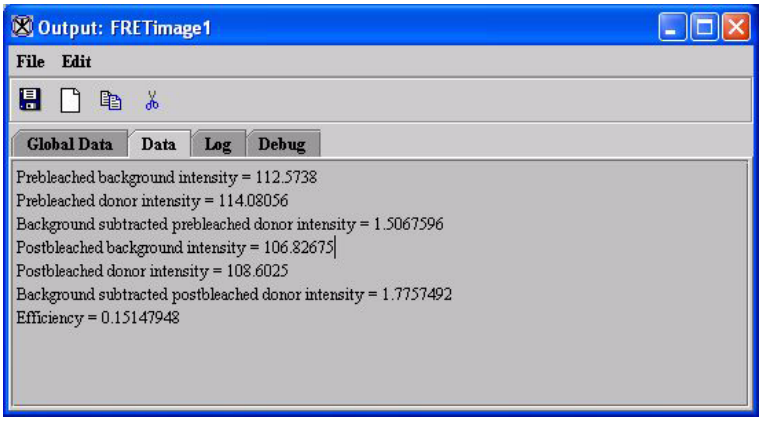


Figure 85. FRET data in the Data page of the Output window

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: 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. 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.

Background

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 243). 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 244. 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 Fa/Fo, in which Fa is the first slice after photobleaching and Fo 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 fluroescence.

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:

$R(t, \alpha, \beta, \Gamma) = \text{bottom} + \text{span} \times (1 - \Gamma e^{\alpha t} - (1 - \Gamma) e^{\beta t})$ with $0 > \alpha > \beta$ 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 micrometers² 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 6

$$I(t) = I(\text{final})\left(1 - \frac{w^2}{\left(\sqrt{w^2 + 4\pi t Deff}\right)}\right)$$

where

t is time

I is the photobleached fluorescent intensity

w is the width of the bleaching along the membrane in micrometers (um)

Deff is the effective 1D diffusion to coefficient in micrometers² 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, $201thalf$ 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 7

$$\frac{du0f}{dt} = -D1 \times uof + D2 \times u1f - ka \times u0f + kd \times u0b$$

$$\frac{du1f}{dt} = D1 \times u0f - D2 \times u1f - ka \times u1f + kd \times u1b$$

$$\frac{du0b}{dt} = ka \times u0f - kd \times u0b$$

$$\frac{du1b}{dt} = ka \times u1f - kd \times u1b$$

$$D1 = \left(\frac{Fa}{(Fa + F0 - Fa)}\right)Dt = \left(\frac{Fa}{F0}\right)Dt$$

$$D2 = \left(\frac{(F0 - Fa)}{(Fa + F0 - Fa)}\right)Dt = \left(\frac{(F0 - Fa)}{F0}\right)Dt$$

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 8

$$\begin{bmatrix} -D1 - ka - e & D2 & kd & 0 \\ D1 & -D2 - ka - e & 0 & kd \\ ka & 0 & -kd - e & 0 \\ 0 & ka & 0 & -kd - e \end{bmatrix}$$

This gives the equation:

$$^{4} + (D1 + D2 + 2ka + 2kd) \times e^{2} + (D1 \times ka + 2D1 \times kd + D2 \times ka + 2D2 \times kd + ka^{2}$$
$$- 2ka \times kd + kd^{2} \times e^{2} + D1 \times ka \times kd + D1 \times kd^{2} + D2 \times ka \times kd + D2 \times kd^{2}) \times e = 0$$

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

$$\alpha = -s1 + s2$$
$$\beta = -s1 - s2$$

where

$$s1 = \frac{(D + ka + kd)}{2}$$

$$s2 = \frac{\sqrt{((D + ka + kd)^2 - 4D \times kd)}}{2}$$

The eigenvalue $e = 0$ gives the eigenvector:

$$\begin{matrix} 1 \\ \left(\frac{D1}{D2}\right) \\ \left(\frac{ka}{kd}\right) \\ \left(\frac{(D1 \times ka)}{(D2 \times kd)}\right) \end{matrix}$$

The eigenvalue $e = -ka - kd$ gives the eigenvector:

$$\begin{matrix} 1 \\ \left(\frac{D1}{D2}\right) \\ -1 \\ -\left(\frac{D1}{D2}\right) \end{matrix}$$

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 v1, v2, v3, and v4. Let the matrix V be given by:

$$\begin{bmatrix} v11 & v21 & v31 & v41 \\ v12 & v22 & v32 & v42 \\ v13 & v23 & v33 & v43 \\ v14 & v24 & v34 & v44 \end{bmatrix}$$

Let the vector C be

$$\begin{bmatrix} c1 \\ c2 \\ c3 \\ c4 \end{bmatrix}$$

Remembering that $u0$ is the vector of initial conditions, we wish to solve:

$$VC = u0$$

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 9

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 10

yields equation 6:

EQUATION 11

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 12

$$\frac{D1}{D2} \times c1 + \frac{D1}{D2} \times c2$$

$$+ \left(\frac{D1 \times (kd + \alpha)}{(D2 \times kd + \alpha \times kd + \alpha \times D2 + \alpha \times ka + \alpha^2)} \right) \times c3$$

$$+ \left(\frac{D1 \times (kd + \beta)}{(D2 \times kd + \beta \times kd + \beta \times kd + \beta \times kd + \beta \times D2 + \beta \times ka + \beta^2)} \right) \times c4$$

$$= \left(\frac{kd}{(ka + kd)} \right) \left(\frac{Fa}{F0} \right) (2)$$

$$\left(\frac{(D1 \times ka)}{(D2 \times kd)} \right) \times c1 + \left(\frac{-D1}{D2} \right) \times c2$$

$$+ \left(\frac{(D1 \times ka)}{(D2 \times kd \times \alpha \times kd + \alpha \times D2 + \alpha \times ka + \alpha^2)} \right) \times c3$$

$$+ \left(\frac{D1 \times ka}{(D2 \times kd + \beta \times kd + \beta \times D2 + \beta \times ka + \beta^2)} \right) \times c4$$

$$= \left(\frac{ka}{(ka + kd)} \right) \left(\frac{Fa}{F0} \right) (4)$$

which simplifies to:

EQUATION 13

Multiplying equation 1 by $\frac{-D1}{D2}$ and adding to equation 2 yields:

EQUATION 14

$$\left[\frac{(D1 \times (ka + kd + \alpha)}{(D2 \times kd + \alpha \times kd + \alpha \times D2 + \alpha \times ka + \alpha^2)} - \left(\frac{D1 \times (ka + kd + \alpha)}{D2 \times (kd + \alpha)}\right)\right] \times c3$$
$$+ \left[\frac{D1 \times (ka + kd + \beta)}{(D2 \times kd + \beta \times kd + \beta \times D2 + \beta \times ka + \beta^2)} - \frac{D1 \times (ka + kd + \beta)}{(D2 \times (kd + \beta))}\right] \times c4 = \frac{Fa}{F0}$$

$$\left[\frac{-D1 \times \alpha \times (ka + kd + \alpha)^2}{(D2 \times (kd + \alpha) \times (D2 \times kd + \alpha \times kd + \alpha \times D2 + \alpha \times ka + \alpha^2))}\right] \times c3$$
$$+ \left[\frac{-D1 \times \beta \times (ka + kd + \beta)^2}{(D2 \times (kd + \beta) \times (D2 \times kd + \beta \times kd + \beta \times D2 + \beta \times ka + \beta^2))}\right] \times c4 = \frac{Fa}{F0}(7)$$

Multiplying equation 8 by $\frac{-(ka + kd + \alpha)}{(kd + \alpha)}$ and adding to equation 7 yields:

$$\left[\frac{-\alpha \times D1 \times (ka + kd + \alpha)}{(D2 \times (D2 \times kd + \alpha \times kd + \alpha \times D2 + \alpha \times ka + \alpha^2))}\right] \times c3$$
$$+ \left[\frac{-\beta \times D1 \times (ka + kd + \beta)}{(D2 \times (D2 \times kd + \beta \times kd + \beta \times D2 + \beta \times \quad + \beta^2))}\right] \times c4 = \left(\frac{kd}{(ka + kd)}\right) \times \frac{Fa}{F0}$$

EQUATION 15

$$c4 = \alpha \times \left(\frac{Fa}{F0}\right) \times D2 \times (D2 \times kd + \beta + kd + \beta \times D2 + \beta \times ka + \beta^2)$$
$$\times \frac{(kd + \beta)}{[(ka + kd) \times \beta \times (\beta - \alpha) \times D1 + (ka + kd + \beta)]}$$

This simplifies to:

EQUATION 16

$$c4 = \frac{\left(\frac{Fa}{F0}\right) \times D2 \times kd + (-D + ka + kd - 2S2)}{[D \times (ka + kd) \times 4S2]}$$

Substituting c4 back into equation 8 yields c3:

EQUATION 17

$$c3 = \frac{\beta \times \left(\frac{Fa}{F0}\right) \times D2 \times (kd + \alpha) \times (D2 \times kd + \alpha \times kd + \alpha \times D2 + \alpha \times k\alpha + \alpha^2)}{[\alpha \times (\alpha - \beta) \times D1 \times (ka + kd) \times (ka + kd + \alpha)]}$$

This simplifies to:

EQUATION 18

$$c3 = \frac{\left(\frac{Fa}{F0}\right) \times D2 \times kd \times (D - ka - kd - 2S2)}{[D \times (ka + kd) \times 4S2]}$$

Putting c3 and c4 in equation 5 and solving for c1 yields:

EQUATION 19

$$c1 = \frac{\left(\frac{Fa}{F0}\right) \times D2 \times kd}{[D \times (ka + kd)]}$$

Putting c1, c3, and c4 into equation 4 and solving for c2 yields c2 = 0.

Thus, the solution is:

EQUATION 20

$$u0f = c1 + c3e^{(\alpha \times t)} + c4e^{(\beta \times t)}$$

$$= \left[\left(\frac{Fa}{F0}\right) \times \frac{D2 \times kd}{(D \times (ka + kd))}\right] \times \left[\frac{1 + D - ka - kd - 2S2}{4S2e^{\alpha \times t}} + \frac{(-D + ka + kd - 2S2)}{4S2e^{\beta \times t}}\right]$$

The above equation is normalized by dividing by

EQUATION 21

$$\frac{\left(\frac{Fa}{F0}\right) \times D2 \times kd}{(D \times (ka + kd))}$$

The coefficient of exp(alpha*t) in the normalized equation equals -gamma so $\Gamma = \frac{(-D + ka + kd + 2S2)}{4S2}$ and the normalized equation can be written as:

EQUATION 22

$$uof \text{ normalized} = 1 - \Gamma e^{(\alpha \times t)} - (1 - \Gamma) e^{(\beta \times t)}$$

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.bioinformaticsservices.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 88) opens.

Add required photobleached VOI	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.
Add required whole organ VOI	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.
All optional background VOI	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.
Color selection (only appears for color images)	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.
First slice after photobleaching	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.
2D narrow band	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.
2D circle	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.
1D	Assumes pure 1D diffusion.

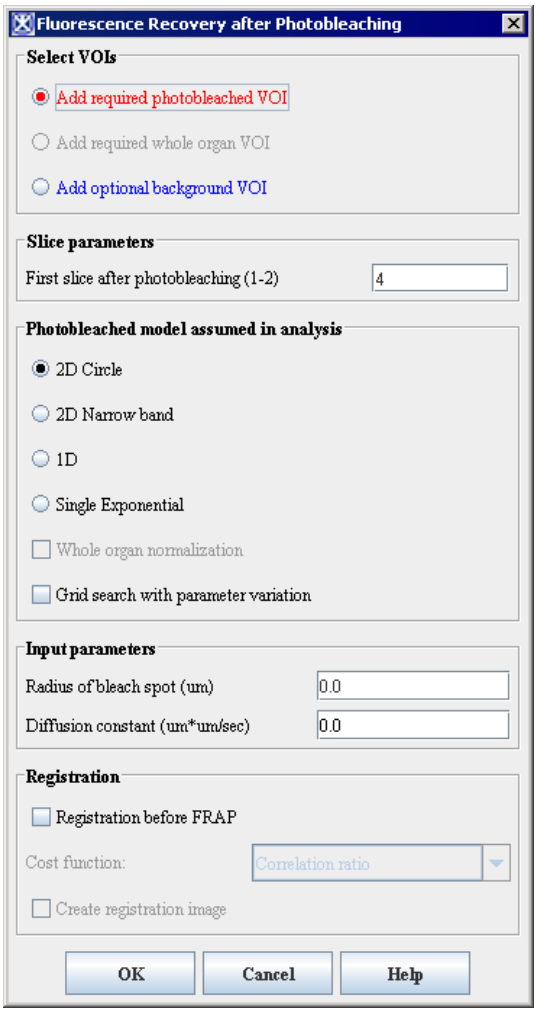


Figure 86. Fluorescence Recovery after Photobleaching dialog box

Single exponential	Assumes recovery according to a single exponential.
Grid search with parameter variation	Conducts a double check by grid search.
Registration before FRAP	Registers the image slices that are not aligned. Select this item so that slice alignment occurs before FRAP. The default is no registration.
Cost function	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.
Create registration image	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.
OK	Applies the algorithm according to the specifications made in the 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.

Figure 86. 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.

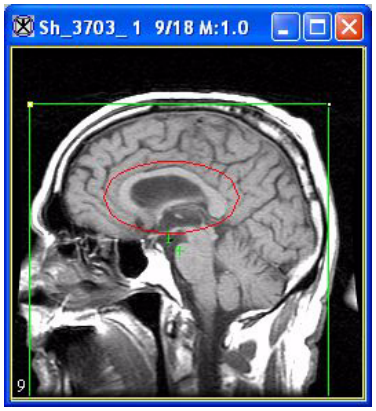


Figure 87. An image with a required photobleached VOI in red and a whole organ VOI in green

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 88), 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)

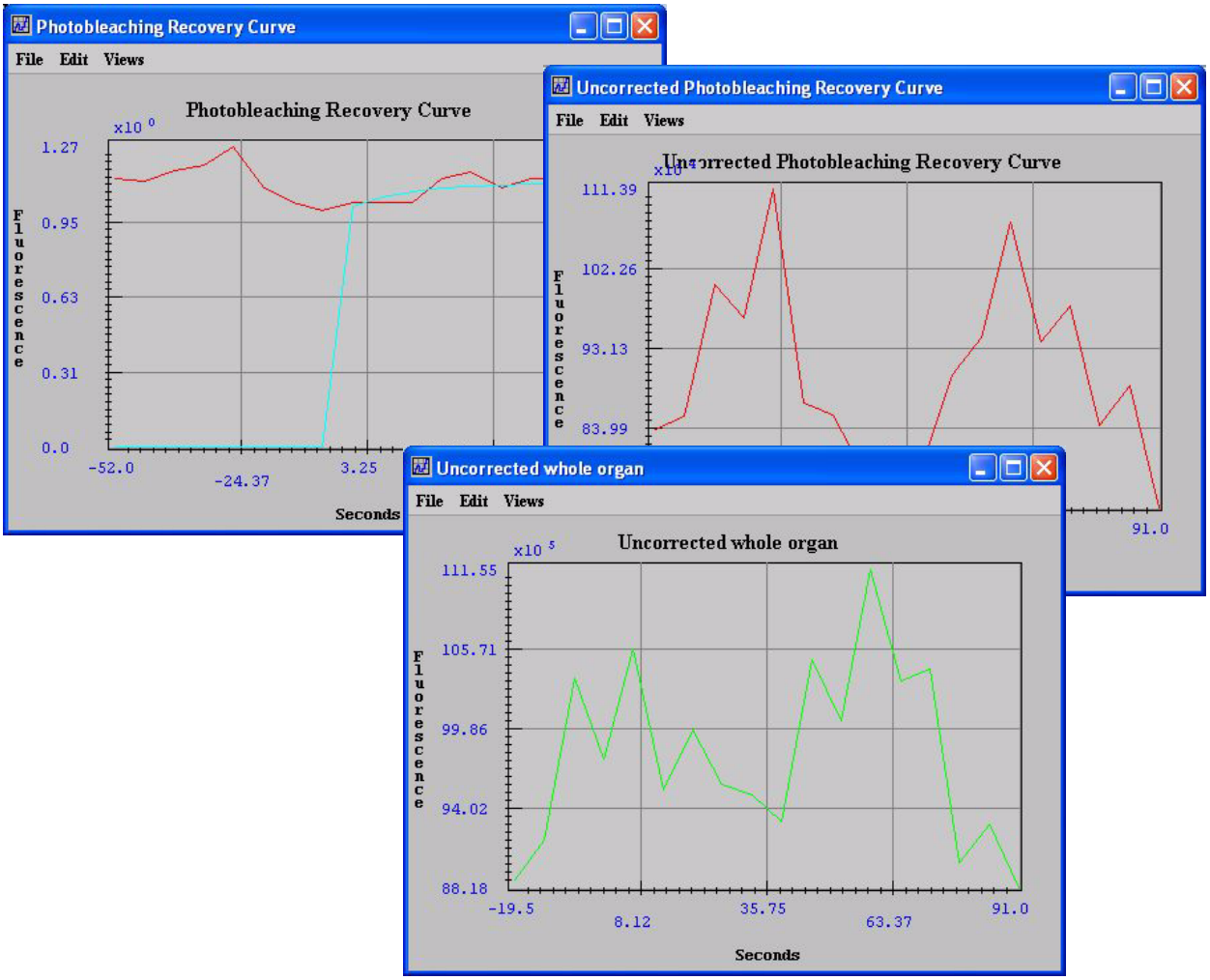


Figure 88. Graph windows: Photobleaching Recovery Curve, Uncorrected Photobleaching Recovery Curve, and Uncorrected Whole Organ windows

File	**Open Graph (Ctrl0)**—Allows you to open a previously saved graph. When you select this command, the Open Graph Data dialog box appears.
	Save Graph (CtrlS)—Allows you to save a graph as a .PLT file. When you select this command, the Save dialog box appears.
	Print Graph (CtrlP)—Allows you to print the graph. When you select this command, the Print dialog box appears.
	Close Graph (CtrlX)—Closes the graph immediately.
Edit	**Delete Function**—Allows you to delete a specified function on the graph unless the graph contains only one function.
	Copy Function—Allows you to copy the function that you specify.
	Paste Function—Allows you to paste a previously copied function to the graph.
Views	**Modify Graph Features**—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.
	Reset Range to Default—Changes the range displayed in the graph to the default range.
	Reset Graph to Original—Changes the range displayed in the graph to the original range.
Graph	Displays the amount of fluorescence over time (seconds).

Figure 88. Graph windows: Photobleaching Recovery Curve, Uncorrected Photobleaching Recovery Curve, and Uncorrected Whole Organ windows (continued)

- Uncorrected Photobleaching Recovery Curve window (Figure 88)
- Uncorrected Whole Organ window (Figure 88)

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 89) appears.

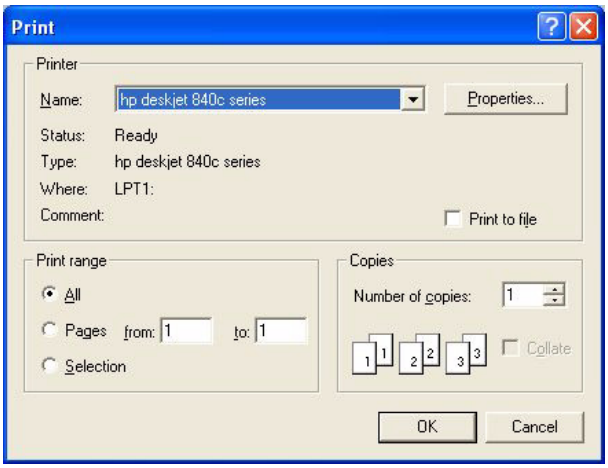


Figure 89. Print dialog box

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 90. Modify Graph Features command on the Views menu in a graph window

The Modify Graph dialog box (Figure 91) 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

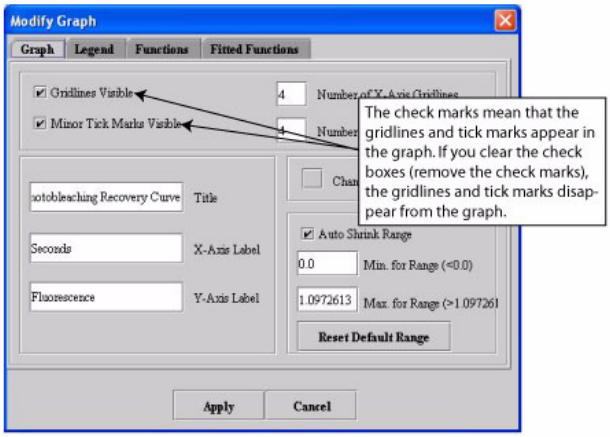


Figure 91. Modify Graph dialog box showing the Graph page

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 92) 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.

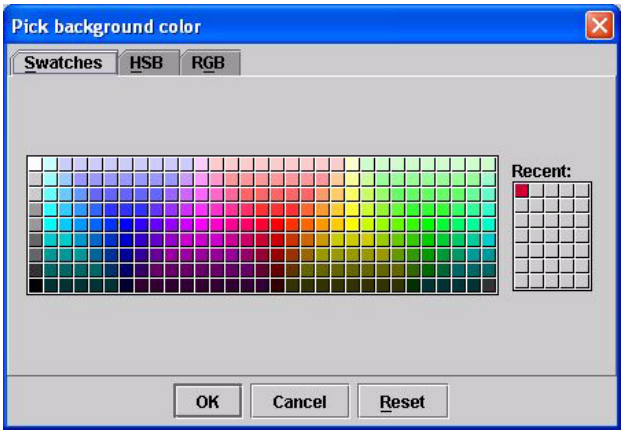
Change background color	Allows you to change the background color of the grid. By default, the background color is light gray. When you click this button, the Pick Background Color dialog box opens. 
Auto shrink range	Allows you to adjust the display of the *Y* axis of the graph. By default, this check box is selected.
Minimum for range	Specifies the minimum value for the *Y* axis, which must be 0.0 or less. The default value is 0.0.
Maximum for range	Specifies the maximum value for the X axis, which must be 1.0 or more. The default value is 1.0.
Reset default range	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.
Apply	Applies immediately the changes to the graph according to the specifications in this dialog box but keeps this dialog box open.
Cancel	Disregards any changes that you made in the dialog box and closes this dialog box.
Help	Displays online help for this dialog box.

Figure 92. 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 93) 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.

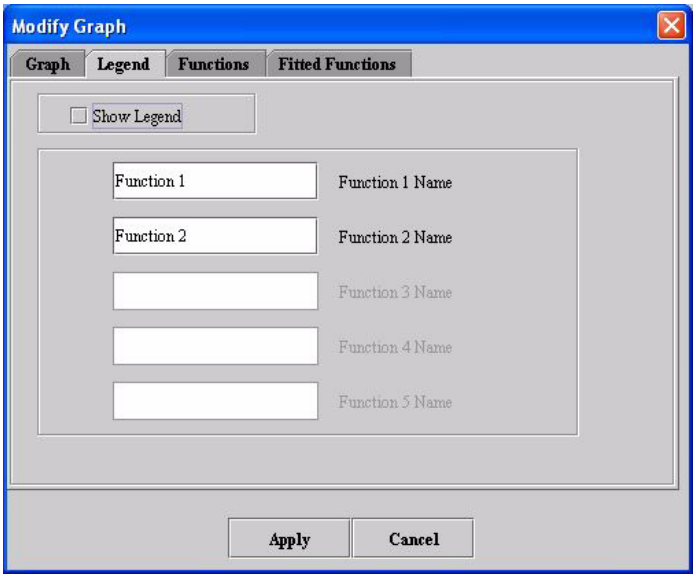


Figure 93. Legend page in the Modify Graph dialog box

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 94) appears.

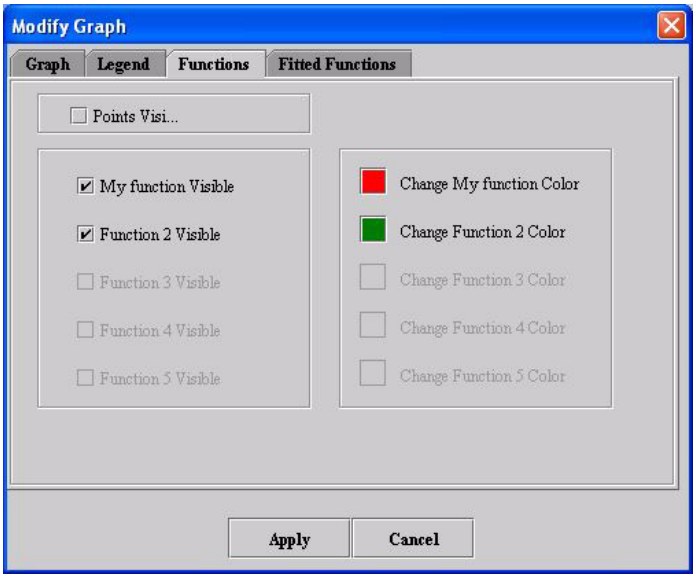


Figure 94. Functions page in the Modify Graph dialog box

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 95) 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.

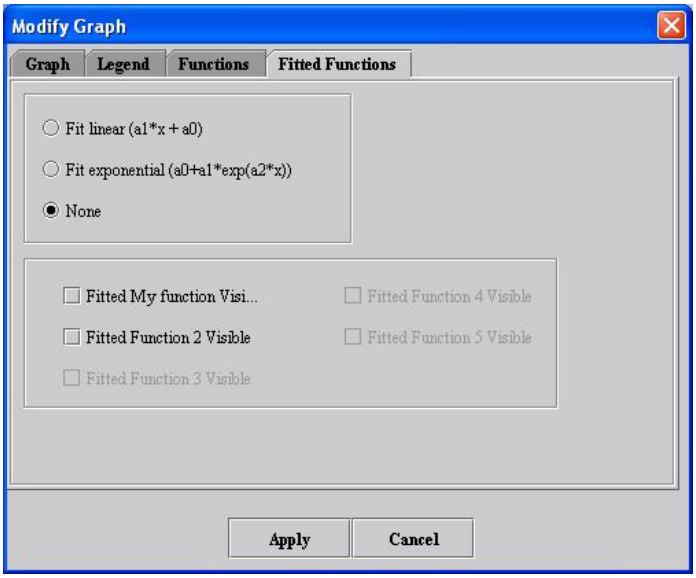


Figure 95. Fitted Functions page in the Modify Graph window

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 264 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 260.

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 23 – 25:

EQUATION 23

$$\lambda^{(k+1)}(x,y,z) = \lambda^{(k)}(x,y,z)[\mu(x,y,z)/(\lambda^{(k)}(x,y,z) \bullet h^{(k)}(x,y,z)) \bullet h^{(k)}(-x,-y,-z)]$$

EQUATION 24

$$h'^{(k+1)}(x,y,z) = \lambda^{(k)}(x,y,z)[\mu(x,y,z)/(\lambda^{(k)}(x,y,z) \bullet h^{(k)}(x,y,z)) \bullet \lambda^{(k)}(-x,-y,-z)]$$

EQUATION 25

$$h^{(k+1)}(x,y,z) = constraints(h'^{(k+1)}(x,y,z))$$

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 23 – 25.

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 261;
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 62 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 96.

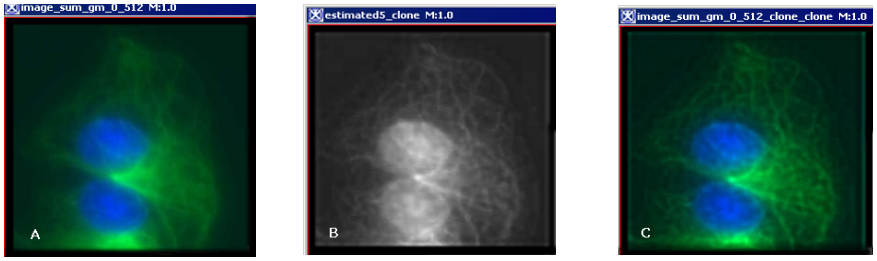


Figure 96. The original color image (A), the intermediate grayscale image (B), and the final estimate color image after 10 iterations (C).

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 97 that shows the Jupiter image.

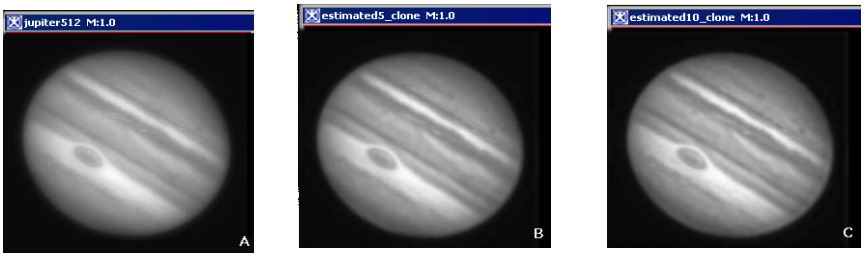


Figure 97. The original image of Jupiter in grayscale (A), the estimate image after five iterations (B), and the final estimate after 10 iterations (C).

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

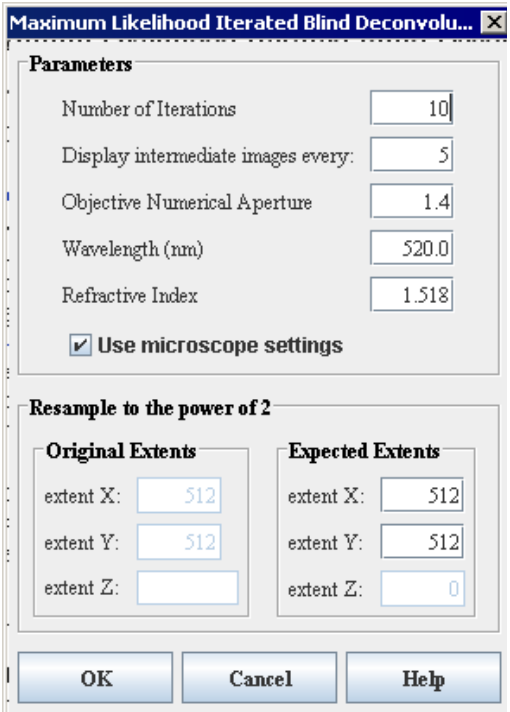
http://micro.magnet.fsu.edu/primer/digitalimaging/deconvolution/deconalgorithms.html

http://www.hamangia.freeserve.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.



Number of Iterations	Enter the number of times to iterate through the blind-deconvolution process. The default value is 10 iterations.
Display intermediate image every	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.
Objective Numerical Aperture	Enter the numerical aperture of the microscope objective used in the imaging system. The default value is 1.4.
Wavelength (nm)	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.
Refractive Index	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 98. The Maximum Likelihood Iterated Blind Deconvolution dialog box

Use microscope settings	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.
Resample to the power of 2	
Original Extents (extent X, extent Y, and extent Z)	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.
Expected Extents (extent X, extent Y, and extent Z)	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.
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.

Figure 98. 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 99.

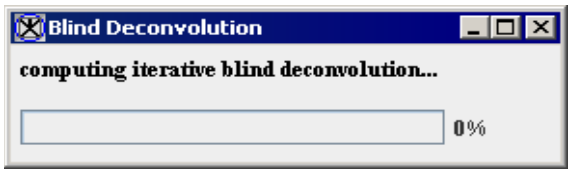


Figure 99. A status pop-up window.

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 100, and then, disables the checkbox. Refer to "PSF constraints" on page 261 to learn more about PSF constraints.

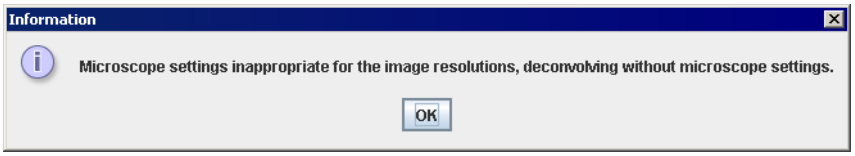


Figure 100. The PSF warning.

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 imaged selected.

Figure 101 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 264 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 261 uses the Objective Numerical Aperture, Wavelength, and Refractive Index parameters provided by the user to determine the size and shape of the PSF.

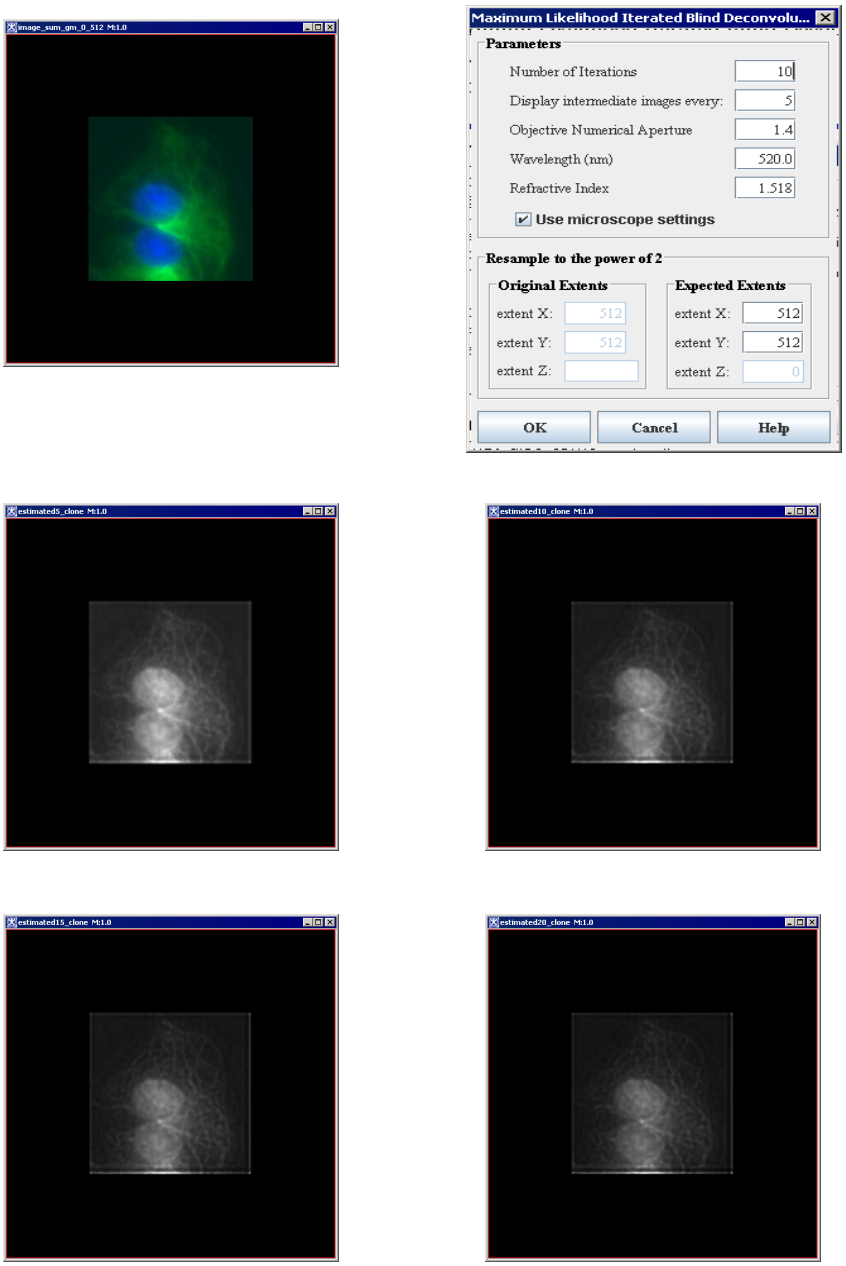


Figure 101. The Blind Deconvolution algorithm was applied on the color image (first slide) with the default parameters values (shown on the second slide). The intermediate grayscale images are also shown.

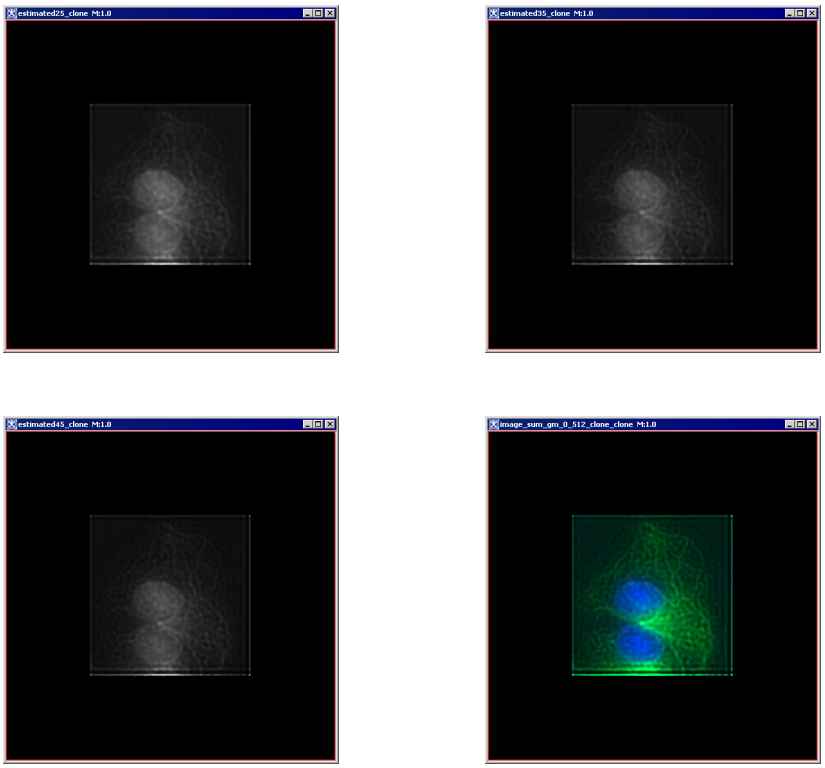


Figure 101. The Blind Deconvolution algorithm was applied on the color image (first slide) with the default parameters values (shown on the second slide). The intermediate grayscale images are also shown.

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 paremeters 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 278

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" on page 284.

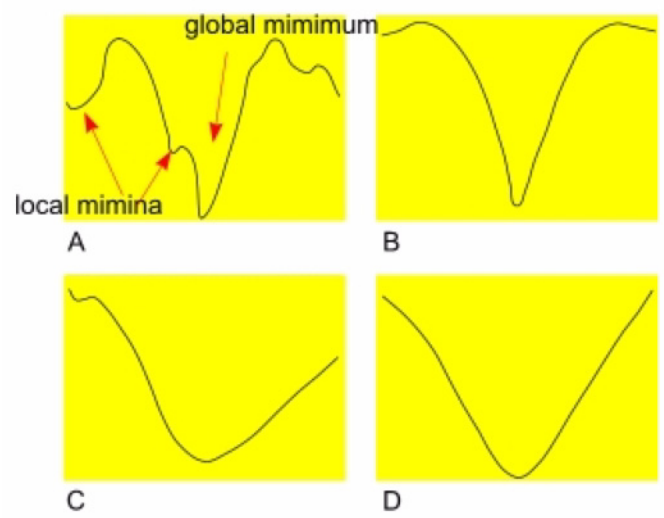


Figure 102. The plots of the Correlation Ratio cost function versus some of the individual parameter values. In each plot (A, B, C, and D), a single parameter is varied while the other are kept constant

IMAGE TYPES

You can apply Optimized Automatic Registratiom 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 272.
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 "Resampling Interpolation" on page 278.
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 273.

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 278.

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/n3 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 26:

EQUATION 26

$$b(x,y) = \begin{cases} 1, & \textit{for points in the image} \\ 0, & \textit{for background points} \end{cases}$$

2. Then, an area of an image can be calculated as

EQUATION 27

$$A = \int\int b(x,y)dxdy$$

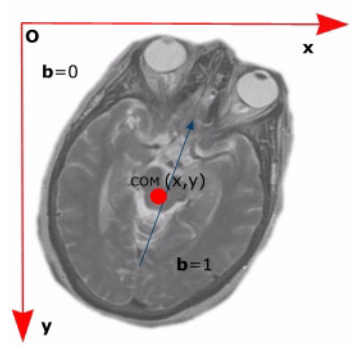


Figure 103. Calculating the center of mass (COM)

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.

EQUATION 28

$$x_{COM} = \frac{\int\int xb(x,y)dxdy}{\int\int b(x,y)dxdy}$$

$$y_{COM} = \frac{\int\int yb(x,y)dxdy}{\int\int b(x,y)dxdy}$$

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 273;
- Aligns the COM of both images, which is an initial transfomation 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 104.

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 290.

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.

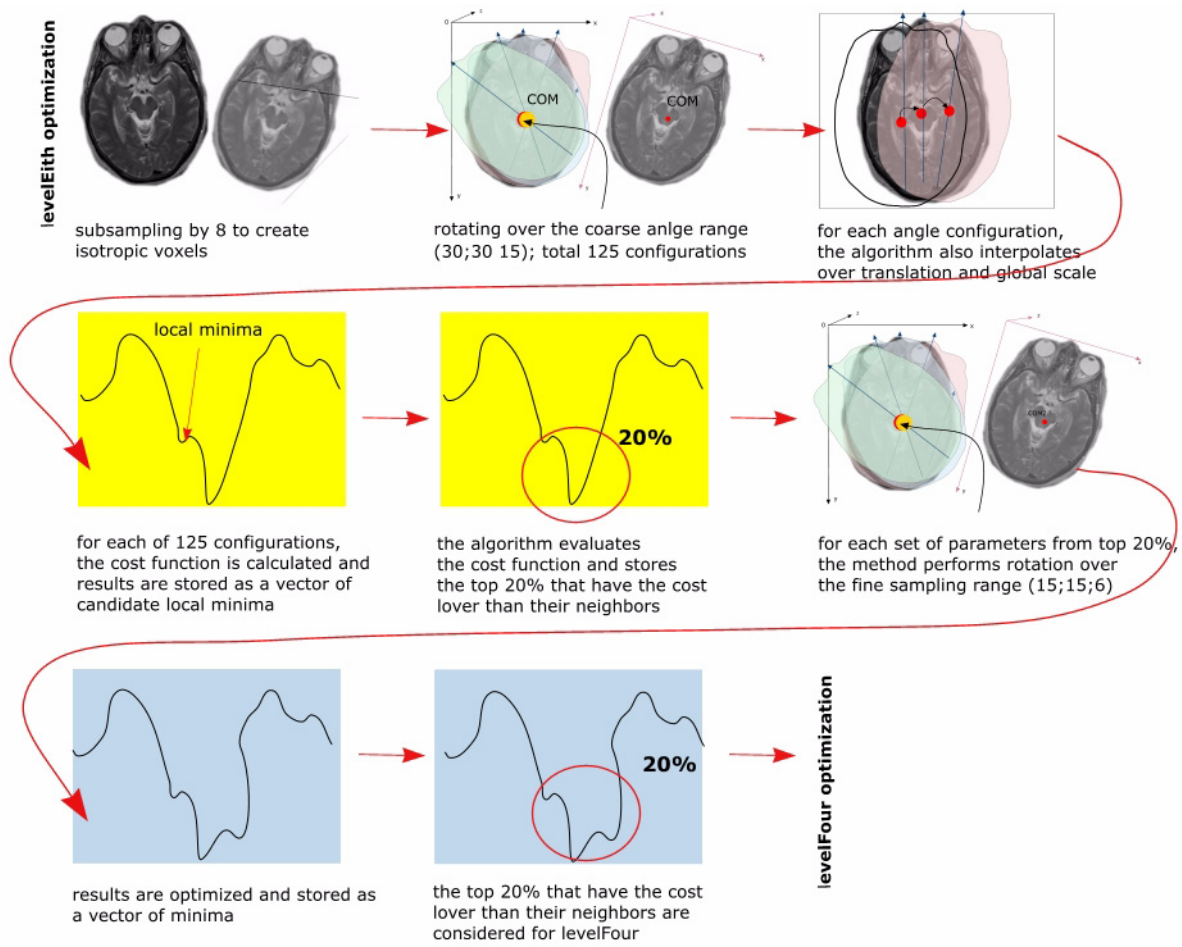


Figure 104. levelEight optimization

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 "Optimized Automatic Registration dialog box options" on page 294 and "Degrees of freedom" on page 290

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 "Optimized Automatic Registration dialog box options" on page 294.

Resampling Interpolation

The Optimized Automatic Registration method resampes 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 following interpolation methods are available in this implementation of the registration technique, including Trilinear, B-spline 3-rd order, B-spline 4-th order, Cubic Lagrangian, Quintic Lagrangian, Heptic Lagrangian, and Windowed *sinc*.

Trilinear Interpolation

Trilinear interpolation computes values located between existing voxel values by linearly weighting the eight closest neighboring values. Specifically, Figure 105 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 29

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

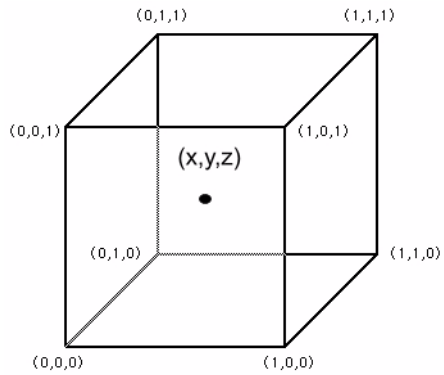


Figure 105. Trilinear interpolation

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 30.

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

EQUATION 30

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

In Equation 30, $N_{i,k}$ 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 31

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

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

Here, t(i) represents an index that refer to the control points and t(i) are generally referred as *knot points*, see Figure 106. 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 32.

EQUATION 32

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

For example: [o,o,o,o,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.

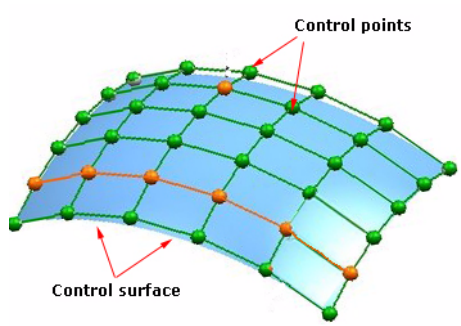


Figure 106. B-spline interpolation

Lagrange interpolations

To obtain higher-order interpolations, the Optimized Automatic Registration algorithm uses Lagrange interpolations. The Lagrange kernel of degree N-1 for N x N region can be defined by Equation 33.

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

EQUATION 33

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

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

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 34:

EQUATION 34

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

The Lagrange kernel for N=5 supporting points results in quadratic polynomials as shown in Equation 35 below:

EQUATION 35

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

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 107 below shows the Lagrange kernels for N=4 and N=5, respectively.

- ${}^{Lagra}h_4(x)$ 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 ${}^{Lagra}h_5(x)$, is not C0-continuous.

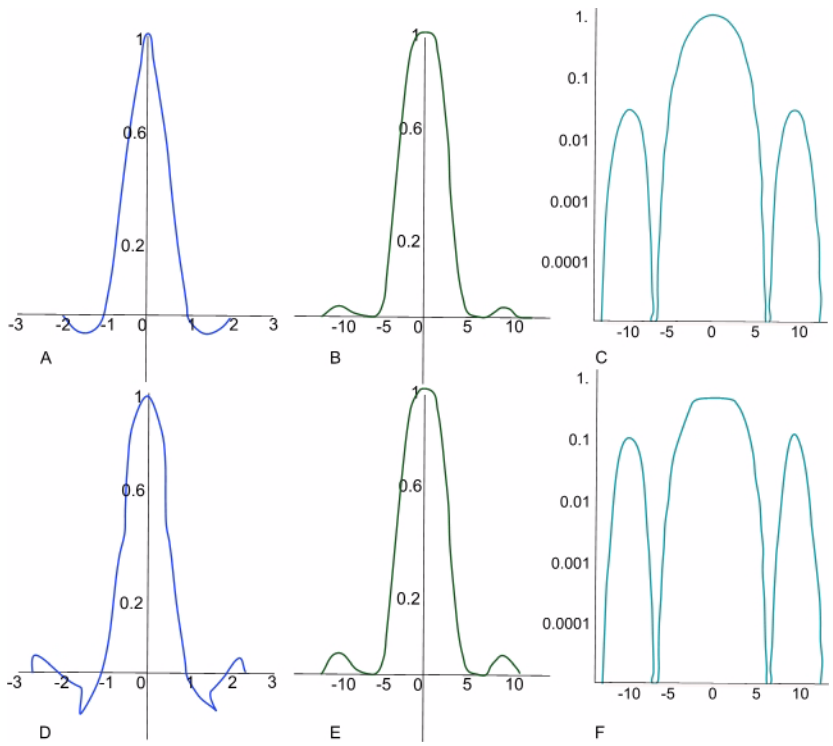


Figure 107. Lagrange third-order interpolation, N = 4: (A) - Kernel, (B) - Magnitude of Fourier transform, (C) - Logarithmic plot of magnitude. Lagrange fourth-order interpolation, N = 5, (D) -Kernel, (E) - Magnitude of Fourier transform, (F) - Logarithmic plot of magnitude

Therefore, the amplitude of the sidelobe is raised up to 10% as shown in Figure 107-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.

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 36:

EQUATION 36

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

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 37:

EQUATION 37

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

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 37 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 apodizating the Hann window function, refer to Equation 38:

EQUATION 38

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

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

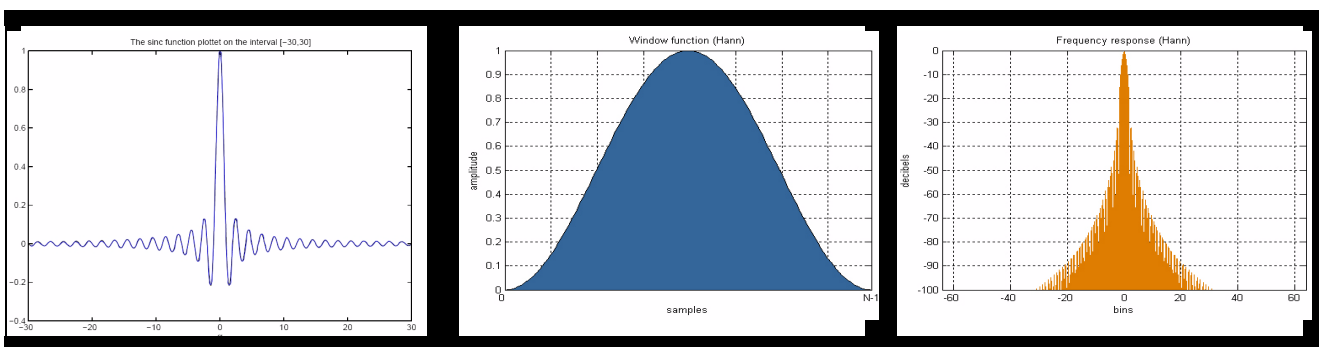


Figure 108. T y=sinc(a), where a=px, xÎ [-30,30]; a sample Hann function and its frequency response. Here, N represents the width, in samples, of a discrete-time window function. Typically, it is an integer power-of-2, such as 2^{10} = 1024.

Cost Functions

A *similarity* or *cost function* measures the similarity between two images. During the 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 page 285
- Least Squares, refer to page 286
- Normalized Cross Correlation, on page 287
- Normalized Mutual Information, see page 289

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

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 39

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

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.

Least Squares

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

EQUATION 40

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

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 41:

EQUATION 41

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

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 42

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

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 = sum(RI)/(count-1) - (sumR*sumI)/((count-1)*count)
varR = sum(R^2)/(count-1) - (sumR*sumR)/((count-1)*count)
varI = sum(I^2)/(count-1) - (sumI*sumI)/((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 43

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

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 44

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

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 44 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:

- [1]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 45

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

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:

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

EQUATION 46

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

- Affine transformations include translations, rotations, scales, and/or skewing parameters. They preserve straight lines but necessarily not 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 47

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

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 48

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

REFERENCES

Bjórn 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 Philipe 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 TR00MJ2. http://www.fmrib.ox.ac.uk/fsl/flirt

Joseph V. Hajnal, Derek L.G. Hill, David J. Hawkes [2001]. *Medical Image Registration*, CRC Press, London, New York, Washington DC.

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 Algorithm

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 “Optimized Automatic Registration dialog box options” on page 294.

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 109.

Recommendation: First, use the default algorithm settings. And then, after completing the first registration run, modify them depending on the result you receive.

A B C

Figure 109. The input image (A), reference image (B) and result image (C). Registration was performed using the default algorithm settings.

Optimized Automatic Registration dialog box options

Input options	
Register	Select the reference image from the list. The list alphabetically displays all opened images.
Degrees of freedom	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.
Interpolation	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
Cost function	Specify a cost function which will be used to evaluate registration. The list includes: Least squares, Correlation ratio, Normalized cross correlation, Normalized mutual information
Use the max of the min resolutions of the two datasets when resampling	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.

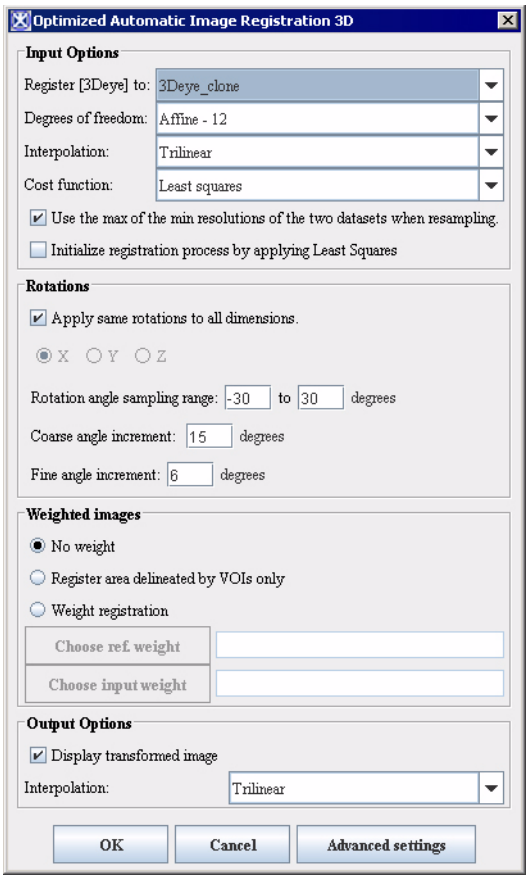


Figure 110. The Optimized Automatic Registration dialog box options

Initiate registration process by applying Least Squares	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.
Rotations	
Use this option if there is a need to make manual adjustments to registration.	
Apply same rotations to all dimensions	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.
Coarse angle increment and Fine angle increment	
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.	
Weighted Images	This gives weights for certain areas within the reference or transformed image. These areas can be specified using VOIs or reference files. Here, **higher weights mean a greater impact in that area on the registration.**
No weight	– the weighting factor is not used for registration.
Register area delineated by VOIs only	uses VOI to register a selected area.
Weight registration	If this option is checked, the following parameters become available: **Choose ref. weight** opens the dialog where you can select the reference file which contains the information about the weighting factor. **Choose input weight** opens the dialog where you can select the input file which contains the information about the weighting factor.
Output options	
Display transformed image	allows to view the transformed image in a separate window. By default, this option is active.
Interpolation:	– 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.
Advanced options	
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:	

Figure 110. The Optimized Automatic Registration dialog box options (continued)

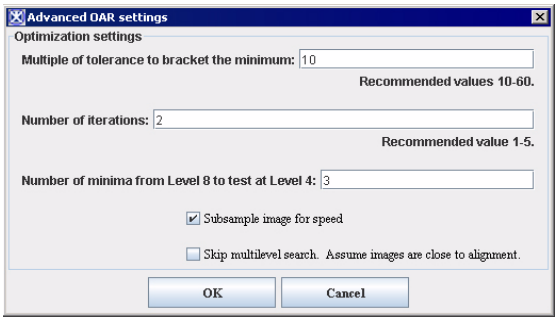
Multiple of tolerance to bracket the minimum	Recommended values are 10–60.	
Number of iterations	Recommended are 1–5 iterations.	
Number of minima from Level 8 to test at Level 4	By default this is the best 20%, but you can enter your own number here.	
Subsample image for speed		subsamples the image.
Skip multilevel search. Assume images are close to alignment.		
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 it.	
Help	Displays online help for this dialog box.	

Figure 110. The Optimized Automatic Registration dialog box options (continued)

Registration: Landmark—Least Squares

The Registration: Landmark—Least Squares algorithm provides a way for you to register 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. That is, this is a rigid transformation based on two homologous landmark datasets.

Background

The Landmark—Least Squares 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 x 2 matrix in 2D or a 3 x 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 uses the transformation matrix to transform the image to be registered into the resulting image (Figure 111).

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 noncolinear points.
- For 3D images, you must identify at least four noncolinear points.

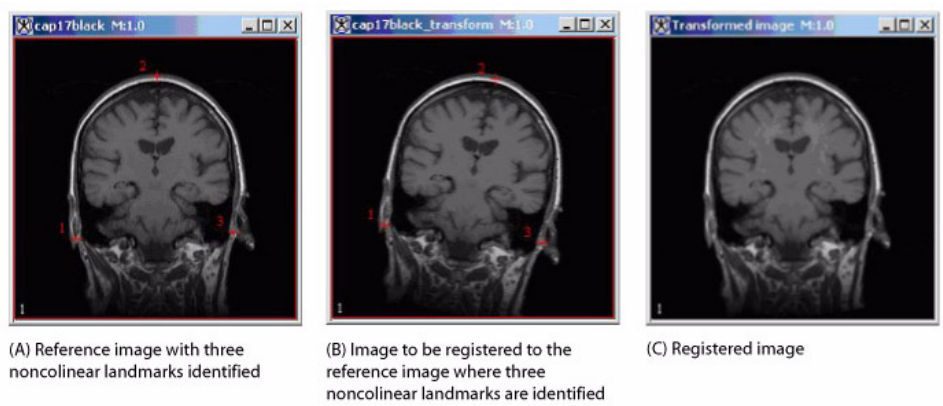


Figure 111. Landmark—Least Squares processing

Both the reference image (A) and the image to be registered (B) have at least three noncolinear 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 image to be registered and a reference image. When you apply this algorithm, MIPAV registers the image to be registered 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 image to be registered. Note that the dialog box shows the last image clicked on as the image to be registered.

Tip: Make sure that you delineate the points on the image to be registered in the same order as on the reference image. That is, point 1 on the reference image should correspond to point 1 on the image to be registered, point 2 on the reference image should correspond to point 2 on the image to be registered, etc.

5. Select Algorithms > Registration > Landmark - Least Squares. The Least Squares Registration dialog box (Figure 112) appears.
6. Select the name of the image to be registered 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.

Register [name of source image] to [name of reference image]	Displays a set of possible reference images. This box registers the image to be registered to the selected reference image.
OK	Applies the algorithm according to the specifications in this dialog box.
Cancel	Disregards any changes that you made and closes the dialog box.
Help	Displays online help for this dialog box.

Figure 112. 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

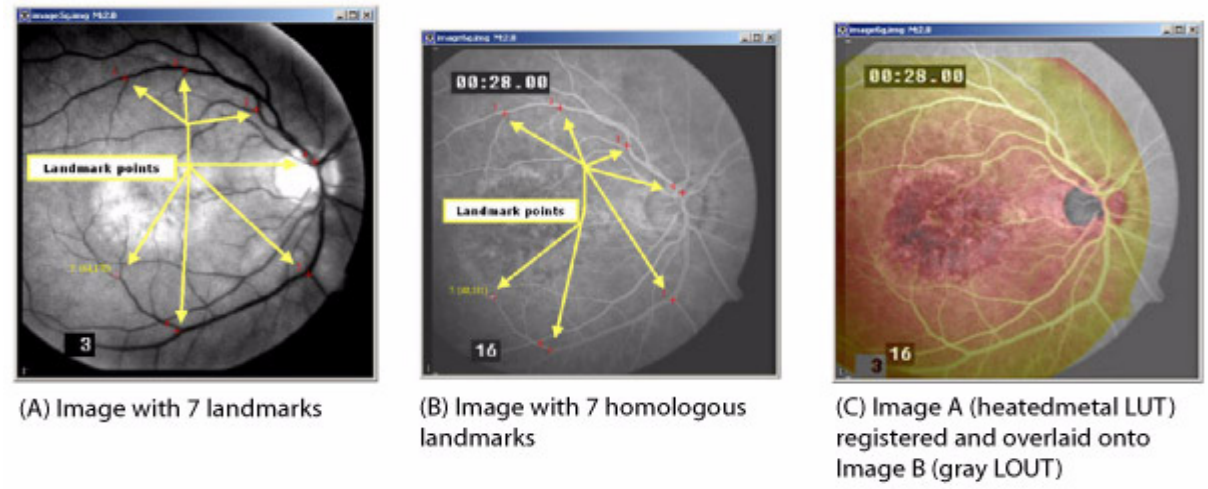
When applied to 2D images, the Landmark—TPSplines algorithm uses a 2D interpolation scheme for arbitrarily spaced tabulated data (x_i,y_i,$f(x_i,y_i)$). These splines are the generalization of the natural cubic splines in 1D. The spline surface represents a thin metal sheet that is constrained not to move at the grid points. When applied to 3D images, the TPSplines 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, 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.



(A) Image with 7 landmarks

(B) Image with 7 homologous landmarks

(C) Image A (heatedmetal LUT) registered and overlaid onto Image B (gray LOUT)

Figure 113. 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.

Register to	Displays a list of images including the base image and any match images.
OK	Applies the mean algorithm to the base image that you chose in this dialog box.
Cancel	Disregards any changes that you made in this dialog box and closes the dialog box.
Help	Displays online help for this dialog box.

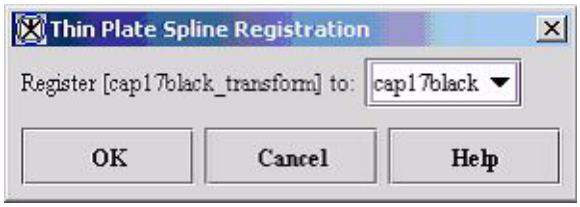


Figure 114. Thin Plate Spline Registration dialog box

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 115), 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.

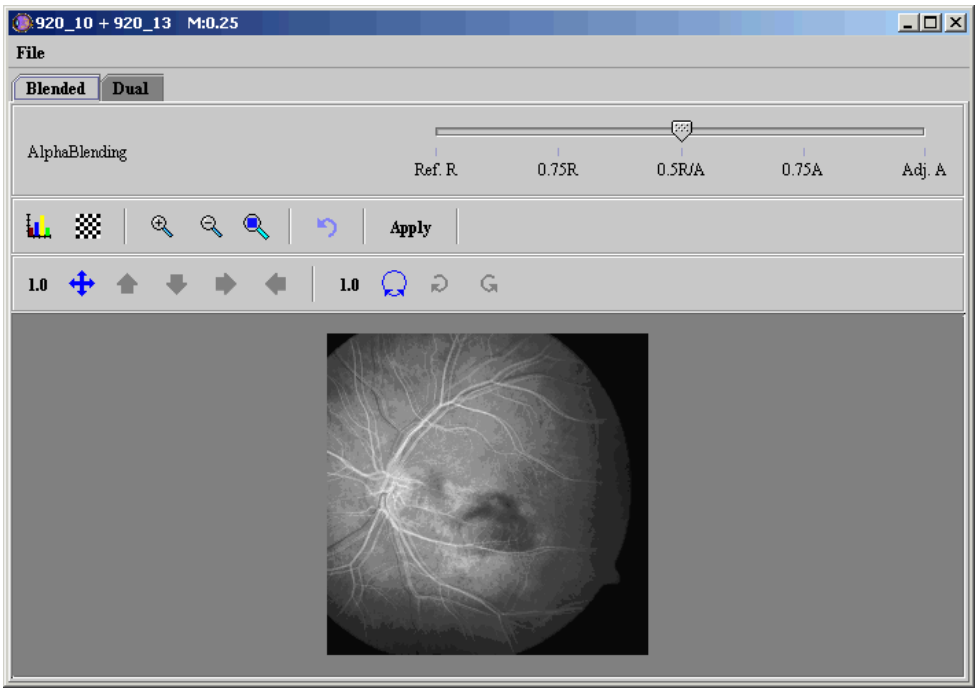


Figure 115. Registration: Manual 2D Series window

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.

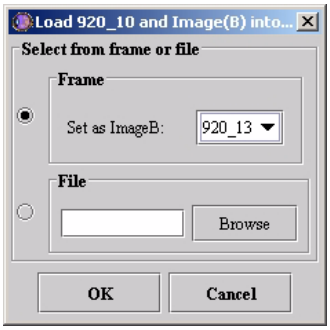


Figure 116. The Load dialog box

4. Select the image that you want to register to the reference image.

Tip: 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.

To select a different image, click Browse to find and open another image file.

5. Click OK.

The Manual 2D Series window opens.

Tip: 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.

6 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.

7 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.

8 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.

9 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 312 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 315)
- Moving the image using the direction arrows (refer to "Moving the adjusted image with the direction arrows" on page 316):
  - , 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 316):
  - , 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 117). In this case, you can use to move the adjusted image again.

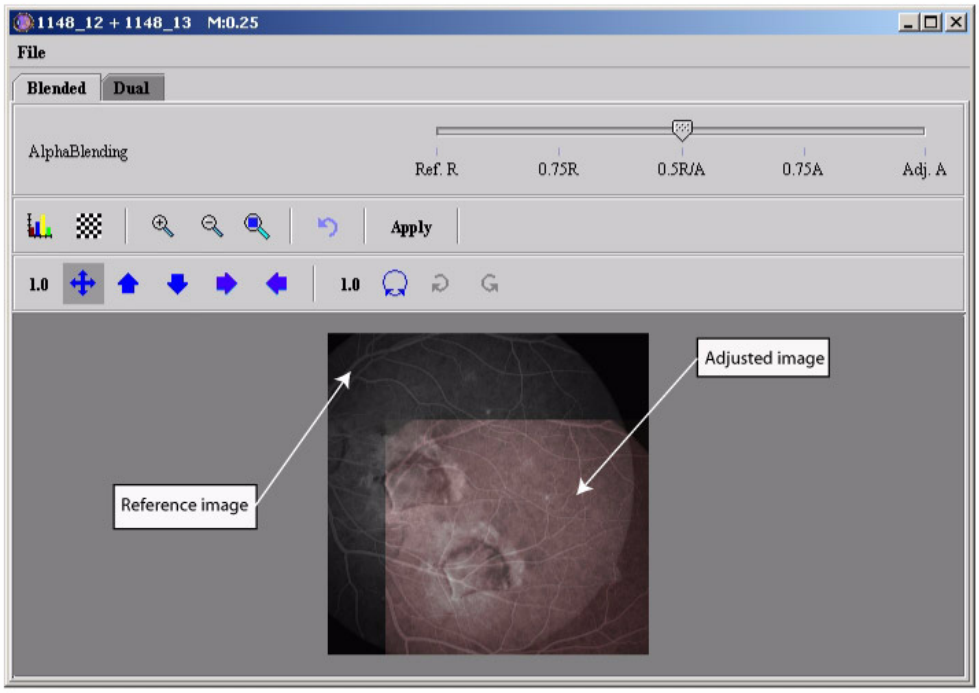


Figure 117. The Blended page of the Manual 2D Series window showing the adjusted image moved down and to the right

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 1.0, 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 118). 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 .

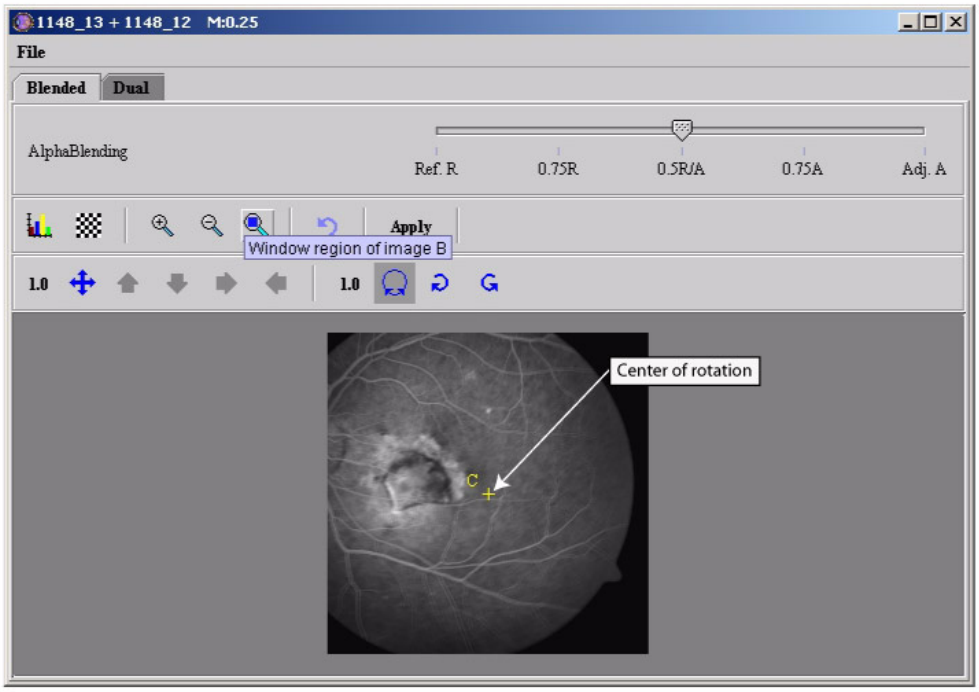
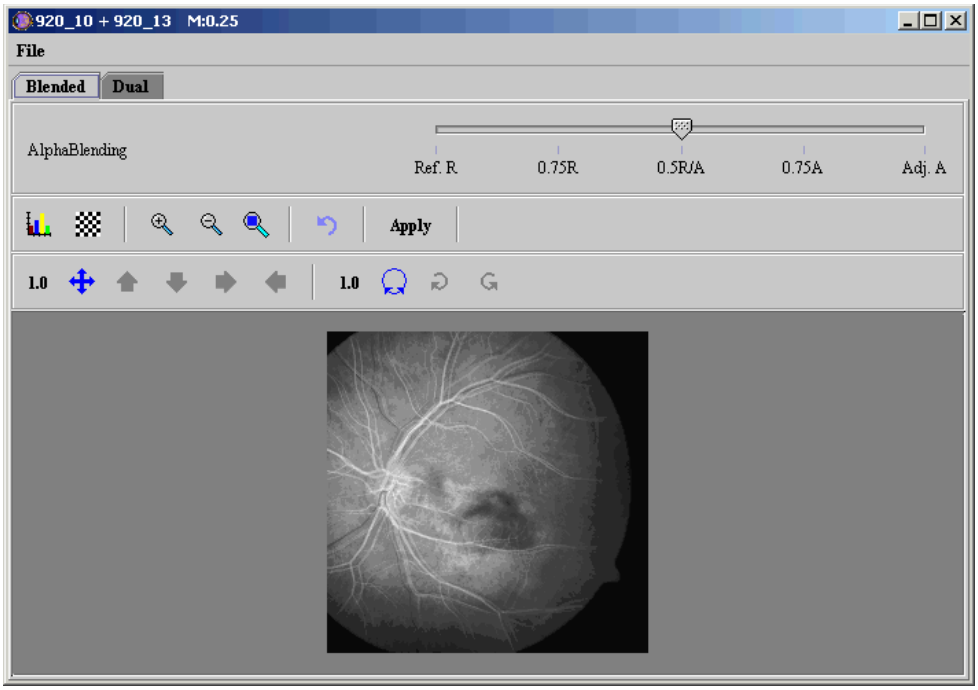


Figure 118. Manual 2D Series window showing center of rotation

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.



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.	
Alphablending slider	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.	
Lookup table	Displays the Lookup Table window, which includes two tabs: Image A, the reference image, and Image B, the image to be registered.	
Checkerboard A & B	Displays a checkerboard pattern on the images. You can use this pattern for assistance in aligning the images.	

Figure 119. Manual 2D Series window showing the Blended page

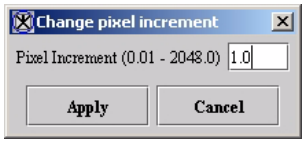
Magnify image 2.0x	Doubles the size of images, magnifying them to twice their current size. If an image is too large for the current window size, scroll bars appear, and you may need to manually adjust the size of the window. To restore an image to its original size, use , the Magnify image 0.5x icon, as many times as you used this icon.
Magnify image 0.5x	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.
Window region of image B	Hides the reference image below a windowed mouse cursor; slide alphablending slider to the reference image
Reset image to 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.
Apply **Apply**	Commits the changes that you made to the images, but it leaves the window open for you to make further changes if desired.
1.0 **Set pixel increment**	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.  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, 550.0 would appear on the toolbar. You can use any pixel increment from 0.01 to 2048.00.
Translate image	Moves Image B, the image to be registered, in any direction you drag the image by the number of pixels indicated in 1.0, the Set pixel increment icon.
Move image up	Moves Image B, the image to be registered, up by the number of pixels indicated in 1.0, the Set pixel increment icon.
Move image down	Moves Image B, the image to be registered, down by the number of pixels indicated in 1.0, the Set pixel increment icon.

Figure 119. Manual 2D Series window showing the Blended page (continued)

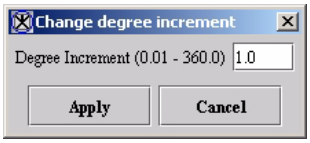
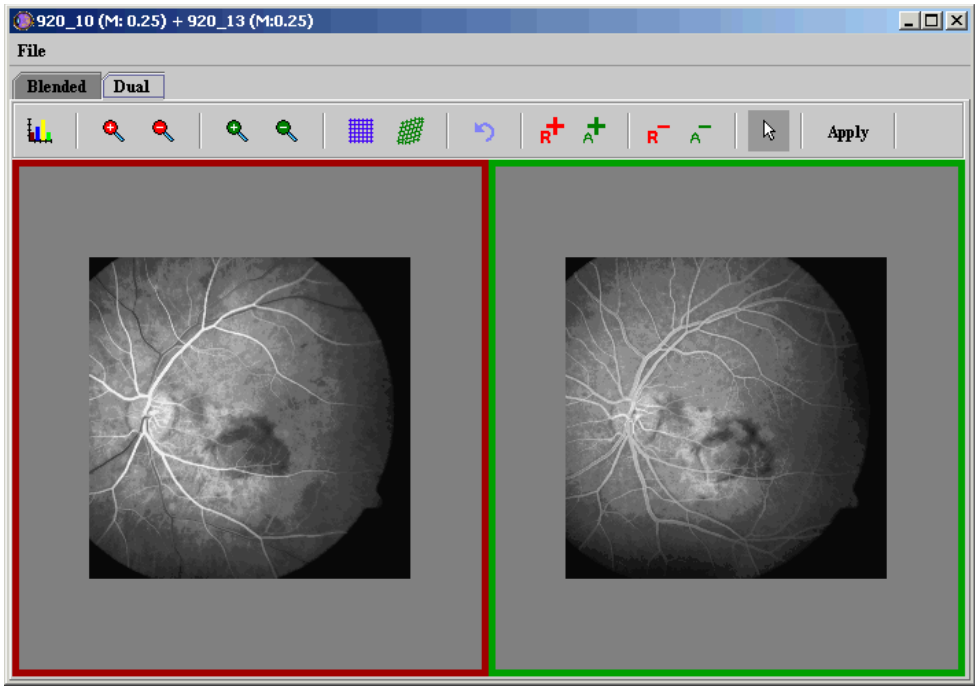
Move image right	Moves Image B, the image to be registered, right by the number of pixels indicated in 1.0, the Set pixel increment icon.
Move image left	Moves Image B, the image to be registered, left by the number of pixels indicated in 1.0, the Set pixel increment icon.
Set degree increment	Determines 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.  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, 180.0 would appear on the toolbar. You can set the degree increment from 0.01 to 360.0.
Rotate image	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.
Rotate clockwise	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.
Rotate counterclockwise	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.
Image area	Superimposes the image to be registered on the reference image and allows for the image to be registered to be moved, magnified, or rotated.

Figure 119. 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	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. To restore reference images to their original size, use , the Magnify reference image 0.5x icon, as many times as you used this icon.	

Figure 120. Manual 2D Series window showing the Dual page

Magnify reference image 0.5x	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.
Magnify adjusted image 2.0x	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. To restore adjusted images to their original size, use , the Magnify reference image 0.5x icon, as many times as you used this icon.
Magnify adjusted image 0.5x	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.
Apply least squares alignment	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.
Apply thin plate spline alignment	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.
Reset image to original state	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.
Reference slice markers	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.
Adjusted slice markers	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.
Delete selected reference slice markers	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.

Figure 120. Manual 2D Series window showing the Dual page (continued)

Delete selected adjusted slice markers	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.
Return to default mode	Returns the cursor to its default mode, which allows you to change from adding points to deleting points from the image.
Apply **Apply**	Commits the changes that you made to the images, but it leaves the window open for you to make further changes if desired.

Figure 120. 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 121) 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 119 and Figure 120).

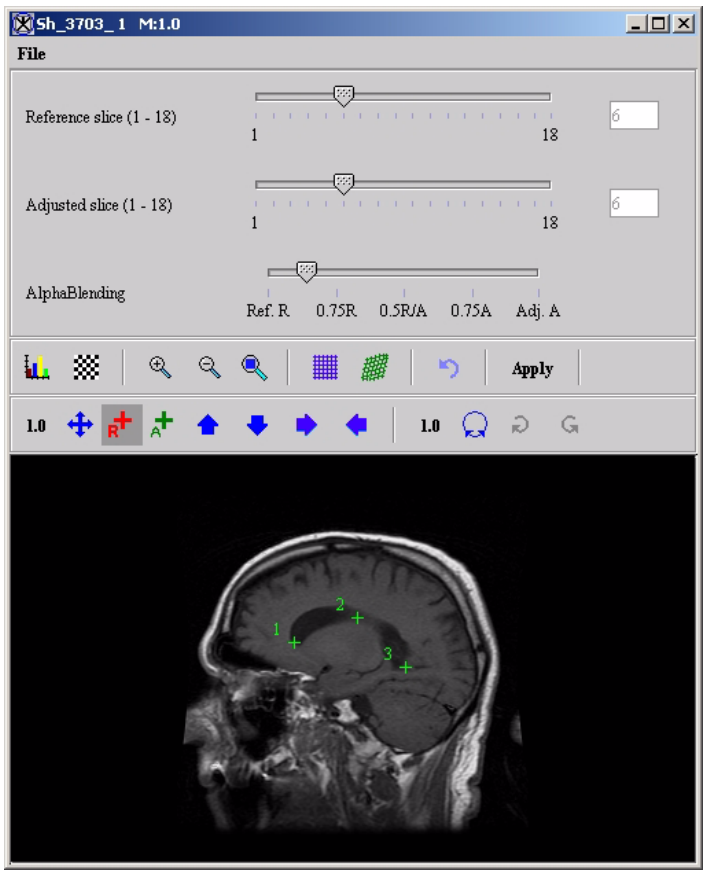


Figure 121. Manual 2D Series window for 2.5D images

For 2.5D images, this window does not include the Blended and Dual tabbed pages. Rather is 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 310.

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 122 on page 332) 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:

$$N = \frac{var(RI)}{\sqrt{varR \cdot varI}}$$

where N is the normalized cross correlation and var is the variance.

$$varRI = \frac{sum(RI)}{(count-1)} - \frac{(sumR \cdot sumI)}{((count-1) \cdot count)}$$

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

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

where

R = the reference image value - reference image minimum value

I = the input image value - input image minimum value

$count$ = total number of values

Correlation ratio

For correlation ratio, the algorithm goes over all pixels forming *numY, sumY,* and *sumY2:*

$$iV = (int)(R(index) - minR) \times \text{constant}$$

$$numY[iV] = numY[iV] + 1$$

$$sumY[iV] = sumY[iV] + I - minI$$

$$sumY2[iV] = sumY2[iV] + (I - minI)^2$$

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 *bth* 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 num*Y*[]:

$$corrRatio = (corrRatio)/(numTotY)$$

Next, it calculates the total variance of input image and then normalizes by the following:

$$vartTot = (totSumY2 - totSumY \times totSumY / numTotY)/(numTotY - 1)$$

$$corRatio = corRatio/variance$$

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} = (Number of elements for which Value$j$ < $R(q)$ < Valuej+1 and Value$k$ < $I(q)$ < Value$k$+1)/total number of elements

$H(R)$ = -sum over j of ($p_j log(p_j)$) where p_j = (Number of elements for which Value$j$ < $R(q)$ <Value$j$+1)/total number of elements

$H(I)$ = -sum over k of ($p_k log(p_k)$) where p_k = (Number of elements for which Value$k$ < $I(q)$ < Value$k$+1)/total number of elements

The normalized mutual information definitions require the specification of a partition of intensities: {Value0,Value1,...,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.

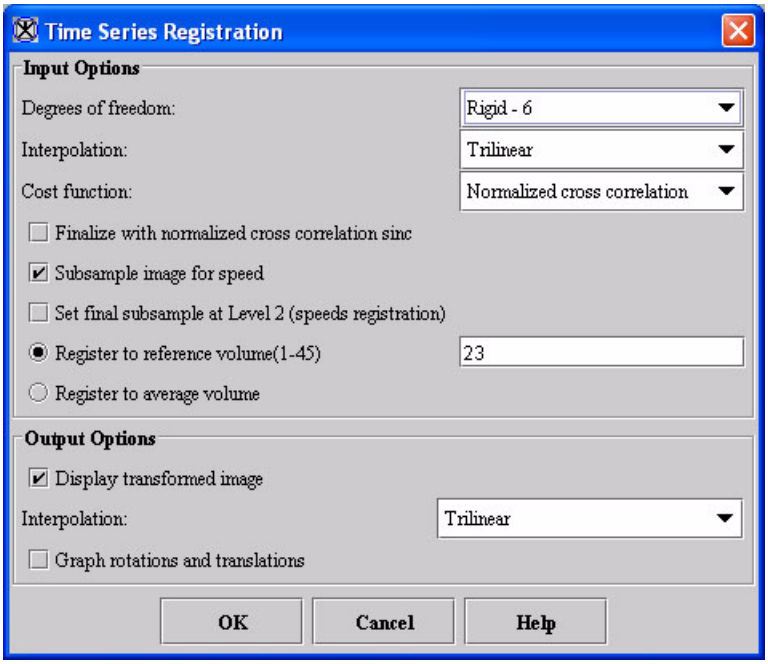
Applying the Time Series Optimized Automatic Registration algorithm

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 122) 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.



Degrees of freedom	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.
Interpolation	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.
Cost function	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.
Finalize with normalized cross correlation sinc	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.
Subsample image for speed	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.
Set final subsample at Level 2 (speeds registration)	Skips the last step in the algorithm that uses full-size images. The default is not selected.

Figure 122. The Time Series Registration dialog box

Register to reference volume (1-number of volumes in times series)	Registers the volume to be registered, or *adjusted volume*, 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.
Register to average volume	Registers all volumes in the time series to the average of all of the volumes.
Display transformed image	Shows the resulting image of the registered volumes.
Interpolation	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.
Graph rotations and translations	Produces two graphs: one graph includes the *x, y,* and *z* translations and the second graph contains *x, y,* and *z* rotations.
Display transformed image	Display transformed image If selected, the transformed image appears in a new window. By default, this option is selected.
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.

Figure 122. 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 the Transform algorithm 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 123 shows an original image and the resulting images after linear, cubic convolution, and cubic B-spline interpolation are applied.

Linear interpolation

The equations for linear interpolation are the following:

$$R(u) = u + 1 \qquad for \qquad 0 \le u < 1$$
$$R(u) = 1 - u \qquad for \qquad 0 \le u < 1$$

Cubic convolution interpolation

The equations that apply to cubic convolution interpolation are:

$$R(u) = \frac{3}{2}|u|^3 - \frac{5}{2}|u|^2 + 1 \qquad for \qquad 0 \le |u| < 1$$
$$R(u) = -\frac{1}{2}|u|^3 + \frac{5}{2}|u|^2 - 4|u| + 1 \qquad for \qquad 1 \le |u| < 2$$

Cubic B-spline interpolation

MIPAV uses the following equations for cubic B-spline interpolation:

$$R(u) = \frac{3}{2} + \frac{1}{2}|u|^3 - |u|^2 \qquad for \qquad 0 \le |u| < 1$$

$$R(u) = \frac{1}{6}[2 - |u|]^3 \qquad for \qquad 1 \le |u| < 2$$

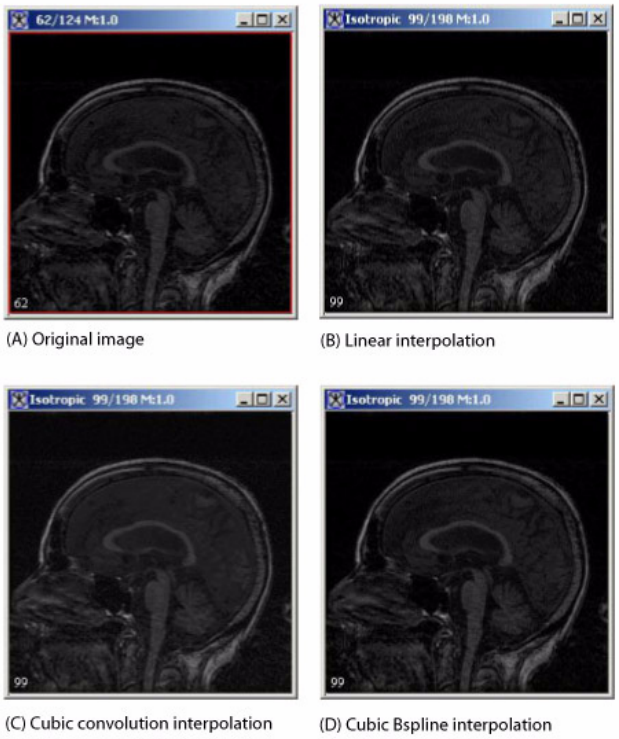


Figure 123. Reslice—Isotropic Voxels algorithm applied to an image

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.

Special notes

None.

References

None.

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 124) 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.

Interpolation	Specifies the type of interpolation to use when applying this algorithm. You can select one of the following: linear, cubic, or cubic B-spline.
OK	Applies the Reslice—Isotropic Voxels algorithm according to the specifications in this dialog box.
Cancel	Disregards any changes you made in this dialog box; closes the dialog box.
Help	Displays online help for this dialog box.

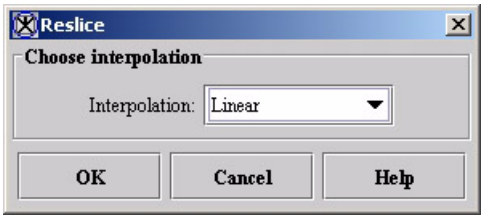


Figure 124. Reslice dialog box

B-Spline Automatic Registration

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

Background

Let the target image have dimensions $p_0 \times p_0 \times p_0$ with image indices satisfying $0 \le j_0 < p_0$, $0 \le j_1 < p_1$, and $0 \le j_2 \le p_2$. Let the source image have dimensions $q_0 \times q_1 \times q_2$ with image indices satisfying $0 \le k_0 < q_0$, $0 \le k_1 < q_1$, and $0 \le k_2 \le q_2$. The B-spline volume function maps indices j_0, j_1, j_2 via Equation 49.

EQUATION 49

$$M_{j0j1j2} = \sum_{i_0=0}^{n_0} \sum_{i_1=0}^{n_1} \sum_{i_2=0}^{n_2} N_{i_0, d_0}\left(\frac{j_0}{p_0 - 1}\right) N_{i_1, d_1}\left(\frac{j_1}{p_1 - 1}\right) N_{i_2, d_2}\left(\frac{j_2}{p_2 - 1}\right) B_{i_0, i_1, i_2}$$

where the $N_{i_e, d_e}(u)$ functions are the B-spline basis functions, and the **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, $M_{j_0 j_1 j_2} = \left(\frac{j_0}{p_0 - 1}, \frac{j_1}{p_1 - 1}, \frac{j_2}{p_2 + 1}\right)$. The initial placement of the control points is described in Section “Improving performance ” on page 344.

- 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 50

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

This constraint implies that the $0 \le k'_w < q_w$ 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 348. The method by which folding is detected is described in Section "Detect folding " on page 349.

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_{j0j1j2} 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 $0 \le k'_w < q_w$ 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 $k_w = |k'_w|$, the largest integer smaller than k'_w. Define $\Delta w = k'_w - k_w$. The interpolated image value is:

EQUATION 51

$$U_0 = (1-\Delta_0)\cdot S_{k_0,\,k_1,\,k_2} + \Delta_0 \cdot S_{k_0+1,\,k_1,\,k_2}$$

$$U_1 = (1-\Delta_0)\cdot S_{k_0,\,k_1+1,\,k_2} + \Delta_0 \cdot S_{k_0+1,\,k_1+1,\,k_2}$$

$$U_2 = (1-\Delta_0)\cdot S_{k_0,\,k_1,\,k_2+1} + \Delta_0 \cdot S_{k_0+1,\,k_1,\,k_2+1}$$

$$U_3 = (1-\Delta_0)\cdot S_{k_0,\,k_1+1,\,k_2+1} + \Delta_0 \cdot S_{k_0+1,\,k_1+1,\,k_2+1}$$

$$V_0 = (1-\Delta_1)\cdot U_0 + \Delta_1 \cdot U_1$$

$$V_1 = (1-\Delta_1)\cdot U_2 + \Delta_1 \cdot U_3$$

$$S(k'_0, k'_1, k'_2) = (1-\Delta_2)\cdot V_0 + \Delta_2 \cdot V_1$$

If the target image samples are T_{j_0,j_1,j_2}, where $0 \le j_w < p_w$ for all w, then the source image S is trilinearly interpolated at (k'_0, k'_1, k'_2), which are the coordinates of M_{j0j1j2} . 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(j0,j1,j2) and $T_{j0,j1,j2}$ 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 Bi_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 Δx , Δy and Δz be small step sizes equivalent to one sample increment in the target image,

EQUATION 52

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

The gradient of the error function is estimated at each control point $B_{i0,i1,i2}$. The gradient at each control point is denoted by

EQUATION 53

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

The gradient descent step is as follows. The current control point under consideration is $B_{i0,i1,i2}$. The ray originating at this point and having direction of greatest decrease in E is

EQUATION 54

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

where the direction is the unit length vector

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

and where ∇E is computed using Equation 53. 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^1 through t_m, evaluate $e(t_i)$ and find the minimum value. Suppose this occurs at t_k. Replace this in Equation 54 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 349 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 $t_{f\,old}$ 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 $t < t_{f\,old}$.

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 55

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

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 56

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

where

- 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 $N = \Sigma_k n_k$.
- N is the number of samples in the target image T which is the same number of samples in the registered source image S'.
- 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 57

$$E_{NMI} = \frac{H(S', T)}{H(S') + H(T)}$$

where

$$H(S', T) = -\sum_{i,j} p_{i,j} \log p_{i,j}$$

is the standard entropy definition for probability pij 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 $N = \Sigma_k n_k$. 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 xed, 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 49 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 $0 <= i_k <= n_k$. The goal is to compute the coordinate values Bi_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 58

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

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 59

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

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 49 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 $0 <= j_k < p_k$.

A system of p_k equations above for $n_k + 1$ unknowns (i.e., the Bi_k coordinate values) is set up in a matrix form A**X** = **C**, where

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

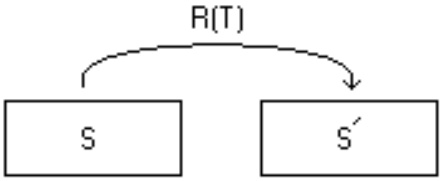
The solution is computed as

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

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

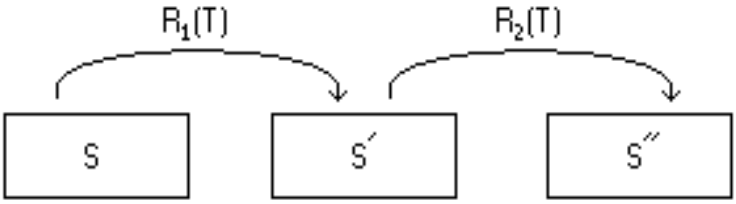
Composition for two-pass registration

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.

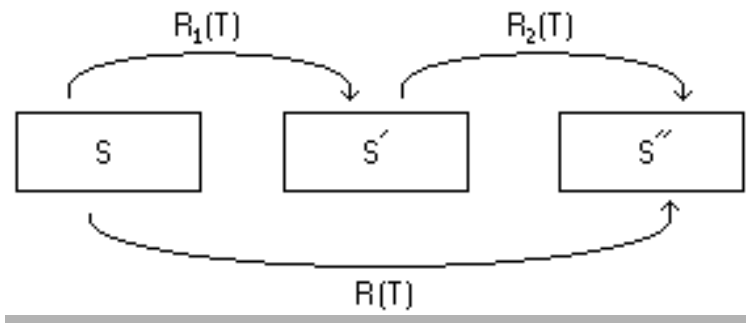
In the case of two-pass registration, the registered source image *S'* is then registered to the same target image *T*, but this time using different mapping parameters, i.e., a different set of B-spline basis and control points for the axes. The output of this second registration is a new registered 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 49. Since each M_{j0j1j2} contains normalized 3D coordinates of the position in the source image mapped for each (j0, j1, j2) in the target image, a map image of 3D coordinates can be created that contains these M_{j0j1j2} 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 348. 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 60

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

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 337. 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 125 below shows an example of B-spline 3D image registration. Figure 126 shows the registration deformation image and its lookup table. The registration was done using the default parameter values, which are explained in Section “User Dialogs in MIPAV ” on page 353.

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 125. 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.

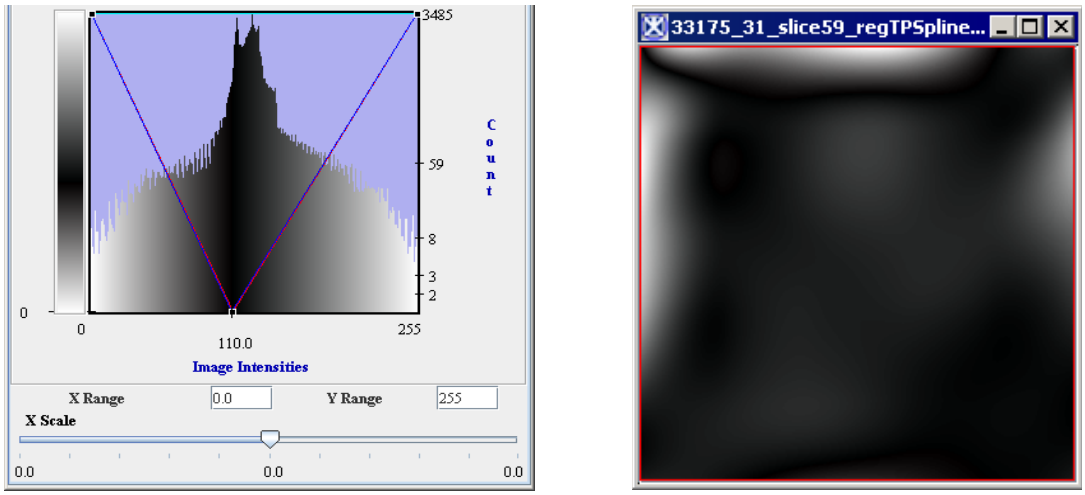


Figure 126. The deformation image and its lookup table. The dark areas on the deformation image are related to the areas on images (A) and (B) which did not perform a deformation during the registration or the deformation was relatively small. The light areas are related to the areas on images (A) and (B) which perform a bigger deformation.

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 127 for details.

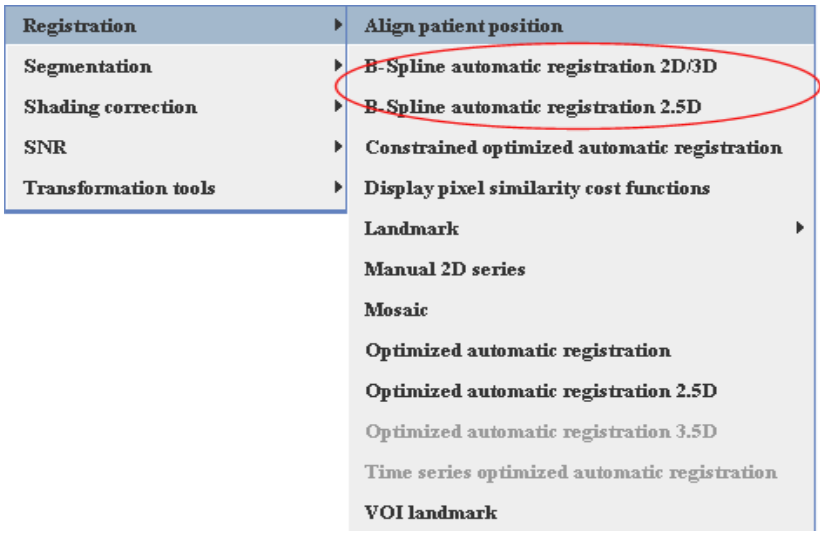


Figure 127. The choice of 2D/3D or 2.5D B-spline registration is made by selecting the appropriate menu item from the Algorithms: Registration submenu

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 128 and 129.

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 130.

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 128 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**.

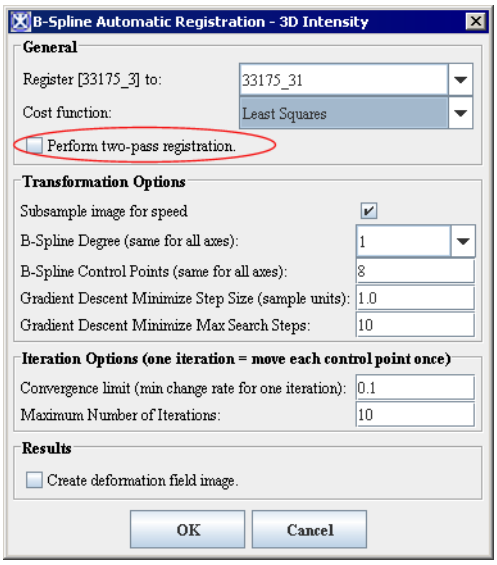
General		
Register	Displays a list of currently active 2D/3D images to registration. The drop down list of options show the names of available images currently loaded to which this image can be registered. This list will only contain the names of images currently loaded in MIPAV which are compatible.	
Cost function	This drop down list contains the following choices for cost measure to use during registration: Least Squares, Correlation Ratio, and Normalized Mutual Information.	
Perform two-pass registra-tion	This dialog is shown with the default options for single-pass registration. If the **Perform two-pass registration** option is enabled, then the dialog will change to make options for **two-pass registration** available.	
Transformation Options **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 363.**		
Sub-sample image for speed	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-Spline Degree (same for all axes)	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.	

Figure 128. B-Spline Single - Pass Automatic Registration 2D/3D dialog box

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.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.
Gradient Descent Minimize Step Size (sample 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**. E.g., a value of 0.5 will move the control point in increments of **half of 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 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.
Iteration options (one iteration=move each control point once)	
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.
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.
Results	
Create deformation field image	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 *deformation* suffix. The deformation image will be automatically displayed along with the registered source image upon completion of registration. Refer to Figure 126 for details.
OK	Applies the algorithm to the input image that you chose in this dialog box.
Cancel	Disregards any changes that you made in this dialog box and closes the dialog box.
Help	Displays online help for this dialog box.

Figure 128. 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:

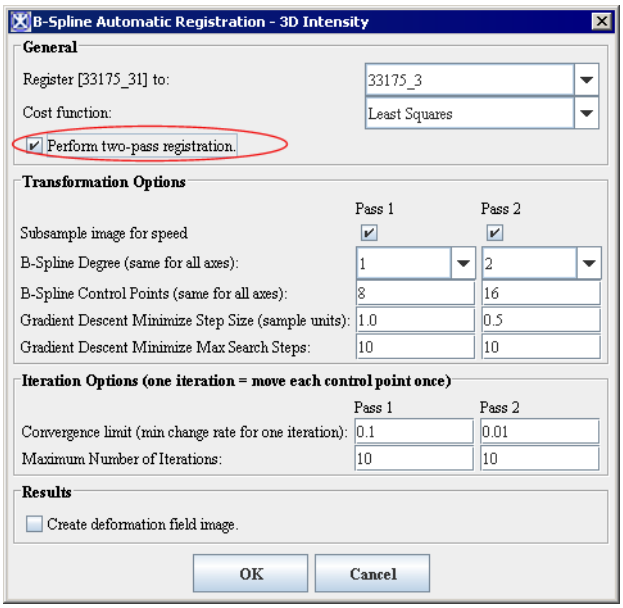
General		
Register	Displays a list of currently active 2D/3D images to registration. The drop down list of options show the names of available images currently loaded to which this image can be registered. This list will only contain the names of images currently loaded in MIPAV which are compatible.	
Cost function	This drop down list contains the following choices for cost measure to use during registration: Least Squares, Correlation Ratio, and Normalized Mutual Information.	
Perform two-pass registra-tion	This dialog is shown with the default options for **two-pass registration**. If the **Perform two-pass registration** option is disabled, then the dialog will change to make options for **single-pass registration** available as shown in Figure 128.	
Transformation Options **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 363.**		
Sub-sample image for speed	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-Spline Degree (same for all axes)	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-Spline Control Points(same for all axes)	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.	

Figure 129. B-Spline Two-Pass Automatic Registration 2D/3D dialog box

Gradient Descent Minimize Step Size (sample 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**. E.g., a value of 0.5 will move the control point in increments of **half of 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 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.
Iteration options (one iteration=move each control point once)	
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.
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.
Results	
Create deforma-tion field image	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 *deformation* suffix. The deformation image will be automatically displayed along with the registered source image upon completion of registration. Refer to Figure 126 for details.
OK	Applies the algorithm to the input image that you chose in this dialog box.
Cancel	Disregards any changes that you made in this dialog box and closes the dialog box.
Help	Displays online help for this dialog box.

Figure 129. 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 129.

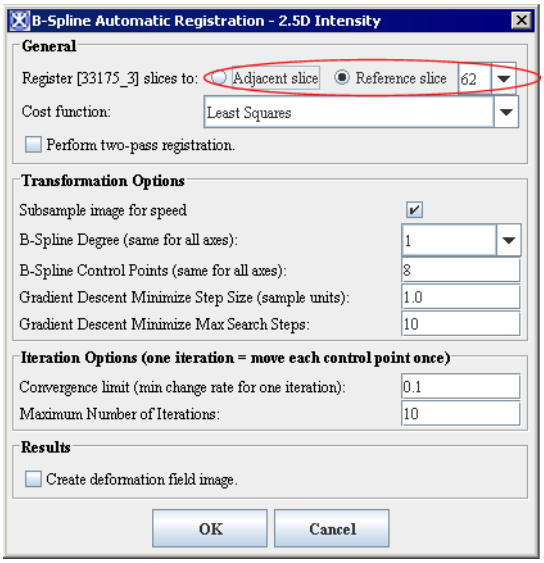
General		
Register	The Register source field is already filled with the name of the currently selected 2.5D image. The reference slice is selected from among the slices in the selected image. Radio buttons are used to select whether the reference slice is the the **adjacent** (previous) slice or a **particular slice number** in the image. If the **Reference Slice** radio button is selected, then a drop down list of slices numbers is enabled where the middle slice is selected by default.	
Cost function	This drop down list contains the following choices for cost measure to use during registration: Least Squares, Correlation Ratio, and Normalized Mutual Information.	
Perform two-pass registration	This dialog is shown with the default options for single-pass registration. If the **Perform two-pass registration** option is enabled, then the dialog will change to make options for **two-pass registration** available. Refer to Figure 131.	
Transformation Options **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 363.**		
Sub-sample image for speed	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.	

Figure 130. B-Spline Single-Pass Automatic Registration 2.5D dialog box

B-Spline Degree (same for all axes)	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-Spline Control Points(same for all axes)	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.
Gradient Descent Minimize Step Size (sample 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**. E.g., a value of 0.5 will move the control point in increments of **half of 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 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.
Iteration options (one iteration=move each control point once)	
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.
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.
Results	
Create deformation field image	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 *deformation* suffix. The deformation image will be automatically displayed along with the registered source image upon completion of registration. Refer to Figure 126 for details.
OK	Applies the algorithm to the input image that you chose in this dialog box.
Cancel	Disregards any changes that you made in this dialog box and closes the dialog box.
Help	Displays online help for this dialog box.

Figure 130. B-Spline Single-Pass Automatic Registration 2.5D dialog box (continued)

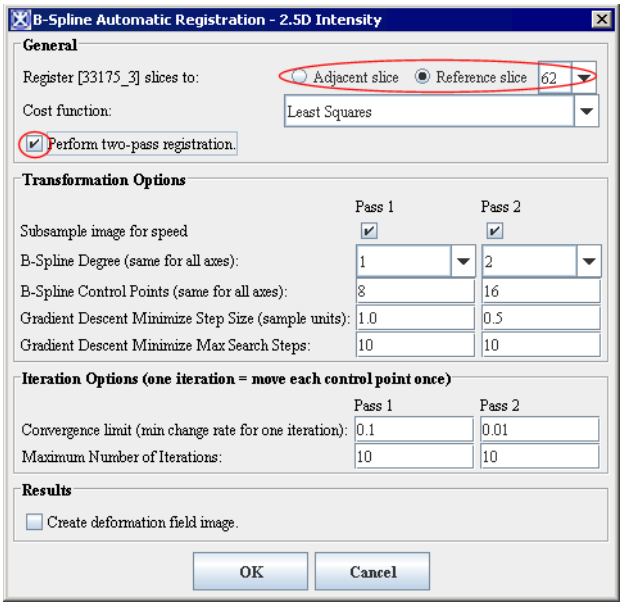
General		
Register	The Register source field is already filled with the name of the currently selected 2.5D image. The reference slice is selected from among the slices in the selected image. Radio buttons are used to select whether the reference slice is the the **adjacent** (previous) slice or a **particular slice number** in the image. If the **Reference Slice** radio button is selected, then a drop down list of slices numbers is enabled where the middle slice is selected by default.	
Cost function	This drop down list contains the following choices for cost measure to use during registration: Least Squares, Correlation Ratio, and Normalized Mutual Information.	
Perform two-pass registration		This dialog is shown with the default options for two-pass registration.
Transformation Options **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 363.**		
Sub-sample image for speed	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-Spline Degree (same for all axes)	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.	

Figure 131. B-Spline Automatic Two-Pass Registration 2.5D dialog box

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.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.
Gradient Descent Minimize Step Size (sample 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**. E.g., a value of 0.5 will move the control point in increments of **half of 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 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.
Iteration options (one iteration=move each control point once)	
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.
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.
Results	
Create deformation field image	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 *deformation* suffix. The deformation image will be automatically displayed along with the registered source image upon completion of registration. Refer to Figure 126 for details.
OK	Applies the algorithm to the input image that you chose in this dialog box.
Cancel	Disregards any changes that you made in this dialog box and closes the dialog box.
Help	Displays online help for this dialog box.

Figure 131. 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 B`SplineRegistration2Df` 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 `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-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

```
(previousError-currentError) / 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.

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

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 ^u = log(u) and ^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:

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 N3 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^v* and the histogram distribution of the log, and use *^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 f*ull-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 132. 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 133.

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

~F is the fourier transform of F

Z is a constant term to limit the magnitude of ~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 132.

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

$$\hat{f}_e(\hat{v}) = \hat{v} - E\lceil \hat{u} | \hat{v} \rceil$$

8 Smooths the bias field estimate—Using the B-spline technique with the parameter *d* field distance input provided by users.

EQUATION 8

$$f_S(\hat{v}) = S\{fe(\hat{v})\}$$

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

$$e = \frac{s\{r_n\}}{m\{r_n\}}, \; n = 1 \ldots N$$

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

$$u = \frac{v}{f}$$

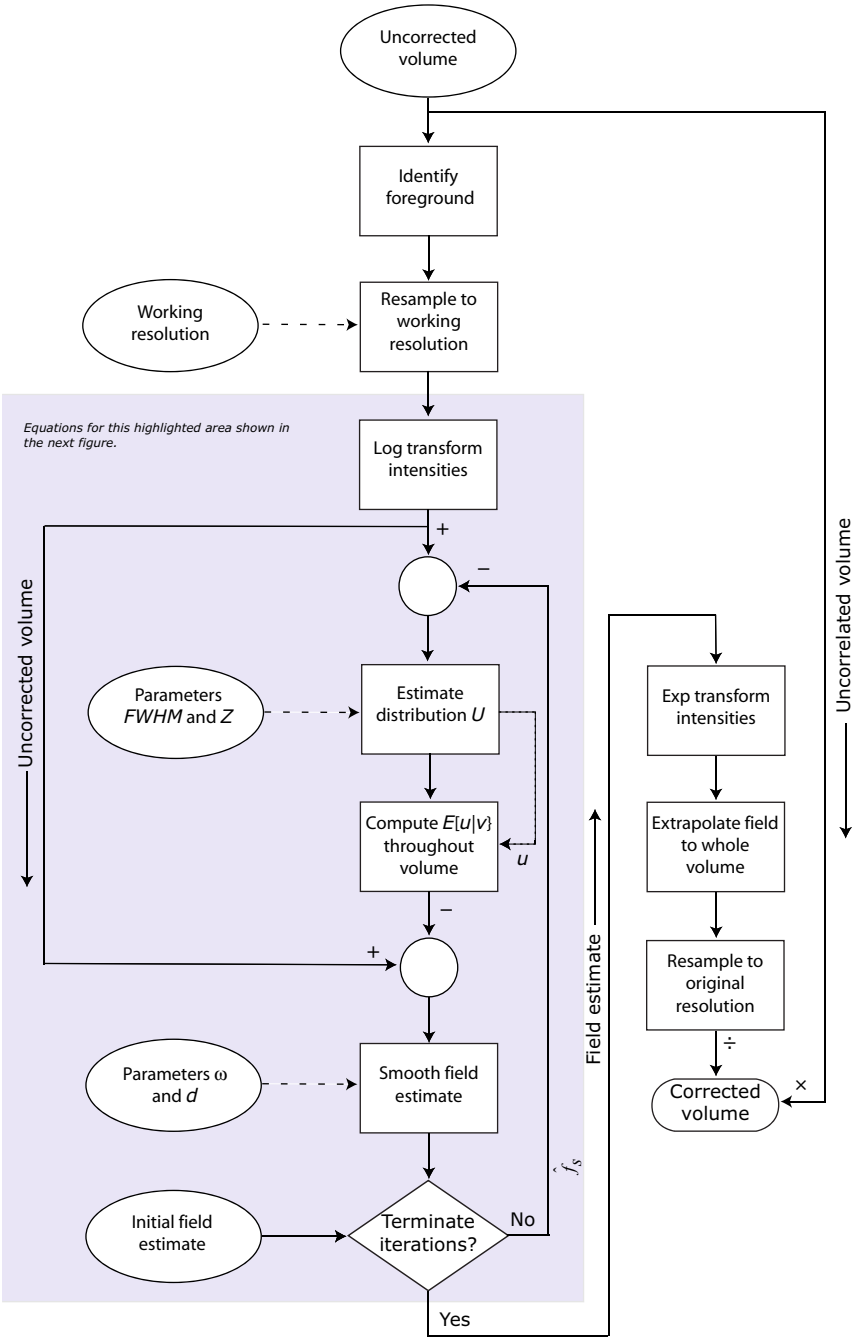


Figure 132. Flow chart describing the N_3 method

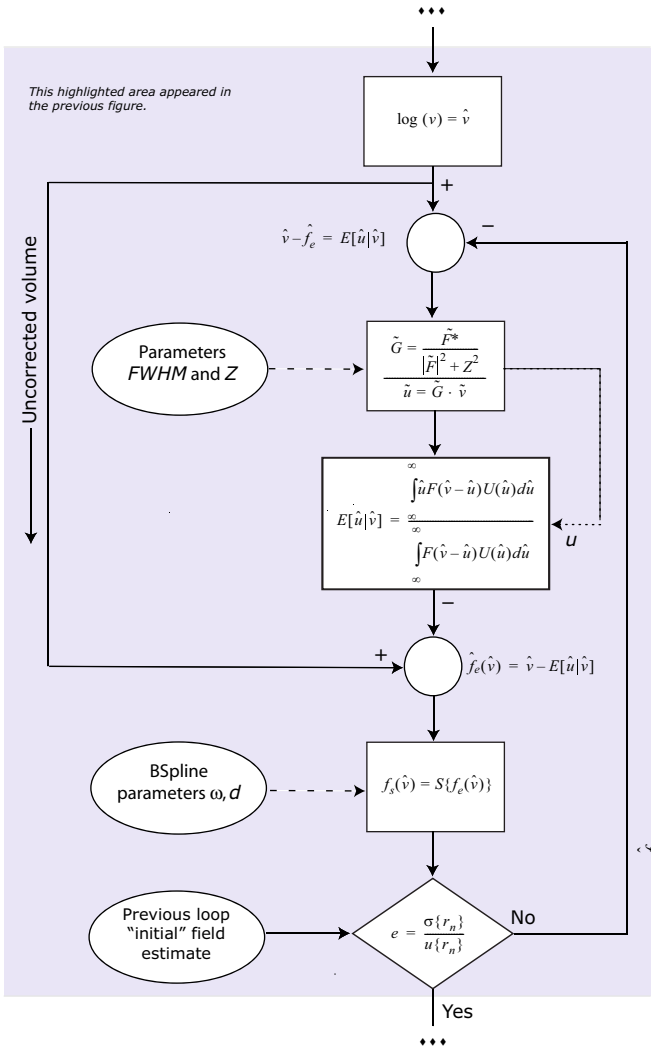
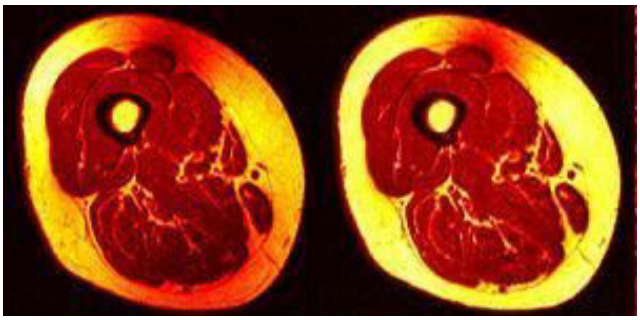


Figure 133. The N_3 method showing the equations



(A) Original image (B) After N3 correction

Figure 134. The effect of N_3 on an image: (A) original image and (B) image after N_3 correction

Figure 134 shows an original image A and the image B produced from using the best N3 paratmeters:

- Treshold = 100
- Maximum iterations = 100
- End tolerance = 0.0001
- Field distance = 33.33
- Subsampling factor = 2
- Kernel FWHM = 0.8
- Wiener = 0.01

Figure 135 shows an original image and the effect of different parameters used in the Inhomogeneity N3 Correction algorithm on the image.

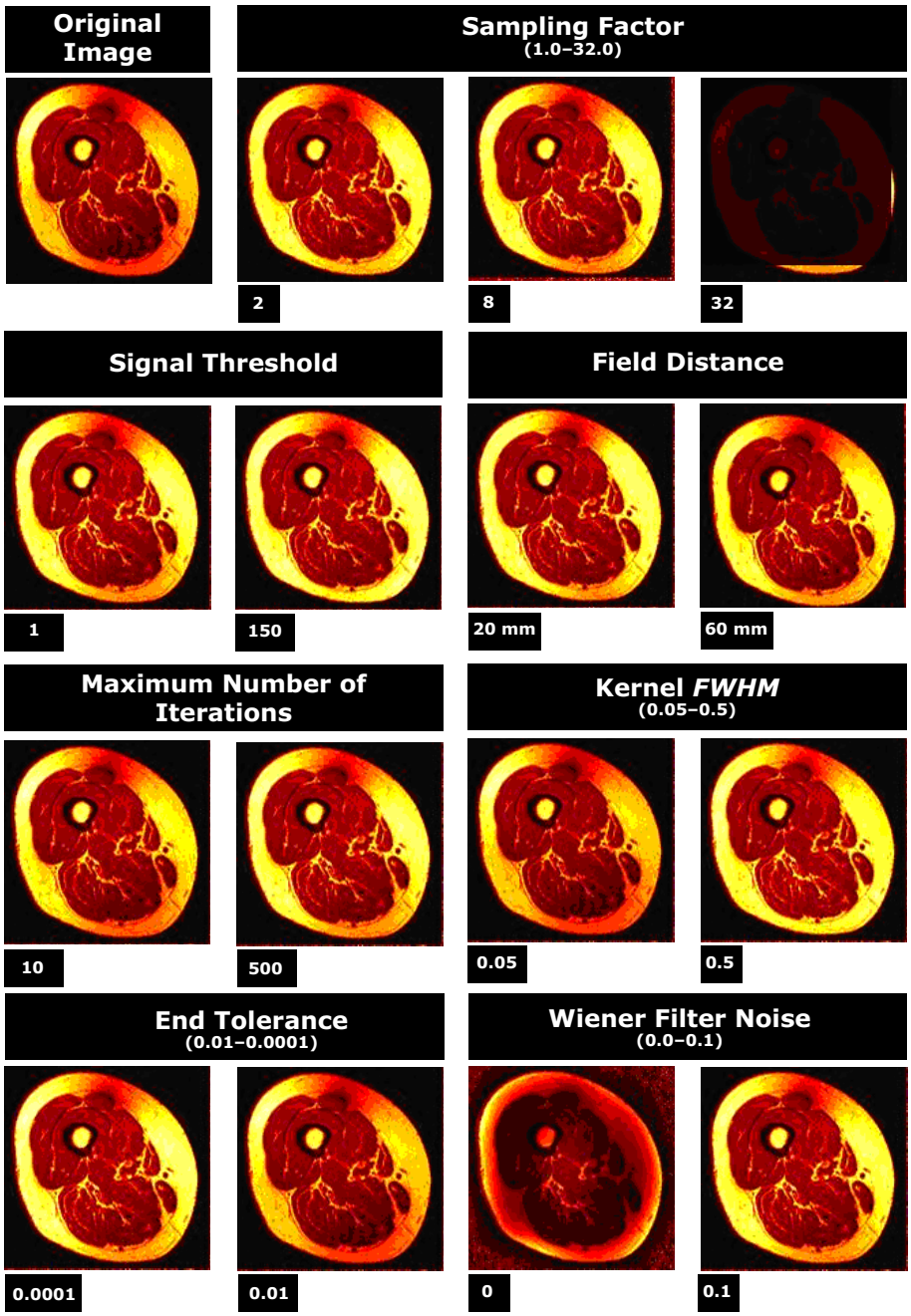


Figure 135. The effects of different settings for the Inhomogeneity N_3 algorithm on an image

Image Types

You should apply this algorithm only to 2D and 3D MRI images.

Special Notes

None.

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.

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 376 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

it all depends on data

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

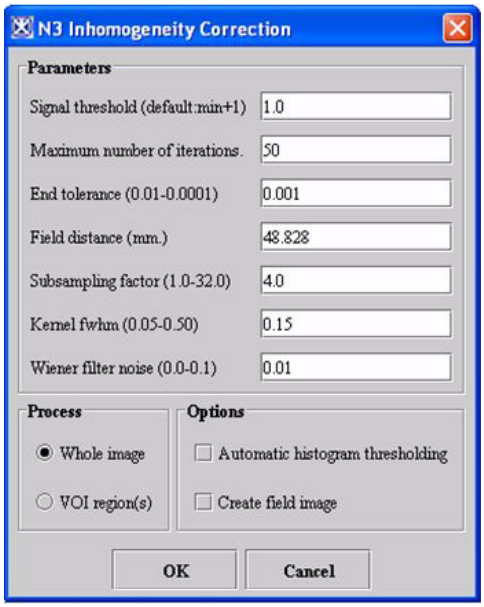
be about 10. That value should really only be used with several intense and isolated artifacts. For regular images a third of the default is sufficient. For quick results or for artifacts largely spread and not too strong, choose a value that is a little less than the default value, but more than half 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.

Signal threshold	Treats those voxels of intensity that are lower than this threshold value as background. The default value is 1.0.
Max number iterations	Specifies the maximum number of iterations allowed before the process is terminated. The default value is 50.
End tolerance	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.
Field distance	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.



Subsampling Factor	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.
Kernel FWHM	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.
Wiener Filter Noise	Indicates the noise used in Wiener filter.
Whole image	Applies the algorithm to the whole image.
VOI region(s)	Applies the algorithm to the volumes (regions) delineated by the VOIs.
Automatic histogram thresholding	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.
Create field image	Indicates whether to create a field image.
OK	Applies the algorithm according to the specifications in this dialog box.
Cancel	Disregards any changes you made in this dialog box and closes the dialog box.
Help	Displays online help for this dialog box.

Figure 136. N3 Inhomogeneity Correction dialog box

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 137 and 138. 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 137 below.

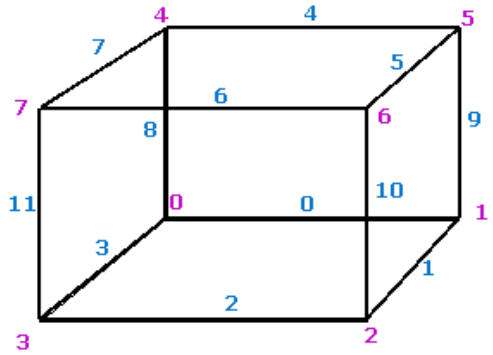


Figure 137. The indexing convention for vertices and edges. Here, edge indexes are shown in blue and vertex indexes are shown in magenta

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 138. 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.

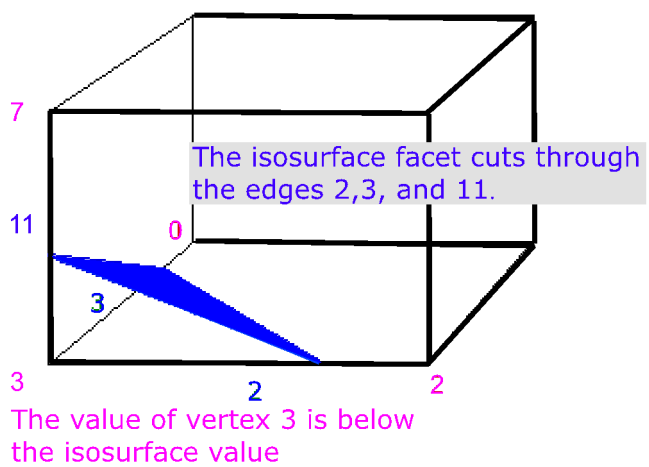


Figure 138. Vertex 3 is inside the volume

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 $\mathbf{2^8 = 256}$ 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 139.

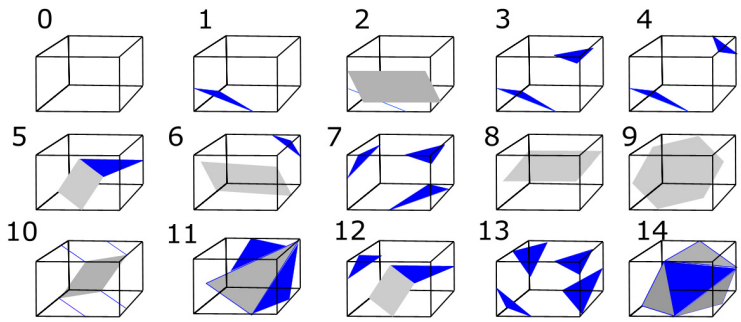


Figure 139. How 256 cases can be reduced to 15 cases

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 3.

Table 3. 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 137, 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 138.

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:

Lorensen, 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 image 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.

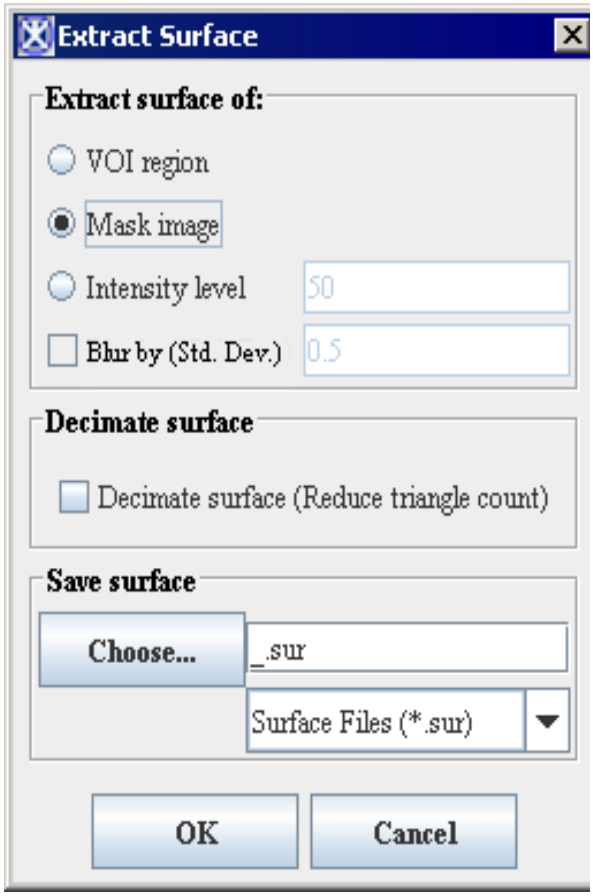
Applying the Extract Surface Algorithm

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 140.



VOI Region	Applies the algorithm to the volume delineated by the VOI.
Mask Image	Applies the algorithm to the volume within a chosen mask. You must apply the mask, first.
Intensity Level	Applies the algorithm to the region with the chosen intensity level. By default, the intensity level value is set to 50.
Blur by (Std.Dev.)	Blurs the image to improve the extracted surface. The value specifies the standard deviation of the Gaussian filter used to regularize the image.
Decimate Surface	If selected, reduces the count of triangle facets, which results the more compact algorithm.
Choose	Use this button to select a catalogue where you would like to store your results and choose a file format (*.txt, *.sur, or *.wrl).
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 it.
Help	Displays online help for this dialog box.

Figure 140. 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.

Volume Tri-planar View

7. The Resample dialog box appears.

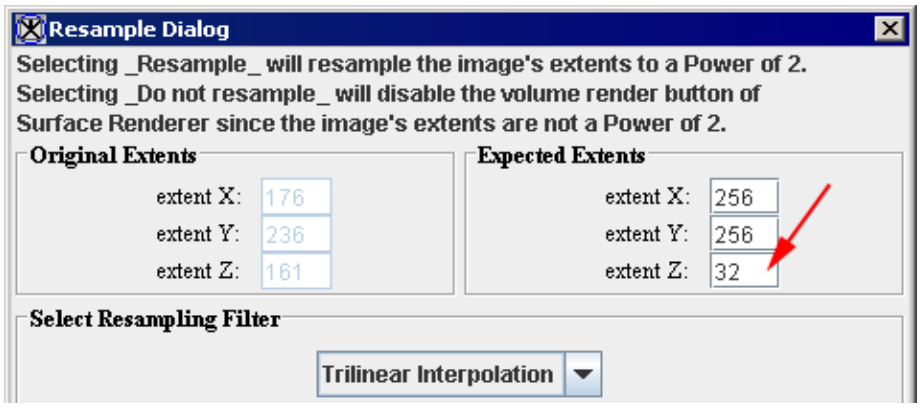


Figure 141. The Resample dialog

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 141. The bigger number you choose, the higher resolution you receive.[1]

1. Please, note that the Resampling in 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.

Add Surface to Viewer

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 142.

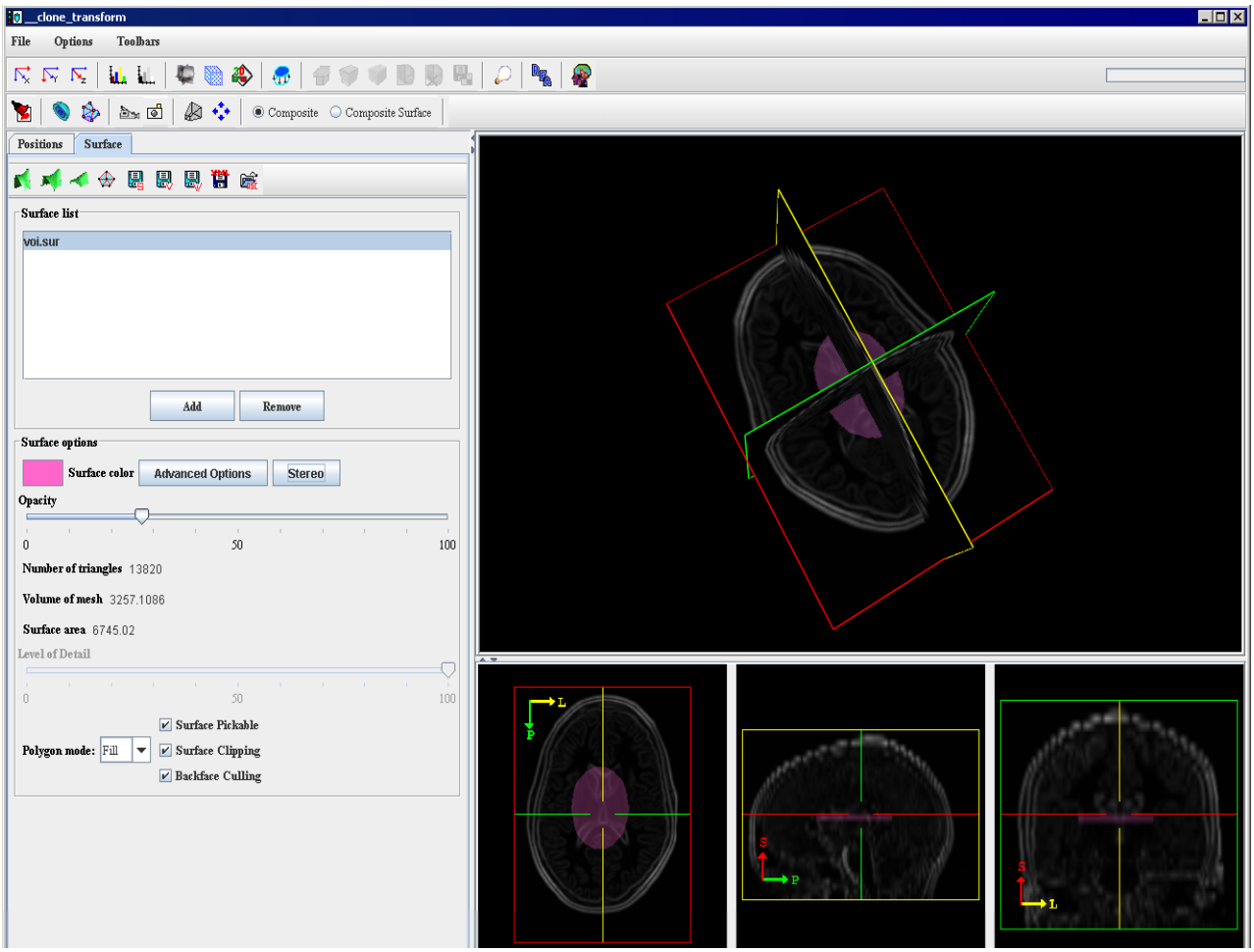


Figure 142. The view of the surface extracted using the VOI Region option

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 385. Refer to Figure 143.

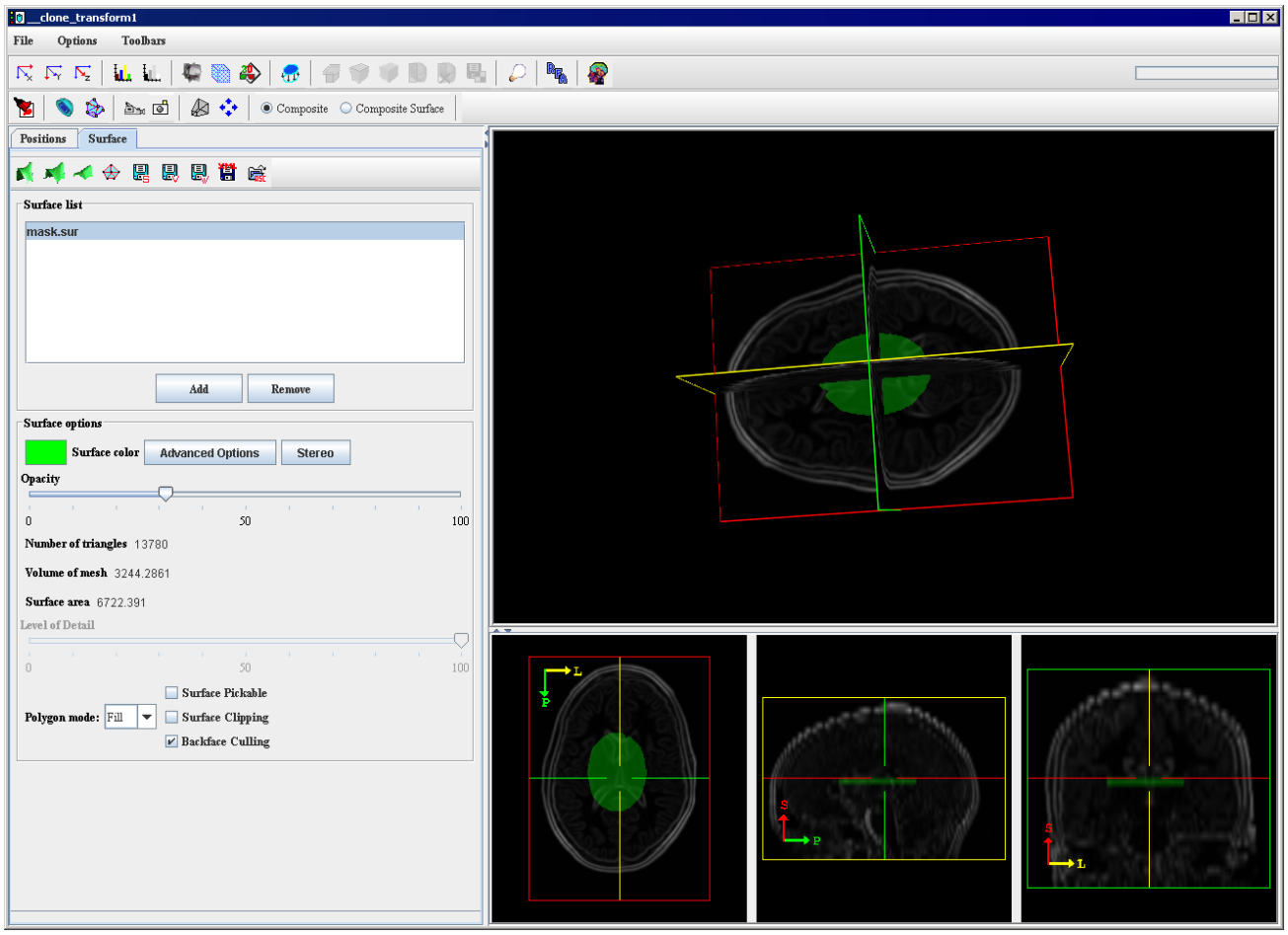


Figure 143. The view of the surface extracted using the Mask Image option

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.

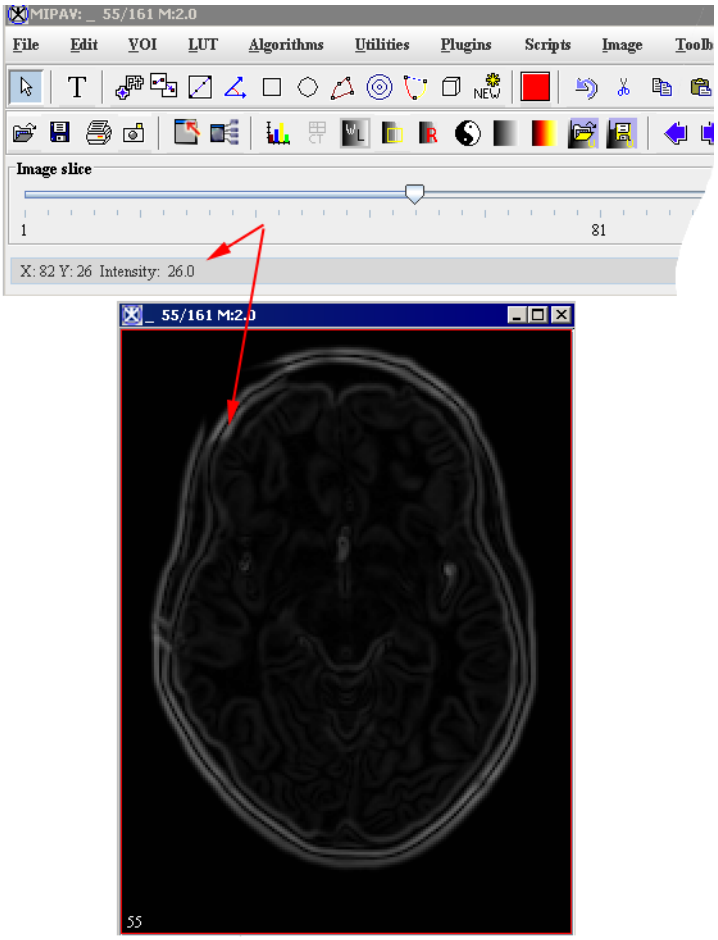


Figure 144. The intensity level is shown on the main MIPAV toolbar

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 144 and 145.

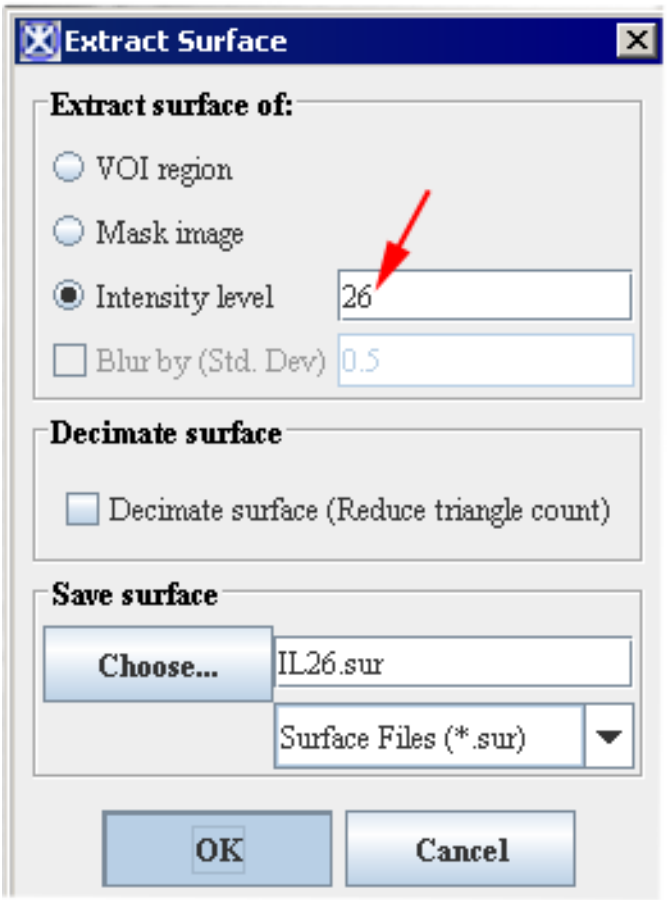


Figure 145. Enter the intensity level value here

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 385.

Figure 146 shows the surface extracted using the Intensity Level option.

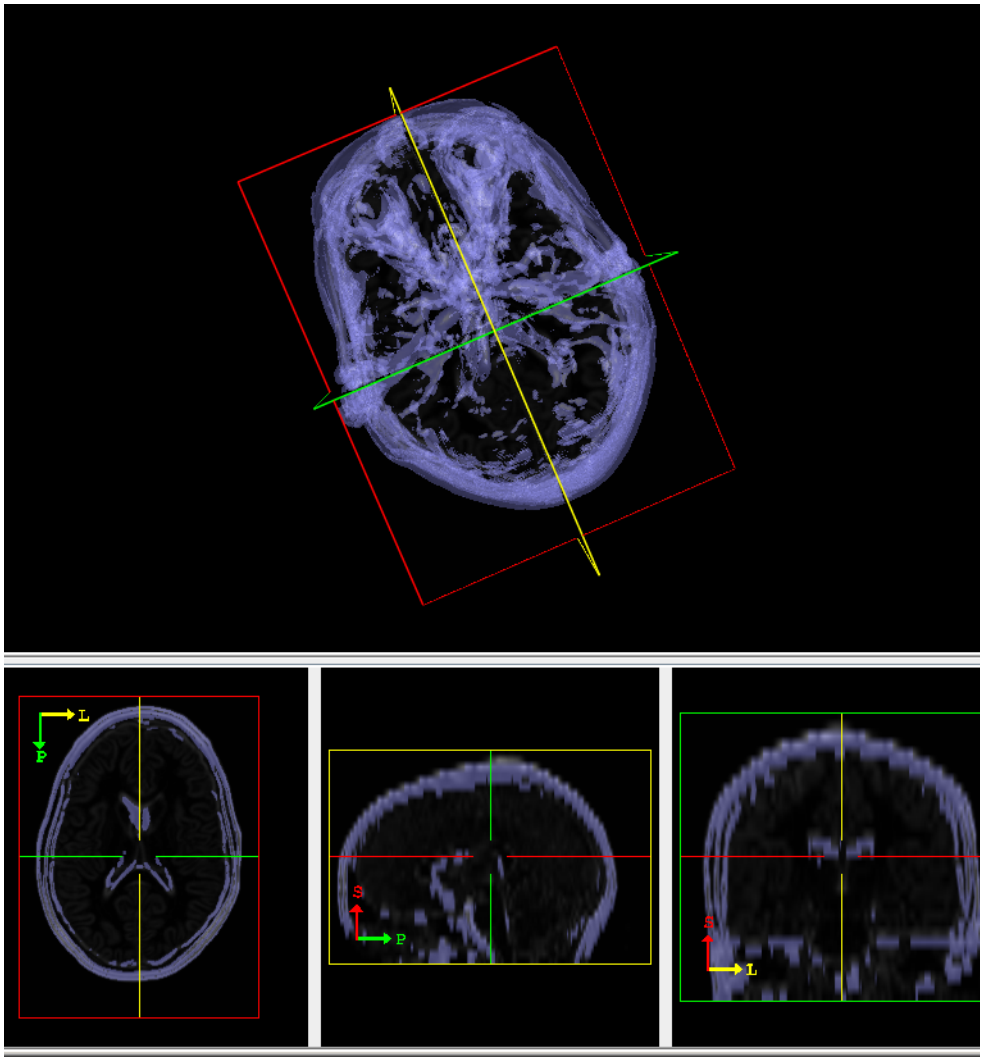


Figure 146. The view of the surface extracted using the Intensity Level option

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 <= 16 will result in 16.

User inputs of >16 and <= 40 will result in 32

User inputs of >40 and <= 80 will result in 64

User inputs of >80 and <= 160 will result in 128

User inputs of >160 and <= 448 will result in 256

Image types

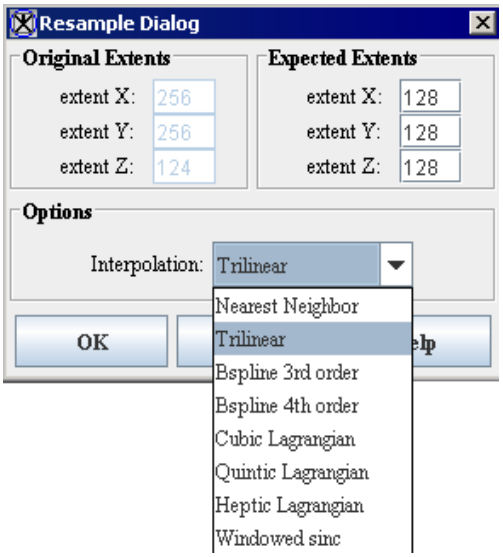
You can apply this algorithm to color and black-and-white images as well as 2D and 3D image types.

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 147.



Original Extents extent X	X dimension of original image.
Original Extents extent Y	Y dimension of original image.
Original Extents extent Z	Z dimension of original image (if image is 3D).
Expected Extents extent X	Expected X dimension.
Expected Extents extent Y	Expected Y dimension.
Expected Extents extent Z	Expected Z dimension (if image is 3D).
Interpolation	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.
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 it.
Help	Displays online help for this dialog box.

Figure 147. The Resample dialog box options

Volume Tri-planar View

Add Surface to Viewer

Registration: 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 415 and "Applying the Manual 2D Series algorithm to 3D images" on page 417.

Background

Through the Registration: Manual 2D Series dialog (Figure 148), 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 401
- "Moving the adjusted image" on page 404
- "Rotating the adjusted image" on page 405
- "Delineating the landmark points" on page 410
- "Applying the image registration algorithms" on page 410

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 148.
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 148.

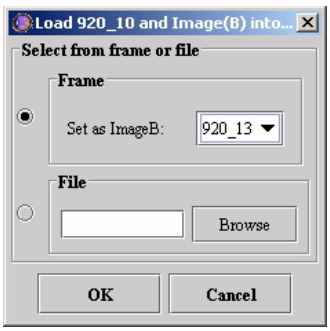


Figure 148. The Load dialog box

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 149, the Registration: Manual 2D Series window has the main menu (see below) and two tabs: Blended and Dual. Refer to "Blended tab" on page 400 and "Dual tab" on page 406.

Options available via the dialog menu

File	**Close Registration**	Closes this dialog without saving the images.
	Show VOIs in blended window	On the Blended tab, displays the VOIs that you've delineated on the Dual tab.
	Thin plate spline for intensity	Uses the Thin Plate Spline algorithm for calculating intensity values. See also "Registration: Landmark—TPSpline" .
Help		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 149 and Section: "Moving and rotating the adjusted image" on page 403.

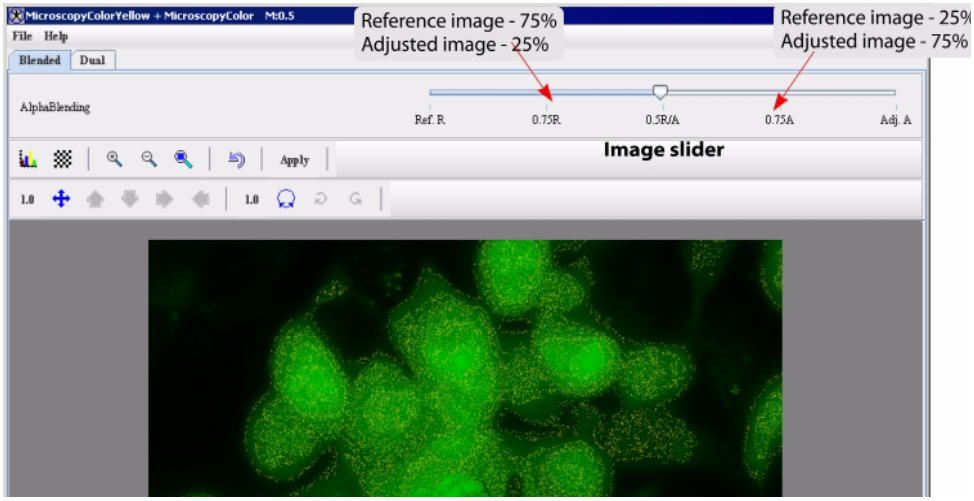


Figure 149. The Blended tab and the image slider

The Blended tab options:

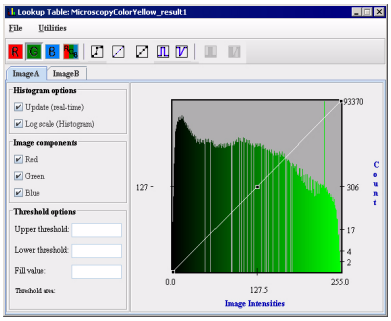
Display Lookup table	Opens the Lookup table dialog box. For the Lookup table (LUT) dialog box options, refer to MIPAV User's Guide, Volume 1, Basics, Figure 117 "Look-up Table window". See also "Using LUT to arrange the images" on page 413.	

Figure 150. The Blended tab options

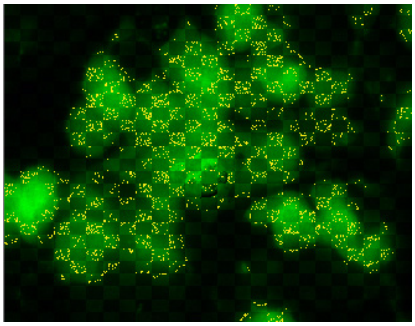
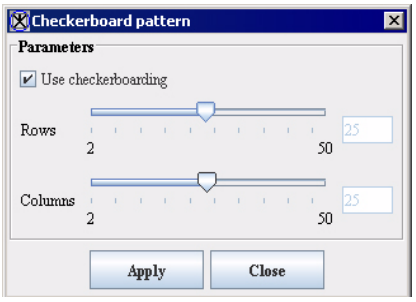
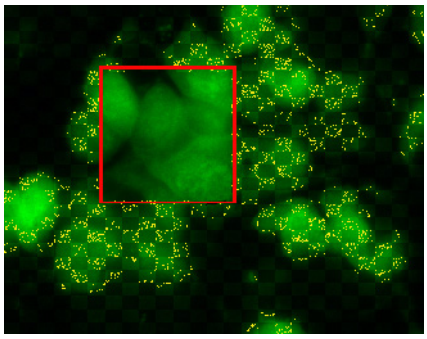
Checker Board	The checkerboard tool displays both portions of both images in an alternating fashion, much like the squares on a checkerboard. Portions of the reference image (Image A) appear where the light-colored squares would appear on the checkerboard; portions of the adjusted image (Image B) appear in place of the dark-colored squares.  You can use the Checkerboard pattern dialog box options to adjust the number of rows and columns displayed.  See also MIPAV User's Guide, Volume 1, Basics, Figure 113 for more infromation about the checkerboard tool.
Magnify image 2.0x	Magnifies the images 2x*n* times.
Magnify image 0.5x	Magnifies the images 0.5x*n* times; or minifies the images 2x*n* times, which is the same.
Window region of Image B	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. 
Reset image to its original state	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.

Figure 150. The Blended tab options (continued)

Apply	Commits the changes that you made to the images, but it leaves the window open for you to make further changes, if desired.
Translate image	Activates **moving mode** 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. 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 403.
Rotate Image	Activates **rotation mode**. 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 403

Figure 150. The Blended tab options (continued)

MOVING AND ROTATING THE ADJUSTED IMAGE

Figure 151 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.

Translate Image

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 151.

Rotate Image

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.

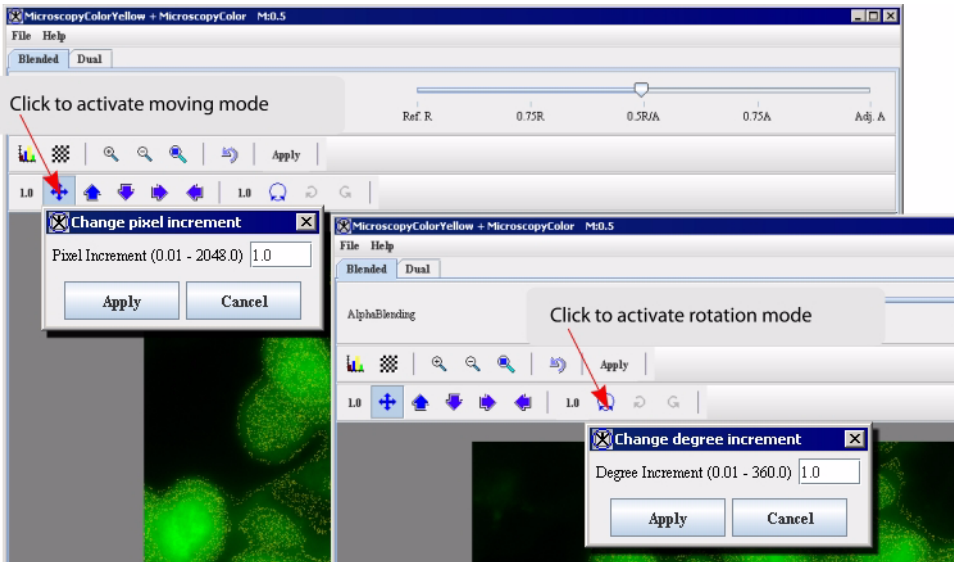


Figure 151. The Blended tab: activating moving and rotation modes

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.

Move up

Move right

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 151.

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 152.

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 of counterclockwise.

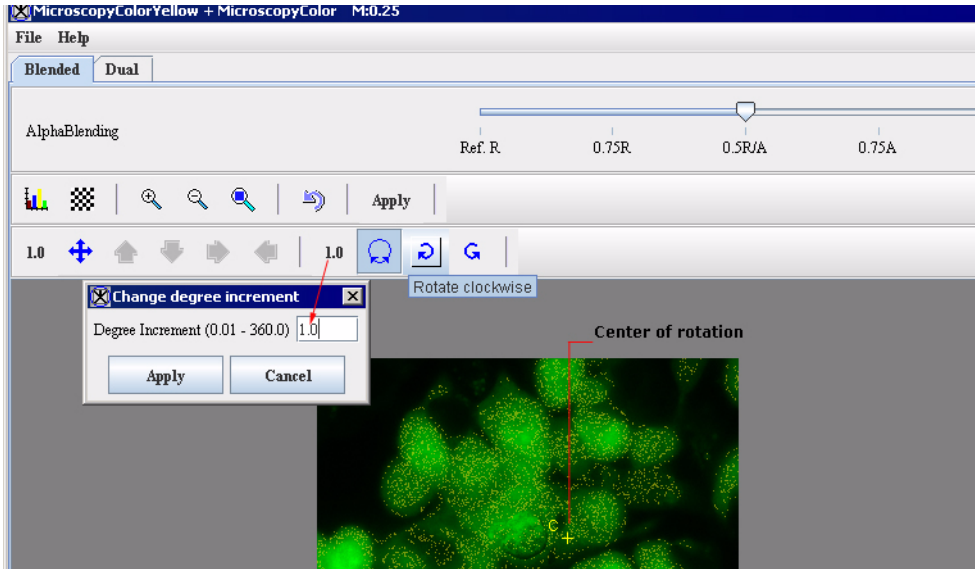


Figure 152. Rotating the adjusted image.

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

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. For the Blended tab options refer to Figure 149.
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	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. To restore reference images to their original size, use the Magnify Reference Image 0.5x icon, as many times as you used this icon.
Magnify reference image 0.5x	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.
Magnify adjusted image 2.0x	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. To restore adjusted images to their original size, use the Magnify Reference Image 0.5x icon, as many times as you used this icon.
Magnify adjusted image 0.5x	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.
Apply least squares alignment	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.

Figure 153. The Dual tab options

Apply thin plate spline alignment	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.
Reset image to original state	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.
Reference slice markers	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 **Return to default mode**, select the point to be deleted, and then click the Delete Selected Reference Slice Markers icon.
Adjusted slice markers	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 on in the order in which you created them. To remove points, return the cursor to its default mode by clicking **Return to default mode**, select the point to be deleted, and click the Delete Selected Adjusted Slice Markers icon. **Note:** landmarks in the reference image should correspond to landmarks in the adjusted image.
Delete selected reference slice markers	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 **Return to default mode**, then select the point to be deleted, and then click the Delete Selected Reference Slice Markers icon.
Delete selected adjusted slice markers	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 **Return to default mode**, then select the point to be deleted, and finally click the Delete Selected Adjusted Slice Markers icon.
Copy reference markers to adjusted image	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.

Figure 153. The Dual tab options (continued)

Copy adjusted markers to reference markers	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.
Return to default mode	Returns the cursor to its default mode, which allows you to change from adding points to deleting points from the image.
Apply	Commits the changes that you made to the images, but it leaves the window open for you to make further changes if desired.

Figure 153. The Dual tab options (continued)

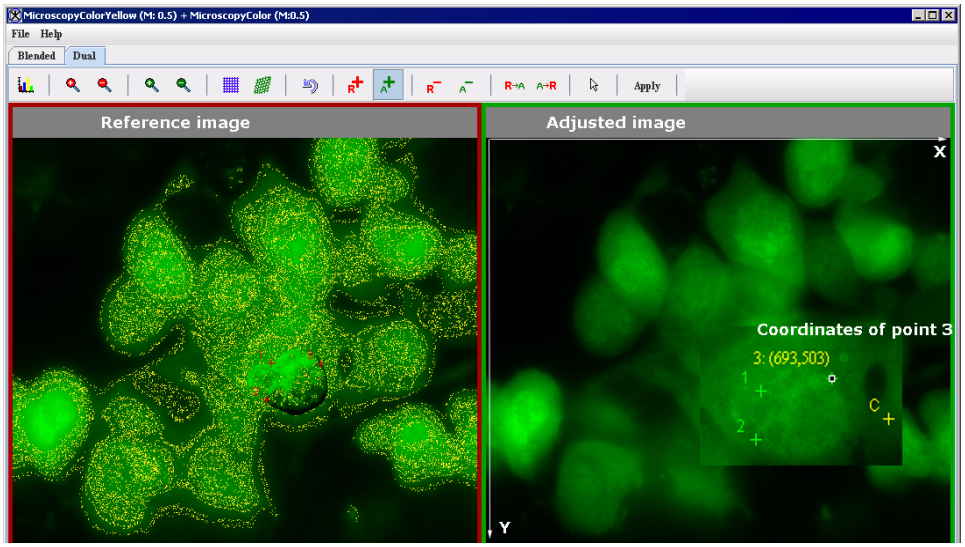


Figure 154. The Dual tab shows both: the reference image and adjusted image side by side. Three corresponding landmark points were delineated in the reference and adjusted image.

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 154.

Delineating the landmark points

Reference Slice Markers

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.

Adjusted Slice Markers

Return to Default mode

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.

Apply Least Squares alignment

Apply Thin Plate Spline alignment

Applying the image registration algorithms

After you have delineated the landmark points on both images, select either the Least Squares algorithm or the Thin Plate Spline algorithm, to register the images.

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 reconcilements to the adjusted image" on page 411.
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

Window region

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:

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 413.

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 155.

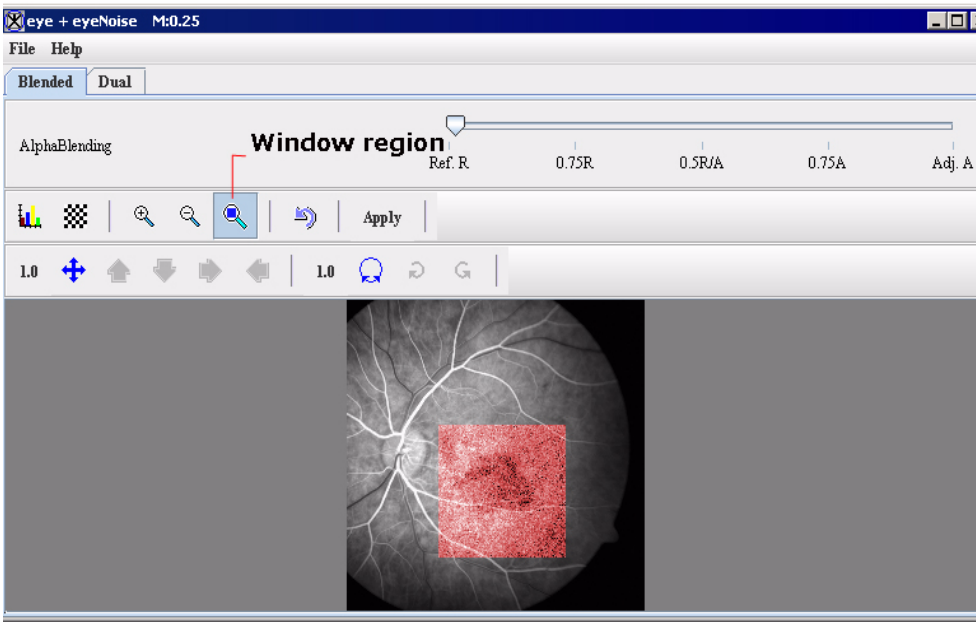


Figure 155. The Window Region box is filled with the adjusted image

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 404.
- Rotating the image using the Rotate Image icon. See also “Rotating the adjusted image” on page 405.

USING LUT TO ARRANGE THE IMAGES

You can use the LUT options to distinguish the adjusted image from the reference image. In order to to 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 156 below.

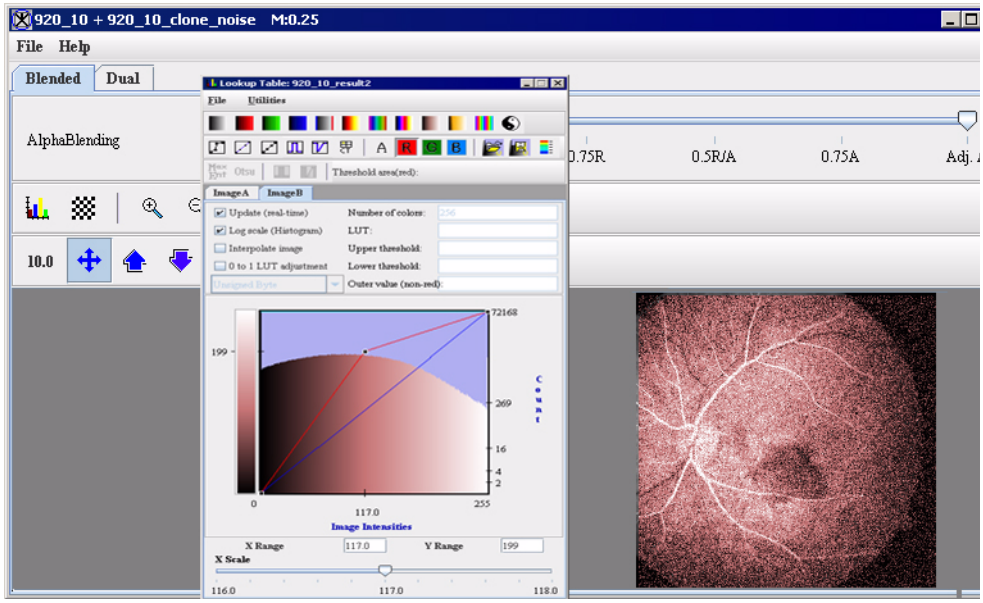


Figure 156. The adjusted image and its Lookup table

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.

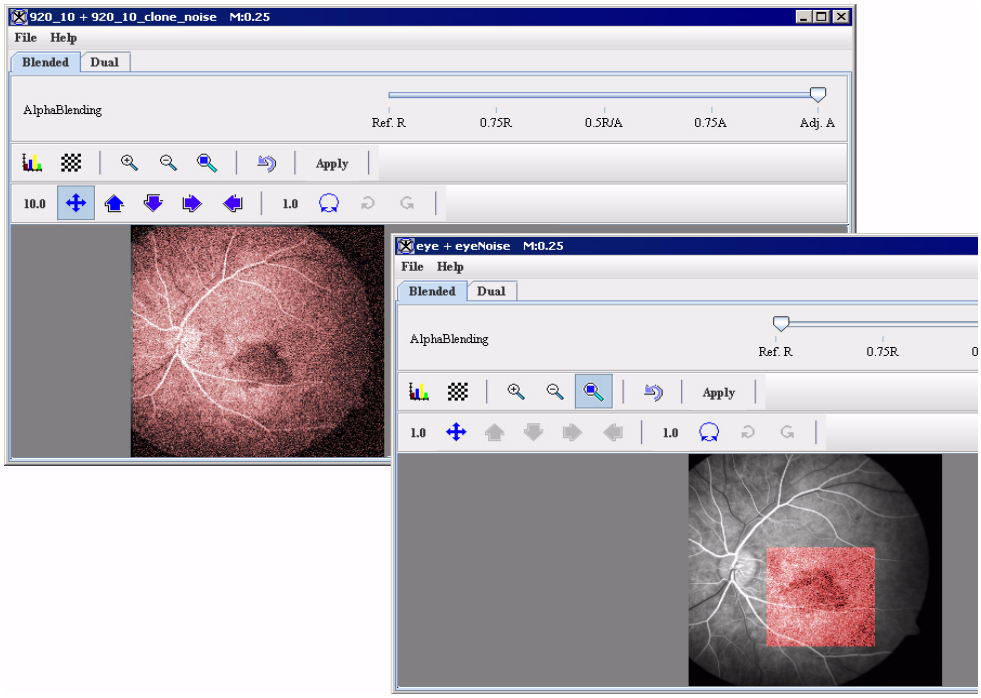


Figure 157. The adjusted image which appears in the Blended tab and also in the Window Region.

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 158) 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 413.
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 158. Use this box to look at alignment of details in the slices.

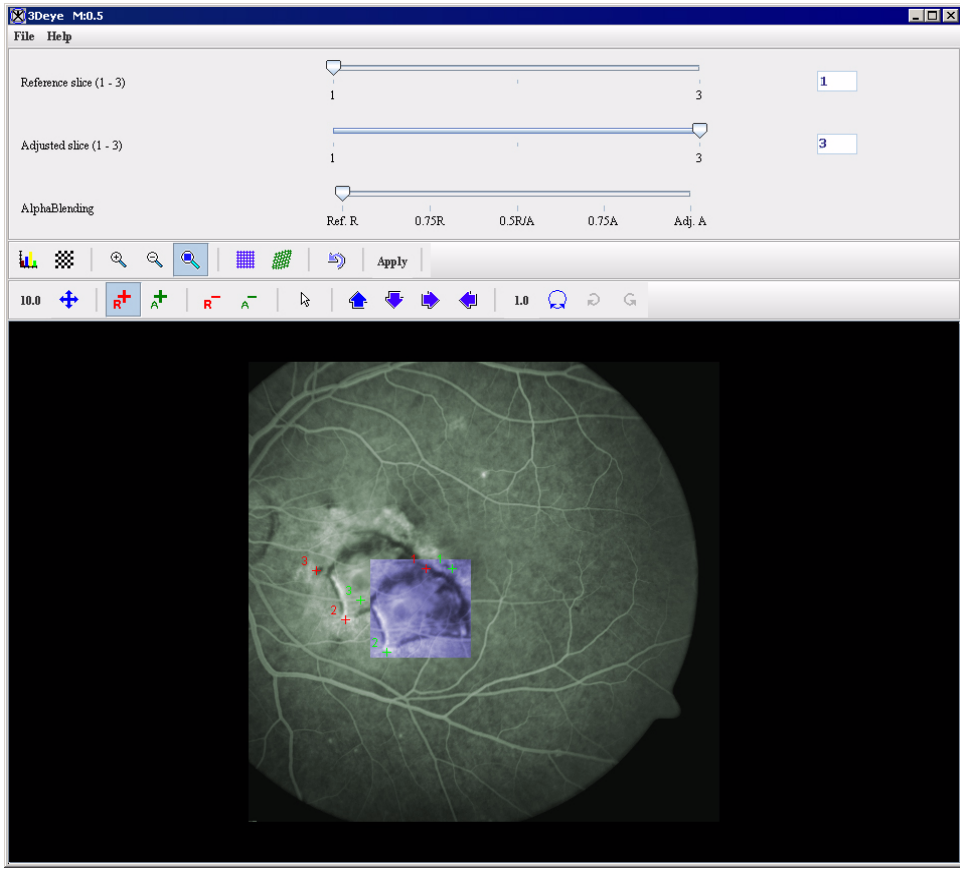


Figure 158. Applying the Manual 2D Series Registration algorithm on a 2.5D image. Slice 3 (adjusted slice) is registered to slice 1 (reference slice). To differentiate the slices, we use LUT options to color the reference slice to pale green and adjusted slice to pale blue.

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 413.
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 159. 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.

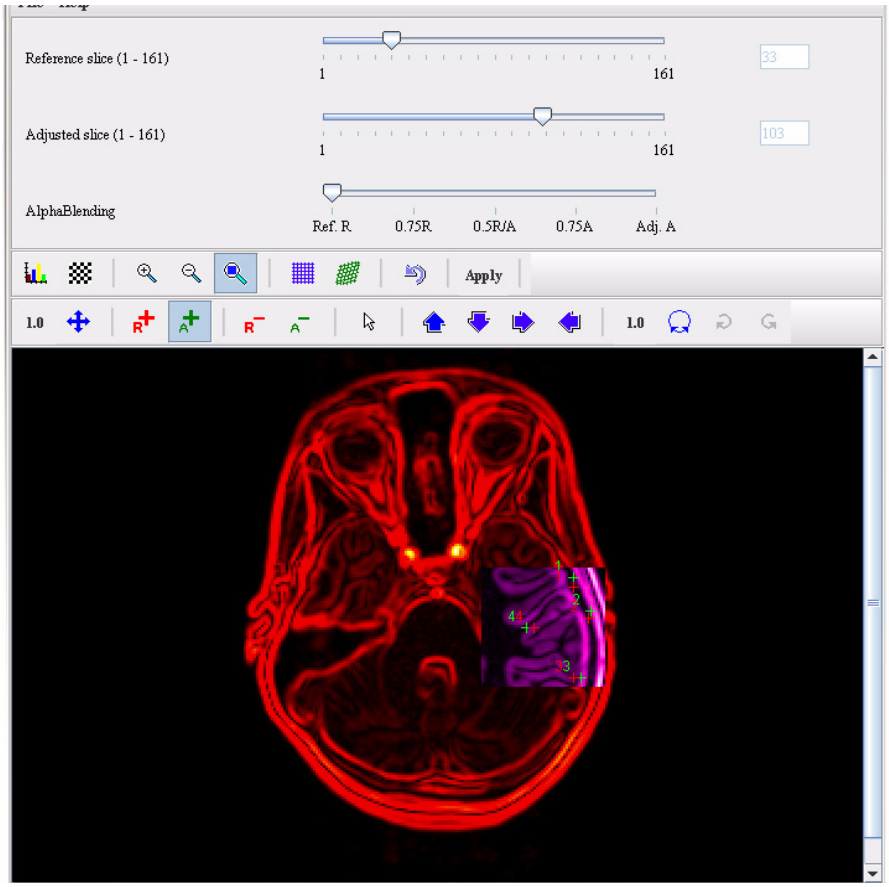


Figure 159. Registering the 3D image.

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 160) 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:

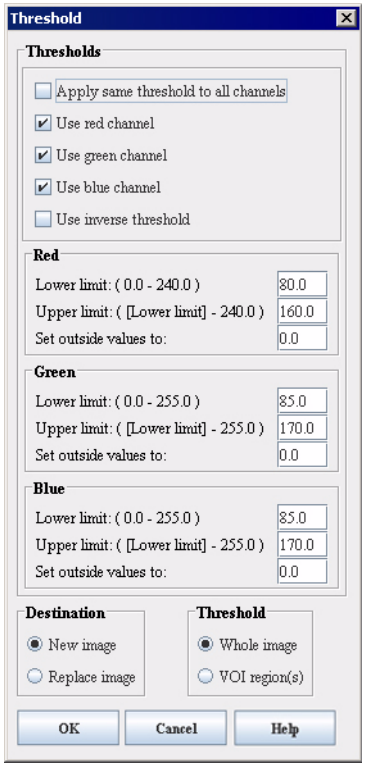


Figure 160. The Threshold dialog box.

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.

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 x 256 x 32, respectively, by a factor of 2 produces a new image with *x*, *y*, and *z* dimensions of 128 x 128 x 16, respectively.

To subsample the images

1. Open an image (Figure 161).
2. Select Algorithms>Transformation tools > Subsample. The Subsample dialog box (Figure 162) opens.

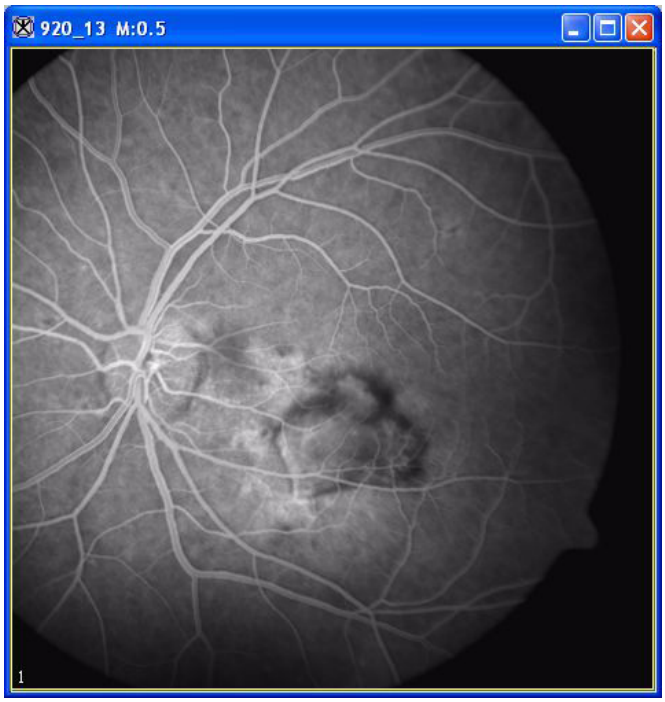


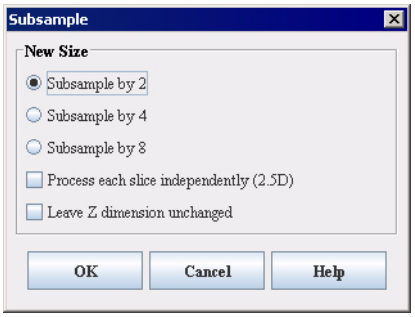
Figure 161. Original image before subsampling

3. Select one of the following:

- Subsample by 2
- Subsample by 4
- Subsample by 8Select, as an option for 2.5D images only, the Process each slice independently (2.5D) check box.

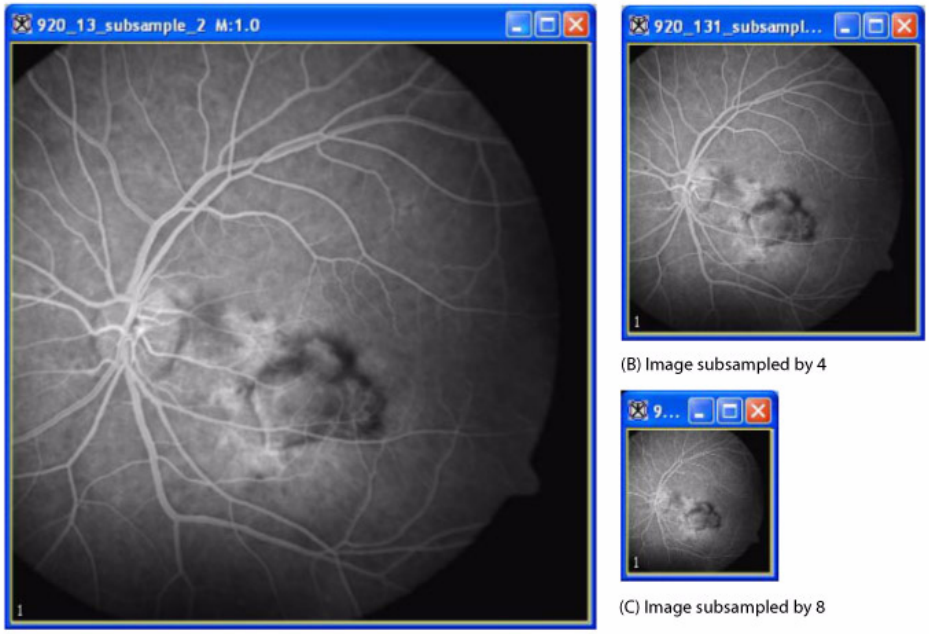
Tip: If you are *not* working with 2.5D images, the Process each slice independently (2.5D) check box does *not* appear in the Subsample dialog box.

4 Click OK. A status message appears briefly while the program generates the subsampled image in a new image window.



Subsample by 2	Subsamples each image dimension by a factor of 2.
Subsample by 4	Subsamples each image dimension by a factor of 4.
Subsample by 8	Subsamples each image dimension by a factor of 8.
Process each slice independently (2.5D)	Filters each slice of the dataset independently of adjacent slices.
OK	Applies the parameters that you specified to subsample the image.
Cancel	Disregards any changes you made in this dialog box, closes the dialog box, and does not subsample the image.
Help	Displays online help for this dialog box.

Figure 162. Subsample dialog box



(A) Image subsampled by 2

(B) Image subsampled by 4

(C) Image subsampled by 8

Figure 163. An image subsampled by 2, by 4, and then by 8

Glossary

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.

D

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 -1.7×10^{308} to 1.7×10^{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 -3.4×10^{38} to 3.4×10^{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.