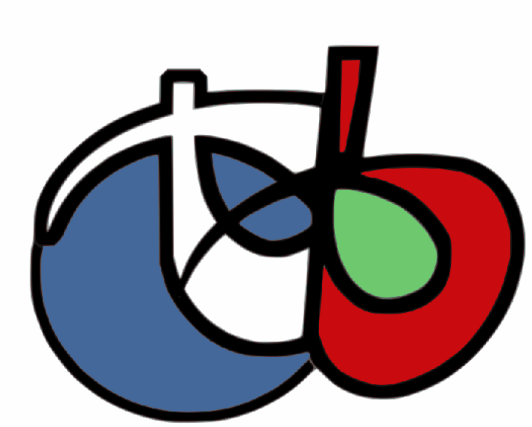
cnes
CENTRE NATIONAL D'ÉTUDES SPATIALES

The Orfeo ToolBox Cookbook, a guide for non-developers Updated for OTB-4.0

OTB Development Team

March 13, 2014

http://www.orfeo-toolbox.org
e-mail: otb@cnes.fr



The ORFEO Toolbox is not a black box.

Ch.D.

Foreword

After almost 5 years of development, the **Orfeo ToolBox** has become a rich library used in many remote sensing context, from research work to operational systems. The **OTB Applications** and more recently the Monteverdi tool has helped to broaden the audience of the library, giving access to its functionalities to non-developers.

Meanwhile, the OTB Software Guide has grown to more than 700 pages of documented code examples, which, combined with the class documentation with the Doxygen, allows developer users to find their way through the **Orfeo ToolBox** so as to write code suiting their needs.

Yet, the documentation available for non-developers users, using Monteverdi and **OTB Applications** to perform everyday remote sensing tasks, has been almost inexistent for all these years, and these users had to learn the software by themselves or ask for help from more experienced users. This cookbook aims at fulfilling the need for an appropriate documentation of the applications built upon the **Orfeo ToolBox**: **Monteverdi**, and **OTB Applications**, which are now integrated into the main **Orfeo ToolBox** package and provide several access mode (command-line, QT interface, QGis plugins, other languages . . .).

A general introduction to these tools is first presented, along with installation instructions. Rather than describing all modules and applications in an exhaustive way, we then decided to focus on very common remote sensing tasks, detailing how they can be achieved with either **Monteverdi** or an application.

For more information on the **Orfeo ToolBox**, please feel free to visit the Orfeo ToolBox website.

CONTENTS

1 A brief tour of OTB-Applications **1**

1.1 Introduction . . . 1

1.2 Installation . . . 2

1.2.1 Windows XP/Seven . . . 3

1.2.2 MacOS X . . . 3

1.2.3 Ubuntu 12.04 and higher . . . 3

1.2.4 OpenSuse 12.X and higher . . . 4

1.3 Using the applications . . . 5

1.3.1 Simplified use . . . 5

1.3.2 Using the command-line launcher . . . 5

1.3.3 Using the GUI launcher . . . 7

1.3.4 Using the Python interface . . . 10

1.3.5 Load/Save OTB-Applications parameters from/to file . . . 11

2 A brief tour of Monteverdi **13**

2.1 Introduction . . . 13

2.2 Installation . . . 13

2.2.1 Windows XP/Seven/8.1 . . . 14

2.2.2 MacOS X . . . 14

2.2.3 Ubuntu 12.04 and higher . . . 14

2.2.4 OpenSuse 11.X and higher . . . 15
2.3 Anatomy of the applications . . . 16
2.3.1 What does it look like? . . . 16
2.3.2 Open an image with **Monteverdi** . . . 17
2.3.3 Visualize an image with **Monteverdi** . . . 17
2.3.4 Cache dataset . . . 19
2.3.5 Dynamic GUI definition . . . 21
2.3.6 Dynamic I/O definition . . . 21
2.4 Available modules . . . 21
2.4.1 I/O operations . . . 21
Extract region of interest . . . 21
Concatenate image bands . . . 21
Save dataset to file . . . 22
2.4.2 Geometric process . . . 22
Reprojection module . . . 23
Estimating sensor model based on ground control points . . . 25
2.4.3 Calibration . . . 25
Optical calibration . . . 25
SAR calibration . . . 26
2.4.4 Filtering Operations . . . 26
Band Math . . . 26
Connected Component Segmentation module . . . 26
Feature extraction . . . 28
Mean-shift segmentation . . . 31
2.4.5 Learning . . . 31
Supervised classification . . . 31
Non-supervised classification . . . 32
2.4.6 Specific SAR functionnalities . . . 32
Despeckle . . . 32
Compute intensity and log-intensity . . . 32
Polarimetry . . . 32

3 A brief tour of Monteverdi2 **35**

3.1 Introduction . . . 35

3.2 Installation . . . 35

3.2.1 Windows XP/Seven/8.1 . . . 36

3.2.2 MacOS X . . . 36

3.2.3 Ubuntu 12.04 and higher . . . 36

3.3 What does it look like? . . . 36

4 Recipes **37**

4.1 Using Pleiades images in **OTB Applications** and **Monteverdi** . . . 38

4.1.1 Opening a Pleiades image in **Monteverdi** . . . 39

4.1.2 Viewing a Pleiades image in **Monteverdi** . . . 40

4.1.3 Handling mega-tiles in **Monteverdi** . . . 41

4.1.4 Partial uncompressing of Pleiades images in **Monteverdi** . . . 42

4.1.5 Other processing of Pleiades images with **Monteverdi** . . . 43

4.1.6 Processing of Pleiades images with **OTB Applications** . . . 43

4.2 Optical pre-processing . . . 43

4.2.1 Optical radiometric calibration . . . 43

Optical calibration with **OTB Applications** . . . 44

Optical calibration with **Monteverdi** . . . 45

4.2.2 Pan-sharpening . . . 45

Pan-sharpening with **OTB Applications** . . . 47

Pan-sharpening with **Monteverdi** . . . 47

4.2.3 Digital Elevation Model management . . . 48

4.2.4 Ortho-rectification and map projections . . . 49

Ortho-rectification with **OTB Applications** . . . 50

4.2.5 Residual registration . . . 53

Extract metadata from the image reference . . . 53

Extract homologous points from images . . . 54

Geometry refinement using homologous points . . . 56

Orthorecrtify image using the affine geometry . . . 56

4.3 Image processing and information extraction . . . 57

4.3.1 Simple calculus with channels . . . 57
4.3.2 Segmentation . . . 57
4.3.3 Large-Scale Mean-Shift (LSMS) segmentation . . . 58
Step 1: Mean-Shift Smoothing . . . 58
Step 2: Segmentation . . . 58
Step 3 (optional): Merging small regions . . . 59
Step 4: Vectorization . . . 60
4.3.4 Dempster Shafer based Classifier Fusion . . . 60
Fuzzy Model (requisite) . . . 60
First Step: Compute Descriptors . . . 61
Second Step: Feature Validation . . . 62
4.4 Classification . . . 62
4.4.1 Pixel based classification . . . 62
Statistics estimation . . . 63
Building the training data set . . . 63
Performing the learning scheme . . . 63
Using the classification model . . . 65
Validating the classification model . . . 65
Fancy classification results . . . 66
Example . . . 66
4.4.2 Fusion of classification maps . . . 66
Majority voting for the fusion of classifications . . . 67
Dempster Shafer framework for the fusion of classifications . . . 69
Recommandations to properly use the fusion of classification maps . . . 70
4.4.3 Majority voting based classification map regularization . . . 70
Majority voting for the classification map regularization . . . 70
Handling ambiguity and not classified pixels in the majority voting based regularization 71
Recommandations to properly use the majority voting based regularization . . . 72
Example . . . 72
4.5 Feature extraction . . . 72
4.5.1 Local statistics extraction . . . 73

4.5.2 Edge extraction . . . 73
4.5.3 Radiometric indices extraction . . . 74
4.5.4 Morphological features extraction . . . 76
Binary morphological operations . . . 76
Gray scale morphological operations . . . 77
4.5.5 Textural features extraction . . . 77
Haralick texture features . . . 77
SFS texture extraction . . . 80
4.6 Stereoscopic reconstruction from VHR optical images pair . . . 81
4.6.1 Estimate epipolar geometry transformation . . . 81
4.6.2 Resample images in epipolar geometry . . . 83
4.6.3 Disparity estimation: Block matching along epipolar lines . . . 83
4.6.4 From disparity to Digital Surface Model . . . 86
4.6.5 One application to rule them all in multi stereo framework scheme . . . 87
4.6.6 Stereo reconstruction good practices . . . 89
4.6.7 Algorithm outline . . . 89

5 Applications Reference Documentation **91**
5.1 Image Manipulation . . . 91
5.1.1 Color Mapping . . . 91
Detailed description . . . 91
Parameters . . . 92
Example . . . 94
Limitations . . . 95
Authors . . . 95
See also . . . 95
5.1.2 Images Concatenation . . . 95
Detailed description . . . 95
Parameters . . . 95
Example . . . 96
Limitations . . . 96
Authors . . . 97

See also . . . 97
5.1.3 Image Conversion . . . 97
Detailed description . . . 97
Parameters . . . 97
Example . . . 99
Limitations . . . 99
Authors . . . 99
See also . . . 99
5.1.4 Download or list SRTM tiles related to a set of images . . . 99
Detailed description . . . 100
Parameters . . . 100
Example . . . 101
Limitations . . . 101
Authors . . . 101
5.1.5 Extract ROI . . . 101
Detailed description . . . 101
Parameters . . . 102
Example . . . 103
Limitations . . . 104
Authors . . . 104
5.1.6 Multi Resolution Pyramid . . . 104
Detailed description . . . 104
Parameters . . . 104
Example . . . 105
Limitations . . . 106
Authors . . . 106
5.1.7 Quick Look . . . 106
Detailed description . . . 106
Parameters . . . 106
Example . . . 108
Limitations . . . 108

Authors . . . 108
5.1.8 Read image information . . . 108
Detailed description . . . 108
Parameters . . . 109
Example . . . 112
Limitations . . . 113
Authors . . . 113
5.1.9 Rescale Image . . . 113
Detailed description . . . 113
Parameters . . . 113
Example . . . 114
Limitations . . . 115
Authors . . . 115
5.1.10 Split Image . . . 115
Detailed description . . . 115
Parameters . . . 115
Example . . . 116
Limitations . . . 116
Authors . . . 116
5.1.11 Image Tile Fusion . . . 116
Detailed description . . . 117
Parameters . . . 117
Example . . . 117
Limitations . . . 118
Authors . . . 118
5.2 Vector Data Manipulation . . . 118
5.2.1 Concatenate . . . 118
Detailed description . . . 118
Parameters . . . 118
Example . . . 119
Limitations . . . 120

Authors . . . 120
5.2.2 Rasterization . . . 120
Detailed description . . . 120
Parameters . . . 120
Example . . . 122
Limitations . . . 123
Authors . . . 123
See also . . . 123
5.2.3 VectorData Extract ROI . . . 123
Detailed description . . . 123
Parameters . . . 123
Example . . . 125
Limitations . . . 125
Authors . . . 125
5.2.4 Vector Data reprojection . . . 125
Detailed description . . . 126
Parameters . . . 126
Example . . . 128
Authors . . . 128
5.2.5 Vector data set field . . . 128
Detailed description . . . 128
Parameters . . . 129
Example . . . 129
Limitations . . . 130
Authors . . . 130
5.2.6 Vector Data Transformation . . . 130
Detailed description . . . 130
Parameters . . . 130
Example . . . 132
Limitations . . . 132
Authors . . . 132

5.3 Calibration . . . 132
5.3.1 Optical calibration . . . 132
Detailed description . . . 133
Parameters . . . 133
Example . . . 135
Limitations . . . 136
Authors . . . 136
See also . . . 136
5.3.2 SAR Radiometric calibration . . . 136
Detailed description . . . 136
Parameters . . . 136
Example . . . 137
Limitations . . . 137
Authors . . . 137
5.4 Geometry . . . 138
5.4.1 Bundle to perfect sensor . . . 138
Detailed description . . . 138
Parameters . . . 138
Example . . . 139
Limitations . . . 140
Authors . . . 140
5.4.2 Cartographic to geographic coordinates conversion . . . 140
Detailed description . . . 140
Parameters . . . 140
Example . . . 142
Limitations . . . 143
Authors . . . 143
5.4.3 Convert Sensor Point To Geographic Point . . . 143
Detailed description . . . 143
Parameters . . . 143
Example . . . 144

Limitations . . . 145
Authors . . . 145
See also . . . 145
5.4.4 Ply 3D files generation . . . 145
Detailed description . . . 145
Parameters . . . 145
Example . . . 147
Authors . . . 147
5.4.5 Generate a RPC sensor model . . . 147
Detailed description . . . 148
Parameters . . . 148
Example . . . 150
Limitations . . . 150
Authors . . . 151
See also . . . 151
5.4.6 Grid Based Image Resampling . . . 151
Detailed description . . . 151
Parameters . . . 151
Example . . . 153
Limitations . . . 154
Authors . . . 154
See also . . . 154
5.4.7 Image Envelope . . . 154
Detailed description . . . 154
Parameters . . . 155
Example . . . 156
Limitations . . . 156
Authors . . . 156
5.4.8 Ortho-rectification . . . 157
Detailed description . . . 157
Parameters . . . 157

Example . . . 161
Limitations . . . 162
Authors . . . 162
See also . . . 162
5.4.9 Pansharpening . . . 162
Detailed description . . . 162
Parameters . . . 162
Example . . . 164
Limitations . . . 164
Authors . . . 164
5.4.10 Refine Sensor Model . . . 164
Detailed description . . . 165
Parameters . . . 165
Example . . . 167
Limitations . . . 168
Authors . . . 168
See also . . . 168
5.4.11 Image resampling with a rigid transform . . . 168
Detailed description . . . 168
Parameters . . . 168
Example . . . 170
Limitations . . . 171
Authors . . . 171
See also . . . 171
5.4.12 Superimpose sensor . . . 171
Detailed description . . . 171
Parameters . . . 172
Example . . . 173
Limitations . . . 174
Authors . . . 174
5.5 Image Filtering . . . 174

5.5.1 Dimensionality reduction . . . 174
Detailed description . . . 174
Parameters . . . 174
Example . . . 176
Limitations . . . 177
Authors . . . 177
See also . . . 177
5.5.2 Mean Shift filtering (can be used as Exact Large-Scale Mean-Shift segmentation, step 1) . . . 177
Detailed description . . . 177
Parameters . . . 177
Example . . . 179
Limitations . . . 179
Authors . . . 179
5.5.3 Smoothing . . . 179
Detailed description . . . 179
Parameters . . . 180
Examples . . . 181
Limitations . . . 182
Authors . . . 182
5.6 Feature Extraction . . . 182
5.6.1 Binary Morphological Operation . . . 182
Detailed description . . . 182
Parameters . . . 182
Example . . . 184
Limitations . . . 185
Authors . . . 185
See also . . . 185
5.6.2 Compute Polyline Feature From Image . . . 185
Detailed description . . . 185
Parameters . . . 186
Example . . . 187

Limitations . . . 188
Authors . . . 188
5.6.3 Fuzzy Model estimation . . . 188
Detailed description . . . 188
Parameters . . . 188
Example . . . 189
Limitations . . . 190
Authors . . . 190
5.6.4 Edge Feature Extraction . . . 190
Detailed description . . . 190
Parameters . . . 191
Example . . . 192
Limitations . . . 192
Authors . . . 192
See also . . . 192
5.6.5 Grayscale Morphological Operation . . . 193
Detailed description . . . 193
Parameters . . . 193
Example . . . 194
Limitations . . . 195
Authors . . . 195
See also . . . 195
5.6.6 Haralick Texture Extraction . . . 195
Detailed description . . . 195
Parameters . . . 195
Example . . . 197
Limitations . . . 198
Authors . . . 198
See also . . . 198
5.6.7 Homologous Points Extraction . . . 198
Detailed description . . . 198

Parameters . . . 199
Example . . . 202
Limitations . . . 202
Authors . . . 202
See also . . . 202
5.6.8 Line segment detection . . . 203
Detailed description . . . 203
Parameters . . . 203
Example . . . 204
Limitations . . . 205
Authors . . . 205
See also . . . 205
5.6.9 Local Statistic Extraction . . . 205
Detailed description . . . 205
Parameters . . . 205
Example . . . 206
Limitations . . . 207
Authors . . . 207
See also . . . 207
5.6.10 Multivariate alteration detector . . . 207
Detailed description . . . 207
Parameters . . . 207
Example . . . 208
Limitations . . . 208
Authors . . . 208
See also . . . 209
5.6.11 Radiometric Indices . . . 209
Detailed description . . . 210
Parameters . . . 210
Example . . . 211
Limitations . . . 212

Authors . . . 212
See also . . . 212
5.6.12 SFS Texture Extraction . . . 212
Detailed description . . . 212
Parameters . . . 212
Example . . . 214
Limitations . . . 214
Authors . . . 214
See also . . . 214
5.6.13 Vector Data validation . . . 215
Detailed description . . . 215
Parameters . . . 215
Example . . . 216
Limitations . . . 216
Authors . . . 216
See also . . . 216
5.7 Stereo . . . 217
5.7.1 Pixel-wise Block-Matching . . . 217
Detailed description . . . 217
Parameters . . . 217
Example . . . 221
Limitations . . . 221
Authors . . . 221
See also . . . 222
5.7.2 Disparity map to elevation map . . . 222
Detailed description . . . 222
Parameters . . . 222
Example . . . 224
Limitations . . . 224
Authors . . . 225
See also . . . 225

5.7.3 Fine Registration . . . 225
Detailed description . . . 225
Parameters . . . 225
Example . . . 227
Limitations . . . 228
Authors . . . 228
5.7.4 Stereo Framework . . . 228
Detailed description . . . 228
Parameters . . . 228
Example . . . 234
Authors . . . 235
5.7.5 Stereo-rectification deformation grid generator . . . 235
Detailed description . . . 235
Parameters . . . 235
Example . . . 238
Limitations . . . 238
Authors . . . 238
See also . . . 238
5.8 Learning . . . 239
5.8.1 Classification Map Regularization . . . 239
Detailed description . . . 239
Parameters . . . 239
Example . . . 240
Limitations . . . 241
Authors . . . 241
See also . . . 241
5.8.2 Confusion matrix Computation . . . 241
Detailed description . . . 241
Parameters . . . 242
Example . . . 243
Limitations . . . 243

Authors . . . 244
5.8.3 Compute Images second order statistics . . . 244
Detailed description . . . 244
Parameters . . . 244
Example . . . 245
Limitations . . . 245
Authors . . . 245
See also . . . 245
5.8.4 Fusion of Classifications . . . 245
Detailed description . . . 245
Parameters . . . 246
Example . . . 247
Limitations . . . 248
Authors . . . 248
See also . . . 248
5.8.5 Image Classification . . . 248
Detailed description . . . 249
Parameters . . . 249
Example . . . 250
Limitations . . . 250
Authors . . . 250
See also . . . 250
5.8.6 Unsupervised KMeans image classification . . . 251
Detailed description . . . 251
Parameters . . . 251
Example . . . 252
Limitations . . . 252
Authors . . . 253
5.8.7 SOM Classification . . . 253
Detailed description . . . 253
Parameters . . . 253

Example . . . 255
Limitations . . . 255
Authors . . . 255
5.8.8 Train a classifier from multiple images . . . 256
Detailed description . . . 256
Parameters . . . 256
Example . . . 264
Limitations . . . 265
Authors . . . 265
See also . . . 265
5.9 Segmentation . . . 265
5.9.1 Connected Component Segmentation . . . 265
Detailed description . . . 266
Parameters . . . 266
Example . . . 267
Limitations . . . 268
Authors . . . 268
5.9.2 Hoover compare segmentation . . . 268
Detailed description . . . 268
Parameters . . . 269
Example . . . 270
Limitations . . . 270
Authors . . . 270
See also . . . 271
5.9.3 Exact Large-Scale Mean-Shift segmentation, step 2 . . . 271
Detailed description . . . 271
Parameters . . . 271
Example . . . 273
Limitations . . . 273
Authors . . . 273
See also . . . 273

5.9.4 Exact Large-Scale Mean-Shift segmentation, step 3 (optional) . . . 274
Detailed description . . . 274
Parameters . . . 274
Example . . . 275
Limitations . . . 275
Authors . . . 275
See also . . . 276
5.9.5 Exact Large-Scale Mean-Shift segmentation, step 4 . . . 276
Detailed description . . . 276
Parameters . . . 276
Example . . . 277
Limitations . . . 277
Authors . . . 278
See also . . . 278
5.9.6 Segmentation . . . 278
Detailed description . . . 278
Parameters . . . 278
Examples . . . 282
Limitations . . . 283
Authors . . . 283
See also . . . 284
5.10 Miscellanous . . . 284
5.10.1 Band Math . . . 284
Detailed description . . . 284
Parameters . . . 284
Example . . . 285
Limitations . . . 285
Authors . . . 285
5.10.2 Images comparaison . . . 285
Detailed description . . . 286
Parameters . . . 286

Example . . . 287
Limitations . . . 288
Authors . . . 288
See also . . . 288
5.10.3 Hyperspectral data unmixing . . . 288
Detailed description . . . 288
Parameters . . . 289
Example . . . 290
Limitations . . . 290
Authors . . . 290
See also . . . 290
5.10.4 Image to KMZ Export . . . 291
Detailed description . . . 291
Parameters . . . 291
Example . . . 292
Limitations . . . 293
Authors . . . 293
See also . . . 293
5.10.5 Open Street Map layers importations applications . . . 293
Detailed description . . . 293
Parameters . . . 293
Example . . . 295
Limitations . . . 295
Authors . . . 295
See also . . . 296
5.10.6 Obtain UTM Zone From Geo Point . . . 296
Detailed description . . . 296
Parameters . . . 296
Example . . . 297
Limitations . . . 297
Authors . . . 297

5.10.7 Pixel Value . . . 297
Detailed description . . . 297
Parameters . . . 297
Example . . . 298
Limitations . . . 299
Authors . . . 299
5.10.8 Vertex Component Analysis . . . 299
Detailed description . . . 299
Parameters . . . 299
Example . . . 300
Limitations . . . 300
Authors . . . 300

LIST OF FIGURES

1.1 GUI of the application Rescale, parameters tab 8
1.2 GUI of the application Rescale, logs tab 8
1.3 GUI of the application Rescale, progress tab 9
1.4 GUI of the application Rescale, parameters tab 9

2.1 Monteverdi main window 16
2.2 Monteverdi main window 17
2.3 Monteverdi main window 18
2.4 Vector Data visualization 18
2.5 Manage RGB composition 19
2.6 Manage the dynamic of each layer 19
2.7 Pixel description via index selection 20
2.8 Cache operation on the panchromatic image in progress 20
2.9 Concatenation of multitemporal data 22
2.10 Export dataset 23
2.11 Preview of the orthorectified imagery 24
2.12 Band math module 26
2.13 Difference of NDWI2 on 2 images 27
2.14 Connected Component Segmentation 29
2.15 Example Boats 30

2.16 Example Boats result . . . 30
2.17 Mean-shift clustering . . . 31
2.18 SAR polarimetry conversion . . . 33

4.1 **Monteverdi** Pleiades image opening . . . 39
4.2 Pleiades dataset in **Monteverdi** . . . 39
4.3 Pleiades image in **Monteverdi** viewer . . . 41
4.4 Handling Pleiades mega-tiles in **Monteverdi** . . . 41
4.5 **Monteverdi** Pleiades uncompress module . . . 42
4.6 GUI of the optical calibration module based on the 6S model . . . 45
4.7 Output of the optical calibration module . . . 46
4.8 Input panchromatic and zoomed and registered multispectral image over it . . . 48
4.9 Pan-sharpened image . . . 48
4.10 InputImagesRegistration . . . 54
4.11 GUI of the vectorization module with data for classification chain . . . 64
4.12 ExampleSVMClassif . . . 67
4.13 ExampleSVMClassifFusion . . . 68
4.14 ExampleSVMClassifFusionMV . . . 69
4.15 ExampleSVMClassifFusionDS . . . 70
4.16 ExampleSVMClassifCMR . . . 72
4.17 ExampleEpipolar . . . 84
4.18 ExampleEpipolar . . . 86
4.19 Example . . . 88

5.1 Parameters table for Color Mapping. . . . 93
5.2 Parameters table for Images Concatenation. . . . 96
5.3 Parameters table for Image Conversion. . . . 98
5.4 Parameters table for Download or list SRTM tiles related to a set of images. . . . 100
5.5 Parameters table for Extract ROI. . . . 102
5.6 Parameters table for Multi Resolution Pyramid. . . . 105
5.7 Parameters table for Quick Look. . . . 107
5.8 Parameters table for Read image information. . . . 110

5.9 Parameters table for Rescale Image. 114
5.10 Parameters table for Split Image. 115
5.11 Parameters table for Image Tile Fusion. 117
5.12 Parameters table for Concatenate. 119
5.13 Parameters table for Rasterization. 121
5.14 Parameters table for VectorData Extract ROI. 124
5.15 Parameters table for Vector Data reprojection. 126
5.16 Parameters table for Vector data set field. 129
5.17 Parameters table for Vector Data Transformation. 131
5.18 Parameters table for Optical calibration. 134
5.19 Parameters table for SAR Radiometric calibration. 136
5.20 Parameters table for Bundle to perfect sensor. 138
5.21 Parameters table for Cartographic to geographic coordinates conversion. 141
5.22 Parameters table for Convert Sensor Point To Geographic Point. 143
5.23 Parameters table for Ply 3D files generation. 146
5.24 Parameters table for Generate a RPC sensor model. 148
5.25 Parameters table for Grid Based Image Resampling. 152
5.26 Parameters table for Image Envelope. 155
5.27 Parameters table for Ortho-rectification. 158
5.28 Parameters table for Pansharpening. 163
5.29 Parameters table for Refine Sensor Model. 165
5.30 Parameters table for Image resampling with a rigid transform. 169
5.31 Parameters table for Superimpose sensor. 172
5.32 Parameters table for Dimensionality reduction. 175
5.33 Parameters table for Mean Shift filtering (can be used as Exact Large-Scale Mean-Shift segmentation, step 1). 178
5.34 Parameters table for Smoothing. 180
5.35 Parameters table for Binary Morphological Operation. 183
5.36 Parameters table for Compute Polyline Feature From Image. 186
5.37 Parameters table for Fuzzy Model estimation. 189
5.38 Parameters table for Edge Feature Extraction. 191
5.39 Parameters table for Grayscale Morphological Operation. 193

5.40 Parameters table for Haralick Texture Extraction. 196
5.41 Parameters table for Homologous Points Extraction. 200
5.42 Parameters table for Line segment detection. 203
5.43 Parameters table for Local Statistic Extraction. 206
5.44 Parameters table for Multivariate alteration detector. 207
5.45 Parameters table for Radiometric Indices. 210
5.46 Parameters table for SFS Texture Extraction. 213
5.47 Parameters table for Vector Data validation. 215
5.48 Parameters table for Pixel-wise Block-Matching. 218
5.49 Parameters table for Disparity map to elevation map. 223
5.50 Parameters table for Fine Registration. 226
5.51 Parameters table for Stereo Framework. 230
5.52 Parameters table for Stereo-rectification deformation grid generator. 236
5.53 Parameters table for Classification Map Regularization. 239
5.54 Parameters table for Confusion matrix Computation. 242
5.55 Parameters table for Compute Images second order statistics. 244
5.56 Parameters table for Fusion of Classifications. 246
5.57 Parameters table for Image Classification. 249
5.58 Parameters table for Unsupervised KMeans image classification. 251
5.59 Parameters table for SOM Classification. 253
5.60 Parameters table for Train a classifier from multiple images. 259
5.61 Parameters table for Connected Component Segmentation. 266
5.62 Parameters table for Hoover compare segmentation. 269
5.63 Parameters table for Exact Large-Scale Mean-Shift segmentation, step 2. 272
5.64 Parameters table for Exact Large-Scale Mean-Shift segmentation, step 3 (optional). 274
5.65 Parameters table for Exact Large-Scale Mean-Shift segmentation, step 4. 276
5.66 Parameters table for Segmentation. 280
5.67 Parameters table for Band Math. 284
5.68 Parameters table for Images comparaison. 286
5.69 Parameters table for Hyperspectral data unmixing. 289
5.70 Parameters table for Image to KMZ Export. 291

5.71 Parameters table for Open Street Map layers importations applications. 294
5.72 Parameters table for Obtain UTM Zone From Geo Point. 296
5.73 Parameters table for Pixel Value. 298
5.74 Parameters table for Vertex Component Analysis. 299

LIST OF TABLES

CHAPTER

ONE

A brief tour of OTB-Applications

1.1 Introduction

OTB Applications was perhaps the older package of the **Orfeo ToolBox** suite after the OTB package itself. Since the **Orfeo ToolBox** is a library providing remote sensing functionalities, the only applications that were distributed at the beginning were the examples from the Software Guide and the tests. These applications are very useful for the developer because their code is very short and only demonstrates one functionality at a time. In many cases, a real application would require :

- combining together two or more functions from the **Orfeo ToolBox**
- providing a nice high level interface to handle : parameters, input data, output data and communication with the user

The **OTB Applications** package was originally designed to provide applications performing simple remote sensing tasks, more complex than simple examples from the Software Guide, and with a more user-friendly interface (either graphical or command-line), to demonstrate the use of the **Orfeo ToolBox** functions. The most popular applications are maybe the *otbImageViewerManager*, which allows to open a collection of images and navigate in them, and the *otbSupervisedClassificationApplication*, which allowed to delineate training regions of interest on the image and classify the image with a SVM classifier trained with these regions (this application is no longer maintained since the same functionnality is available through the corresponding **Monteverdi** module). During the first 3 years of the **Orfeo ToolBox** development, many more applications have been added to this package, to perform various tasks. Most of them came with a graphical user interface, apart from some small utilities that are command-line.

The development and release of the **Monteverdi** software (see chapter 2 at the end of year 2009 changed a lot of things for the **OTB Applications** package: most of non-developer users were looking for quite a long time for an application providing **Orfeo ToolBox** functionalities under a unified graphical interface. Many applications from the **OTB Applications** package were integrated to **Monteverdi** as modules, and the **OTB Applications** package lost a lot of its usefulness. No more

applications were added to the package and it was barely maintained, as new graphical tools were directly embedded within **Monteverdi**.

Then, some people started to regain interest in the **OTB Applications** package. **Monteverdi** is a great tool to perform numerous remote sensing and image processing task in a minute, but it is not well adapted to heavier (and longer) processing, scripting and batch processing. Therefore, in 2010 the **OTB Applications** package has been revamped: old applications have been moved to a legacy folder for backward compatibility, and the development team started to populate the package with compact command-line tools to perform various heavy processing tasks.

Later on in 2011, the **OTB Applications** has been further revamped. Because of the increasing need to interface the **OTB Applications** into other software and to provide auto-generated interfaces, the **Orfeo ToolBox** development team decided to develop a new application framework. The main idea of this framework is the following: each application is written once for all in a shared library (also known as plugin). This plugin can be auto-loaded into appropriate tools wihtout recompiling, and is able to fully describe its parameters, behaviour and documentation.

The tools to use the plugins can be extended, but **Orfeo ToolBox** shipped the following:

- A command-line laucher, which is almost equivalent to the former **OTB Applications** command-line interface,
- A graphical launcher, with an auto-generated QT interface, providing ergonomic parameters setting, display of documentation, and progress reporting,
- A SWIG interface, which means that any application can be loaded set-up and executed into a high-level language such as Python or Java for instance.

Additionally, QGis plugins built on top of the SWIG/Python interface are available with seamless integration within QGis. You can find a short guide about it here.

To facilitate the use of these tools and applications, they will now be shipped with the standard **Orfeo ToolBox** package. It means that the former **OTB-Applications** package has entered its maintenance cycle : no new feature will be pushed there, and all development is done directly inside the **Orfeo ToolBox** paackage.

The **OTB Applications** are now rich of more than 40 tools, which are listed in the the applications reference documentation, presented in chapter 5, page 91.

1.2 Installation

We provide different binary packages for OTB-Applications:

- for Windows platform (XP/Seven) through OsGeo4W installer (32/64bit)
- for Ubuntu 12.04 and higher

- for OpenSuse 12.X and higher
- for MacOSX through MacPorts software

If you want build from source or if we don't provide packages for your system, some informations are available into the OTB Software Guide, in the section (Building from Source)

1.2.1 Windows XP/Seven

Since version 3.12, we provide OTB Applications packages through OSGeo4W for Windows XP/-Seven users:

- **otb-bin** for command line and QT applications
- **otb-python** for python applications

Follow the instructions in the installer and select the packages you want to add. The installer will proceed with the installation of selected packages and all their dependencies. For the **otb-bin** packages, it will be available directly in the OSGeo4W shell, for example run

```
otbgui_BandMath.
```

For the **otb-python** packages, you can simply check from an OSGeo4W shell the list of available applications:

```
python
import otbApplication
print str( otbApplication.Registry.GetAvailableApplications() )
```

1.2.2 MacOS X

OTB Applications are now available on MacPorts. The port name is called orfeotoolbox. You can follow the MacPorts documentation to install MacPorts first, then install the orfeotoolbox port. After the installation, you can used directly on your system, the OTB applications.

1.2.3 Ubuntu 12.04 and higher

For Ubuntu 12.04 and higher, OTB Applications packages may be available as Debian packages through APT repositories:

- **otb-bin** for command line applications

- **otb-bin-qt** for Qt applications
- **python-otb** for python applications

Since release 3.14.1, OTB Applications packages are available in the ubuntugis-unstable repository.

You can add it by using these command-lines:

```
sudo aptitude install add-apt-repository
sudo apt-add-repository ppa:ubuntugis/ubuntugis-unstable
```

After you can run:

```
sudo aptitude install otb-bin otb-bin-qt python-otb
```

If you are using *Synaptic*, you can add the repositories, update and install the packages through the graphical interface.

For further informations about Ubuntu packages go to ubuntugis-unstable launchpad page and click on **Read about installing**.

apt-add-repository will try to retrieve the GPG keys of the repositories to certify the origin of the packages. If you are behind a http proxy, this step won't work and apt-add-repository will stall and eventually quit. You can temporarily ignore this error and proceed with the update step. Following this, aptitude update will issue a warning about a signature problem. This warning won't prevent you from installing the packages.

1.2.4 OpenSuse 12.X and higher

For OpenSuse 12.X and higher, OTB Applications packages are available through *zypper*.

First, you need to add the appropriate repositories with these command-lines (please replace 11.4 by your OpenSuse version):

```
sudo zypper ar
http://download.opensuse.org/repositories/games/openSUSE_11.4/ Games
sudo zypper ar
http://download.opensuse.org/repositories/Application:/Geo/openSUSE_11.4/ GEO
sudo zypper ar
http://download.opensuse.org/repositories/home:/tzotsos/openSUSE_11.4/ tzotsos
```

Now run:

```
sudo zypper refresh
sudo zypper install OrfeoToolbox
sudo zypper install OrfeoToolbox-python
```

Alternatively you can use the One-Click Installer from the openSUSE Download page or add the above repositories and install through Yast Package Management.

There is also support for the recently introduced 'rolling' openSUSE distribution named 'Tumbleweed'. For Tumbleweed you need to add the following repositories with these command-lines:

```
sudo zypper ar
http://download.opensuse.org/repositories/games/openSUSE_Tumbleweed/ Games
sudo zypper ar
http://download.opensuse.org/repositories/Application:/Geo/openSUSE_Tumbleweed/ GEO
sudo zypper ar
http://download.opensuse.org/repositories/home:/tzotsos/openSUSE_Tumbleweed/ tzotsos
```

and then add the OTB packages as shown above.

1.3 Using the applications

Using the new **OTB Applications** framework is slightly more complex than launching a command-line tool. This section describes all the ways to launch the new applications. Apart from the simplified access, which is similar to the former access to **OTB Applications**, you will need to know the application name and optionally the path where the applications plugins are stored. For applications shipped with **Orfeo ToolBox**, the name of each application can be found in chapter 5, page 91.

1.3.1 Simplified use

All standard applications delivered in with **Orfeo ToolBox** comes with simplified scripts in the system path, allowing to launch the command-line and graphical user interface versions of the application in the same simple way we used to launch the old applications. The command-line interface is prefixed by `otbcli_`, while the Qt interface is prefixed by `otbgui_`. For instance, calling `otbcli_Convert` will launch the command-line interface of the **Convert** application, while `otbgui_Convert` will launch its GUI.

Passing arguments to the command-line version (prefixed by `otbcli_`) is explained in next subsection.

1.3.2 Using the command-line launcher

The command-line application launcher allows to load an application plugin, to set its parameters, and execute it using the command line. Launching the `otbApplicationLauncherCommandLine` without argument results in the following help to be displayed:

```
$ otbApplicationLauncherCommandLine
Usage : ./otbApplicationLauncherCommandLine module_name [MODULEPATH] [arguments]
```

The `module_name` parameter corresponds to the application name. The `[MODULEPATH]` argument is optional and allows to pass to the launcher a path where the shared library (or plugin) corresponding to `module_name` is.

It is also possible to set this path with the environment variable `ITK_AUTOLOAD_PATH`, making the `[MODULEPATH]` optional. This variable is checked by default when no `[MODULEPATH]` argument is given. When using multiple paths in `ITK_AUTOLOAD_PATH`, one must make sure to use the standard path separator of the target system, which is `:` on Unix, and `;` on Windows.

An error in the application name (i.e. in parameter `module_name`) will make the `otbApplicationLauncherCommandLine` lists the name of all applications found in the available path (either `[MODULEPATH]` and/or `ITK_AUTOLOAD_PATH`).

To ease the use of the applications, and try avoiding extensive environment customization, ready-to-use scripts are provided by the OTB installation to launch each application, and takes care of adding the standard application installation path to the `ITK_AUTOLOAD_PATH` environment variable.

These scripts are named `otbcli_<ApplicationName>` and do not need any path settings. For example you can start the Orthorectification application with the script called `otbcli_Orthorectification`.

Launching an application with no or incomplete parameters will make the launcher display a summary of the parameters, indicating the mandatory parameters missing to allow for application execution. Here is an example with the **OrthoRectification** application:

```
$ otbcli_OrthoRectification

ERROR: Waiting for at least one parameter...

====================== HELP CONTEXT ======================
NAME: OrthoRectification
DESCRIPTION: This application allows to ortho-rectify optical images from supported sensors.

EXAMPLE OF USE:
otbcli_OrthoRectification -io.in QB_TOULOUSE_MUL_Extract_500_500.tif -io.out QB_Toulouse_ortho.tif

DOCUMENTATION: http://www.orfeo-toolbox.org/Applications/OrthoRectification.html
======================== PARAMETERS ======================
        -progress                       <boolean>        Report progress
MISSING -io.in                          <string>         Input Image
MISSING -io.out                         <string> [pixel] Output Image  [pixel=uint8/int8/uint16/int16/uint32/int32/float/double]
        -map                            <string>         Output Map Projection [utm/lambert2/lambert93/transmercator/wgs/epsg]
MISSING -map.utm.zone                   <int32>          Zone number
        -map.utm.northhem               <boolean>        Northern Hemisphere
        -map.transmercator.falseeasting <float>          False easting
        -map.transmercator.falsenorthing <float>         False northing
        -map.transmercator.scale        <float>          Scale factor
        -map.epsg.code                  <int32>          EPSG Code
        -outputs.mode                   <string>         Parameters estimation modes [auto/autosize/autospacing]
MISSING -outputs.ulx                    <float>          Upper Left X
MISSING -outputs.uly                    <float>          Upper Left Y
MISSING -outputs.sizex                  <int32>          Size X
```

```
MISSING -outputs.sizey                <int32>          Size Y
MISSING -outputs.spacingx             <float>          Pixel Size X
MISSING -outputs.spacingy             <float>          Pixel Size Y
        -outputs.isotropic            <boolean>        Force isotropic spacing by default
        -elev.dem                     <string>         DEM directory
        -elev.geoid                   <string>         Geoid File
        -elev.default                 <float>          Average Elevation
        -interpolator                 <string>         Interpolation [nn/linear/bco]
        -interpolator.bco.radius      <int32>          Radius for bicubic interpolation
        -opt.rpc                      <int32>          RPC modeling (points per axis)
        -opt.ram                      <int32>          Available memory for processing (in MB)
        -opt.gridspacing              <float>          Resampling grid spacing
```

For a detailed description of the application behaviour and parameters, please check the application reference documentation presented chapter 5, page 91 or follow the DOCUMENTATION hyperlink provided in otbApplicationLauncherCommandLine output. Parameters are passed to the application using the parameter key (which might include one or several . character), prefixed by a -. Command-line examples are provided in chapter 5, page 91.

1.3.3 Using the GUI launcher

The graphical interface for the applications provides a usefull interactive user interface to set the parameters, choose files, and monitor the execution progress.

This interface can be activated through the CMake option OTB_WRAP_QT.

This launcher needs the same two arguments as the command line launcher :

```
$ otbApplicationLauncherQt module_name [MODULEPATH]
```

The application paths can be set with the ITK_AUTOLOAD_PATH environment variable, as for the command line launcher. Also, as for the command-line application, a more simple script is generated and installed by OTB to ease the configuration of the module path : to launch the *Rescale* graphical user interface, one will start the otbgui_Rescale script.

The resulting graphical application displays a window with several tabs:

- **Parameters** is where you set the parameters and execute the application.
- **Logs** is where you see the informations given by the application during its execution.
- **Progress** is where you see a progress bar of the execution (not available for all applications).
- **Documentation** is where you find a summary of the application documentation.

In this interface, every optional parameter has a check box that you have to tick if you want to set a value and use this parameter. The mandatory parameters cannot be unchecked.

The interface of the application *Rescale* is shown here as an example.

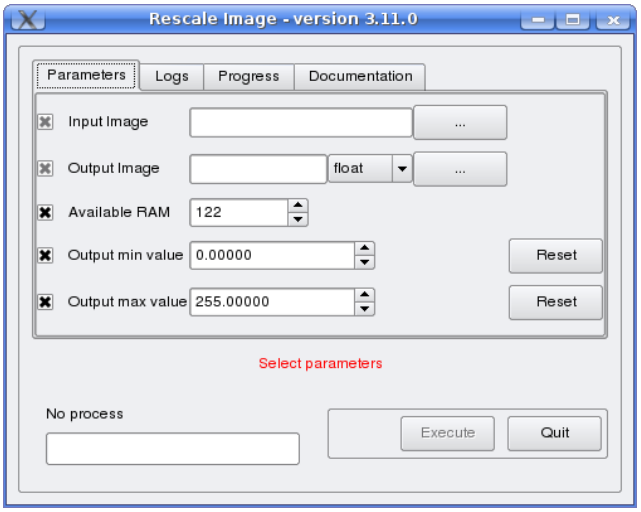


Figure 1.1: Parameters tab in *Rescale* application.

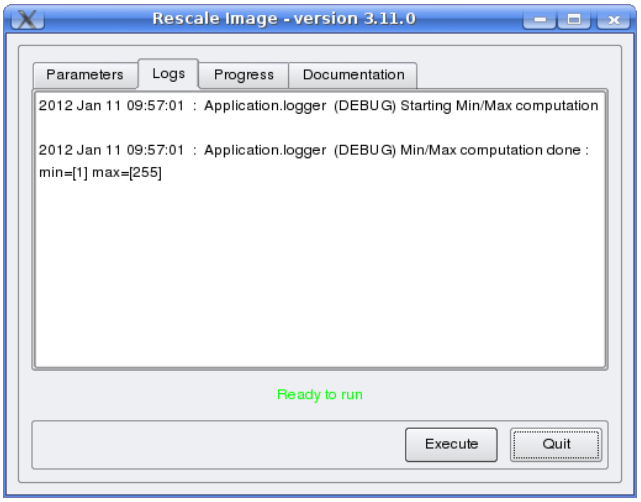


Figure 1.2: Logs tab in *Rescale* application.

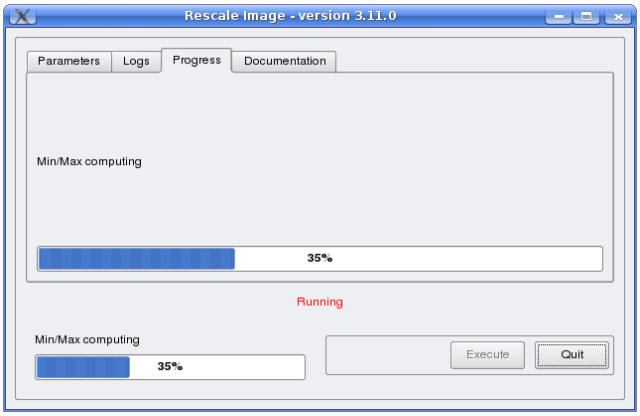


Figure 1.3: Progress tab in *Rescale* application.

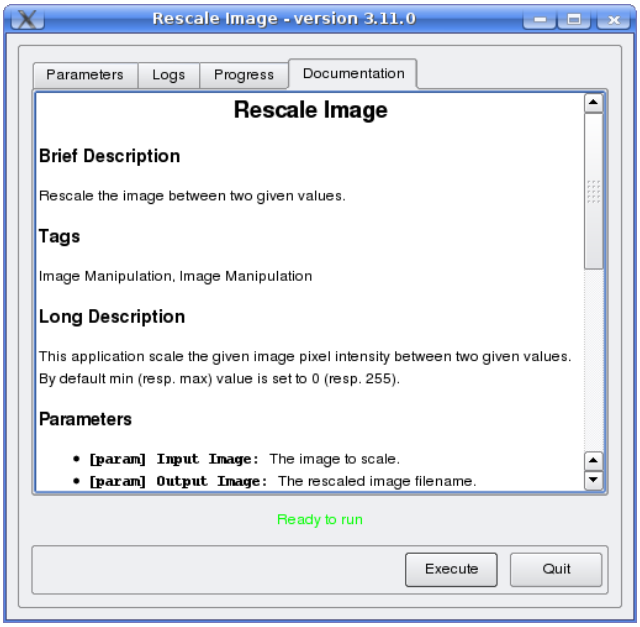


Figure 1.4: Parameters tab in *Rescale* application.

1.3.4 Using the Python interface

The applications can also be accessed from Python, through a module named `otbApplication`

On Unix systems it is typically available in the `/usr/lib/otb/python` directory. You may need to configure the environment variable `PYTHONPATH` to include this directory so that the module becomes available from an Python shell.

On Windows, you can install the `otb-python` package, and the module will be available from an OSGeo4W shell automatically.

In this module, two main classes can be manipulated :

- `Registry`, which provides access to the list of available applications, and can create applications
- `Application`, the base class for all applications. This allows to interact with an application instance created by the `Registry`

As for the command line and GUI launchers, the path to the application modules needs to be properly set with the `ITK_AUTOLOAD_PATH` environment variable. The standard location on Unix systems is `/usr/lib/otb/applications`. On Windows, the applications are available in the `otb-bin` OSGeo4W package, and the environment is configured automatically so you don't need to tweak `ITK_AUTOLOAD_PATH`.

Here is one example of how to use Python to run the `Smoothing` application, changing the algorithm at each iteration.

```
#  Example on the use of the Smoothing application
#

# We will use sys.argv to retrieve arguments from the command line.
# Here, the script will accept an image file as first argument,
# and the basename of the output files, without extension.
from sys import argv

# The python module providing access to OTB applications is otbApplication
import otbApplication

# otbApplication.Registry can tell you what application are available
print "Available applications : "
print str( otbApplication.Registry.GetAvailableApplications() )

# Let's create the application with codename "Smoothing"
app = otbApplication.Registry.CreateApplication("Smoothing")

# We print the keys of all its parameter
print app.GetParametersKeys()

# First, we set the input image filename
app.SetParameterString("in", argv[1])

# The smoothing algorithm can be set with the "type" parameter key
# and can take 3 values : 'mean', 'gaussian', 'anidif'
for type in ['mean', 'gaussian', 'anidif']:
```

```
print 'Running with ' + type + ' smoothing type'

# Here we configure the smoothing algorithm
app.SetParameterString("type", type)

# Set the output filename, using the algorithm to differenciate the outputs
app.SetParameterString("out", argv[2] + type + ".tif")

# This will execute the application and save the output file
app.ExecuteAndWriteOutput()
```

1.3.5 Load/Save OTB-Applications parameters from/to file

Since OTB 3.20, OTB applications parameters can be export/import to/from an XML file using inxml/outxml parameters. Those parameters are available in all applications.

An example is worth a thousand words

```
otbcli_BandMath -il input_image_1 input_image_2
                -exp "abs(im1b1 - im2b1)"
                -out output_image
                -outxml saved_applications_parameters.xml
```

Then, you can run the applications with the same parameters using the output xml file previously saved. For this, you have to use the inxml parameter:

```
otbcli_BandMath -inxml saved_applications_parameters.xml
```

Note that you can also overload parameters from command line at the same time

```
otbcli_BandMath -inxml saved_applications_parameters.xml
                -exp "(im1b1 - im2b1)"
```

In this cas it will use as mathematical expression ”(im1b1 - im2b1)” instead of ”abs(im1b1 - im2b1)”.

Finally, you can also launch applications directly from the command-line launcher executable using the inxml parameter without having to declare the application name. Use in this case:

```
otbApplicationLauncherCommandLine -inxml saved_applications_parameters.xml
```

It will retrieve the application name and related parameters from the input xml file and launch in this case the BandMath applications.

CHAPTER

TWO

A brief tour of Monteverdi

2.1 Introduction

The **OTB Applications** package makes available a set of simple software tools, which were designed to demonstrate what can be done with **Orfeo ToolBox**. Many users started using these applications for real processing tasks, so we tried to make them more generic, more robust and easy to use. **Orfeo ToolBox** users have been asking for an integrated application for a while, since using several applications for a complete processing (ortho-rectification, segmentation, classification, etc.) can be a burden. Recently, the OTB team received a request from CNES' Strategy and Programs Office in order to provide an integrated application for capacity building activities (teaching, simple image manipulation, etc.). The specifications included ease of integration of new processing modules.

2.2 Installation

The application is called **Monteverdi**, since this is the name of the Orfeo composer. The application allows you to build interactivelly remote sensing processes based on the **Orfeo ToolBox**. This is also in remembering of the great (and once open source) Khoros/Cantata software.

Installation of **Monteverdi** is very simple. Standard installer packages are available on the main platforms thanks to OTB-Developpers and external users. These packages are available few days after the release. Get the latest information on binary packages on the Orfeo ToolBox website in the section download.

We will discribe in the following sections the way to install monteverdi on:

- Windows platform (XP/Seven)
- Ubuntu 12.04 and higher
- OpenSuse 12.X and higher

- MacOSX 10.8

If you want build from source or if we don't provide packages for your system, some informations are available into the OTB Software Guide, in the section (Building from Source)

2.2.1 Windows XP/Seven/8.1

For Windows XP/Seven/8.1 users, there is a classical standalone installation program for Monteverdi, available from the OTB download page after each release.

Since version 1.12, it is also possible to get Monteverdi package through OSGeo4W for Windows XP/Seven users. Package for Monteverdi is available directly in the OSGeo4W installer when you select the **otb-monteverdi** package. Follow the instructions in the OSGeo4W installer and select the **otb-monteverdi**. The installer will proceed with the installation of the package and all its dependencies. Monteverdi will be directly installed in the OSGeo4W repository and a shortcut will be added to your desktop and in the start menu (in the OSGeo4W folder). You can now use directly Monteverdi from your desktop, from the start menu and from an OSGeo4W shell with command `monteverdi`. Currently, you should use the 32bit OSGeo4W installer but we will soon distribute monteverdi package for 64 bit installer.

2.2.2 MacOS X

A standard DMG package is available for Monteverdi for MacOS X 10.8. Please go the OTB download page. Click on the file to launch Monteverdi. This DMG file is also compatible with MacOSX 10.9.

2.2.3 Ubuntu 12.04 and higher

For Ubuntu 12.04 and higher, Monteverdi package may be available as Debian package through APT repositories.

Since release 1.14, Monteverdi packages are available in the ubuntugis-unstable repository.

You can add it by using these command-lines:

```
sudo aptitude install add-apt-repository
sudo apt-add-repository ppa:ubuntugis/ubuntugis-unstable
```

Now run:

```
sudo aptitude install monteverdi
```

If you are using *Synaptic*, you can add the repository, update and install the package through the graphical interface.

apt-add-repository will try to retrieve the GPG keys of the repositories to certify the origin of the packages. If you are behind a http proxy, this step won't work and apt-add-repository will stall and eventually quit. You can temporarily ignore this error and proceed with the update step. Following this, aptitude update will issue a warning about a signature problem. This warning won't prevent you from installing the packages.

2.2.4 OpenSuse 11.X and higher

For OpenSuse 12.X and higher, Monteverdi packages is available through *zypper*.

First, you need to add the appropriate repositories with these command-lines (please replace 11.4 by your OpenSuse version):

```
sudo zypper ar
http://download.opensuse.org/repositories/games/openSUSE_11.4/ Games
sudo zypper ar
http://download.opensuse.org/repositories/Application:/Geo/openSUSE_11.4/ GEO
sudo zypper ar
http://download.opensuse.org/repositories/home:/tzotsos/openSUSE_11.4/ tzotsos
```

Now run:

```
sudo zypper refresh
sudo zypper install Monteverdi
```

Alternatively you can use the One-Click Installer from the openSUSE Download page or add the above repositories and install through Yast Package Management.

There is also support for the recently introduced 'rolling' openSUSE distribution named 'Tumbleweed'. For Tumbleweed you need to add the following repositories with these command-lines:

```
sudo zypper ar
http://download.opensuse.org/repositories/games/openSUSE_Tumbleweed/ Games
sudo zypper ar
http://download.opensuse.org/repositories/Application:/Geo/openSUSE_Tumbleweed/ GEO
sudo zypper ar
http://download.opensuse.org/repositories/home:/tzotsos/openSUSE_Tumbleweed/ tzotsos
```

and then add the monteverdi packages as shown above.

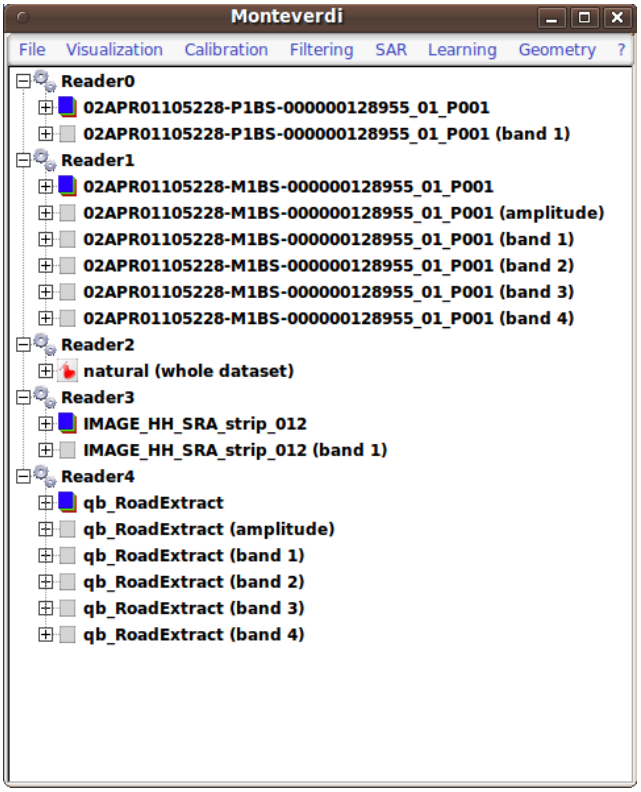


Figure 2.1: Monteverdi main window.

2.3 Anatomy of the applications

2.3.1 What does it look like?

This is Monteverdi's main window (figure 2.1) where the menus are available and where you can see the different modules, which have been set up for the processing. Input data are obtained by readers. When you choose to use a new module, you select its input data, and therefore, you build a processing pipeline sequentially. Figure 2.2 shows the generic window which allows to specify output(s) of Monteverdi's modules.

Let's have a look at the different menus. The first one is of course the "File" menu. This menu allows you to open a data set, to save it and to cache it. The "data set" concept is interesting, since you don't need to define by hand if you are looking for an image or a vector file. Of course, you don't need to do anything special for any particular file format. So opening a data set will create a "reader" which will appear in the main window. At any time, you can use the "save data set" option

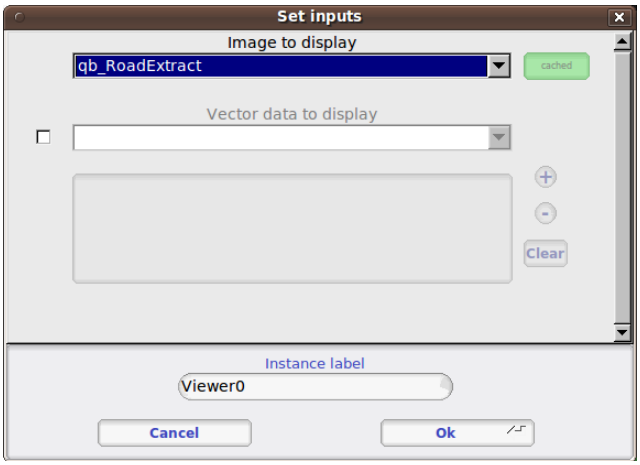


Figure 2.2: Monteverdi inputs selection generic window.

in order to store to a file the result of any processing module.

2.3.2 Open an image with **Monteverdi**

The application allows to interactively select raster/vector dataset by browsing your computer. Monteverdi takes advantage of the automatic detection of images' extensions to indicate the dataset type (optical, SAR or vector data).

The input dataset is added to the "Data and Process" tree, which describes the dataset content and each node corresponds to a layer.

2.3.3 Visualize an image with **Monteverdi**

This module allows to visualize raster or vector data. It allows to create RGB composition from the input rasters. It is also possible to add vector dataset which are automatically reprojected in the same projection of the input image or Digital Elevation informations.

The viewer offers three types of data visualisation:

- The Scroll window : to navigate quickly inside the entire scene
- The Full resolution window: the view of the region of interest selected in the scroll window
- The Zoom window

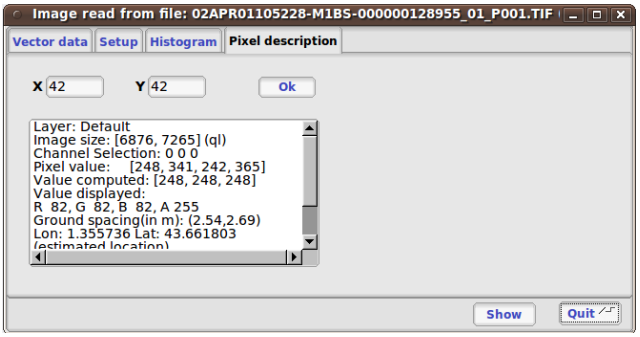


Figure 2.3: Monteverdi pixel description box.

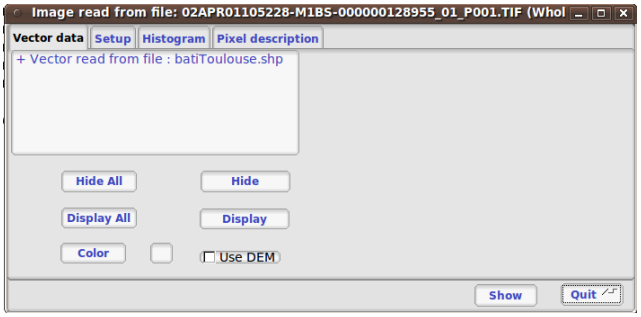


Figure 2.4: Monteverdi pixel description box.

- The Pixel description: give access to dynamic informations on the current pixel pointed. Informations display are:
  - The current index
  - The pixel value
  - The computed value (the dynamic of hte input image is modified to get a proper visualization
  - The coordinates of the current pixel (longitude and latitude)
  - In case where there is a Internet connection available, Monteverdi displays the estimate location of the current pixel (country + city)

The Visualization offers others great functionnalities which are available in the detached window. It is for example possible to superpose vector dataset to the input image (see figure 2.4).

The ”Setup Tab” allows to modify the RGB composition or use the grayscale mode to display only one layer.

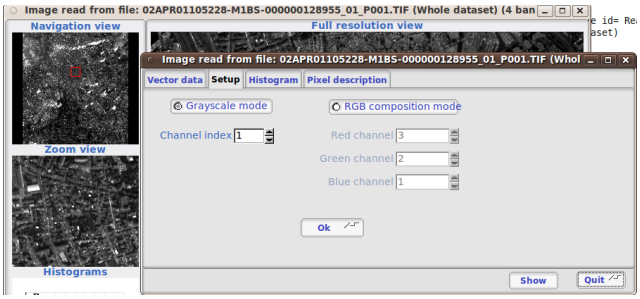


Figure 2.5: Manage RGB composition.

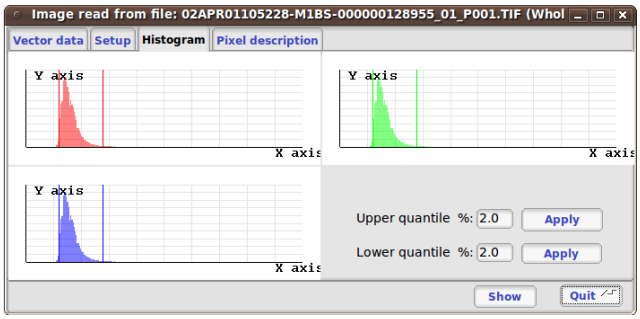


Figure 2.6: Manage the dynamic.

The ”Histogram Tab” get access to the dynamic of the displayed layers. The basic idea is to convert the output of the pixel representation to a RGB pixel for rendering on conventional displays. Values are constrained to 0-255 with a transfer function and a clamping operation. By default, the dynamic of each layer is modified by clamping the histogram at $min + 2\%$ and $max - 2\%$.

There is also possible to select pixel coordinates and get access to all the informations available in the “Pixel description Box”.

2.3.4 Cache dataset

The ”cache data set” (see figure 2.8) is a very interesting functionality. As you know, **Orfeo ToolBox** implements processing on demand, so when you build a processing pipeline, no processing takes place unless you ask for it explicitly. That means that you can plug together the opening of a data set, an orthorectification and a speckle filter, for example, but nothing will really be computed until you trigger the pipeline execution. This is very convenient, since you can quickly build a processing pipeline and let it execute afterwards while you have a coffee. In **Monteverdi**, the process is executed by saving the result of the last module of a pipeline. However, sometimes, you may want to execute a part of the pipeline without having to set the file name to the obtained result. You can do

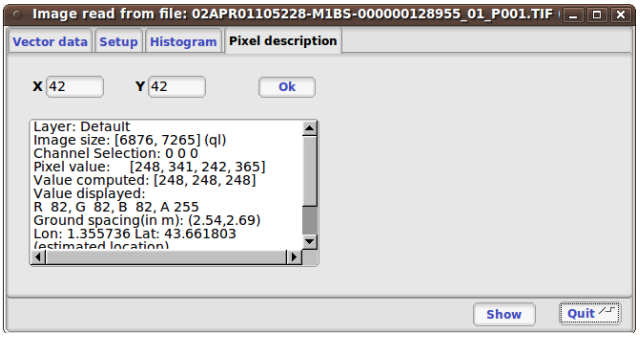


Figure 2.7: Index description via index selection.

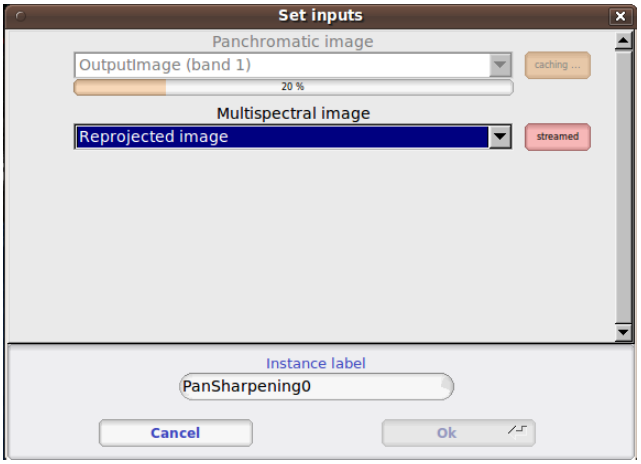


Figure 2.8: Cached and streamed dataset.

this by caching a data set. That is, the result will be stored in a temporary file which will be created in the ”Caching” directory created by the application. Another situation in which you may need to cache a data set is when you need that the input of a module exists when you set its parameters. This is nor a real requirement, since Monteverdi will generate the needed data by streaming it, but this can be inefficient. This for instance about visualization of the result of a complex processing. Using streaming for browsing through the result image means processing the visible part every time you move inside the image. Caching the data before visualization will generate the whole data set in advance allowing for a more swift display. All modules allow you to cache their input data sets.

2.3.5 Dynamic GUI definition

The aim of **Monteverdi** is to provide a generic interface which is based on the definition of the internal processes. In this frame, the way that you have to manage modules are identical during the definition of a new process. Selecting a module on the upper main window, open automatically the ”Inputs definition Window” wich allows to select data which are inputs of the current module. **Monteverdi** module can manage single or multiple inputs and these inputs can be images on your computer or results of previous module already registered in the ”Data and Process” tree.

2.3.6 Dynamic I/O definition

Management of image formats in **Monteverdi** works in the same manner as in the **Orfeo ToolBox**. The principle is that the software automatically recognize the image format. Communication between modules follow also the same principle and the Input definition of modules request to all available outputs of the same type in the ”Data and process” tree. Internally, all the treatments in **Monteverdi** are computed in float precision by default. It is also possible to switch to double precision by compiling the application from source and set the CMAKE option compile float to ON.

2.4 Available modules

2.4.1 I/O operations

Extract region of interest

It allows to extract regions of interest (ROI) from an image. There are two ways to select the region:

- By indicating the X and Y coordinatres of the upper-left coordinates and the X-Y size of the regions.
- By interactivelly selecting the region of interest in the input image.

Concatenate image bands

With **Monteverdi**, you could generate a large scale of value added informations from lots of inputs data. One of the basic functionnality is to be able to superpose result’s layers into the same dataset. Concatenating images into one single multiple-bands image (they need to have the same size), and to be able to create for example RGB composition with the inputs layer.

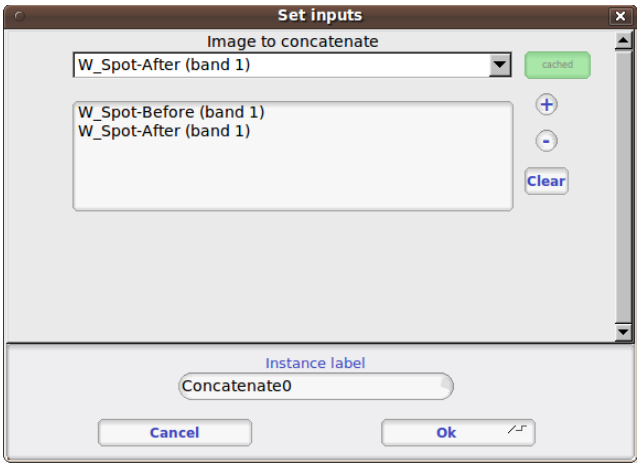


Figure 2.9: Concatenation module.

Save dataset to file

Monteverdi allows to export raster or vector dataset to a file to your system. In the case of raster images, it is possible to cast output pixel type. In **Monteverdi** all the processes are done in floating point precision. On large remote sensing dataset, saving your result in float data type could lead to file too large(more than 25 Go for pan-sharpened 8 bands WorldView2 with a resolution of 46 centimeters). Since the module allows to cast pixels in other types :

- unsigned char (8 bits)
- short (16 bits)
- int (32 bits)
- float (32 bits)
- double (64 bits)
- unsigned short (16 bits)
- unsigned int (32 bits)

2.4.2 Geometric process

In the frame of remote sensing process, one common operation is to be able to superpose and manipulate data which come from different sources. This section gives access to a large set of geometric

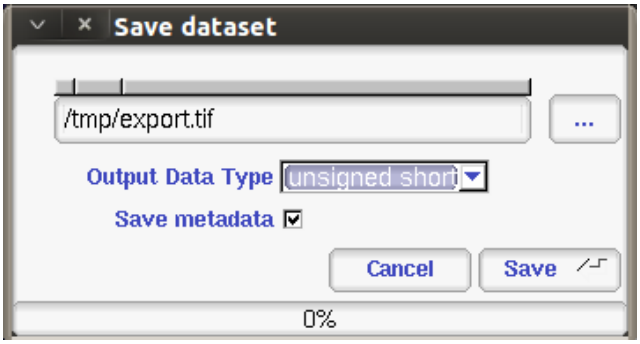


Figure 2.10: Save dataset module.

operations. It performs re-projection and orthorectification operations on Optical or SAR dataset using the available sensor models (image informations available in the meta-data are automatically read by the application).

Reprojection module

The application is derived from the otbOrthorectificationApplication in the **OTB Applications** package and allow to produce orthorectified imagery from level 1 product. The application is able to parse metadata informations and set default parameters. The application contains 4 tabs:

- Coordinates: Define the center or upper-left pixel coordinates of the orthorectified image (the longitude and latitude coordinates are calculated through meta-data informations. It is also possible to specify the map projection of the output.
- Output image: The module allows to only orthorectified a Region Of interest inside the input dataset. This tab allows to set the size of the ROI around the center pixel coordinate or from the upper left index. The orthorectified imagery can also be resampled at any resolution in the line or column directions by setting the ”Spacing X” and the ”Spacing Y” respectively, and choosing interpolation method.
- DEM: Indicate path to a directory containing SRTM elevation file. The application is able to detect inside the direcory which DEM files are relevant in the process. You can find detailed informations on how to get a usable DEM
- Image extent: Compare the initial image extension with the preview the orthorectified result. This preview is automatically updated if the user change the ”Size X” or ”Size Y” values in the ”Output Image” tab.

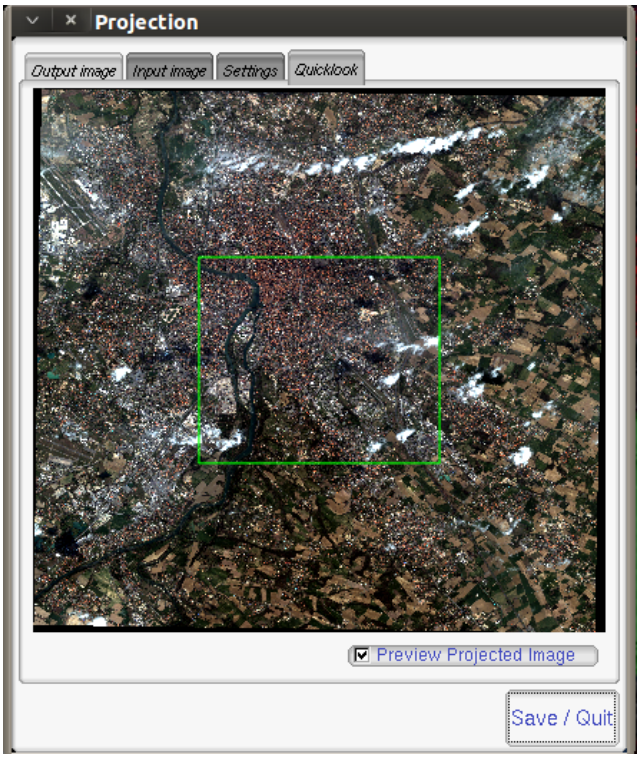


Figure 2.11: Reprojection module. Quick Look tab.

Estimating sensor model based on ground control points

This module allows to take ground control points on a raster image where no geographic informations are available. This GCPs list is making correspondence between pixel coordinate in the input image and physical coordinates. The list allows to derive a general function which convert any pixel coordinates in physical positions. This function is based on a RPC transformation (Rational Polynomial Coefficients). As a consequence, the module enriches the output image with metadata informations defining a RPC sensor model associated with the input raster. There are several ways to generate the GCPs:

- With Internet access: dynamically generate the correspondance on the input image and Open Street Map layers.
- Without Internet access: Set manually Ground control points : indicate index position and cartographic coordinates in the input image.

It is also possible to import/export the list of Ground Control points from/to an XML file.

Moreover, if the input image has GCPs in its metadata, the module allows to add or remove points from the existing list, which is automatically loaded.

2.4.3 Calibration

In the solar spectrum, sensors on Earth remote sensing satellites measure the radiance reflected by the atmosphere-Earth surface system illuminated by the sun. This signal depends on the surface reflectance, but it is also perturbed by two atmospheric processes, the gaseous absorption and the scattering by molecules and aerosols.

Optical calibration

In the case of the Optical calibration, the basic idea is to be able to retrieve reflectance of the observed physical objects. The process can be split in 3 main steps:

- Derived luminance from the raw value in the input image.
- Convert the luminance to reflectance to produce the TOA image(Top Of Atmosphere).
- Inverse a radiative transfer code, which simulates the reflection of solar radiation by a coupled atmosphere-surface system. This step produce the TOC (Top of Canopy) imagery, which is the final result of the optical calibration module.

SAR calibration

The calibration and validation of the measurement systems are important to maintain the reliability and reproducibility of the SAR measurements, but the establishment of correspondence between quantities measured by SAR and physical measure requires scientific background. The SAR calibration module allows to estimate quantitative accuracy. For now only calibration of TerraSARX data is available.

2.4.4 Filtering Operations

Band Math

The Band Math module allows to perform complex mathematical operations over images. It is based on the mathematical parser library muParser and comes with a bunch of build-in functions and operators (listed here). This home-brewed digital calculator is also bundled with custom functions allowing to compute a full expression result simply and really quickly, since the filter supports streaming and multi-threading. The **Monteverdi** module provides an intuitive way to easily perform complex band computation. The module also prevents error in the mathematical command by checking the expression as the user types it, and notifying information on the detected error:

Figure 2.12 presents an example on how the band math can produce a threshold image on the NDVI value computed in one pass using built-in conditional operator "if" available in the parser.

An other operational example, on how this simple module can produce reliable information. Figure 2.13 shows the result of the subtraction of the Water indice on 2 images which was taken before and during the crisis event. The difference was produced by the band math module and allows to get a reliable estimation of the flood events.

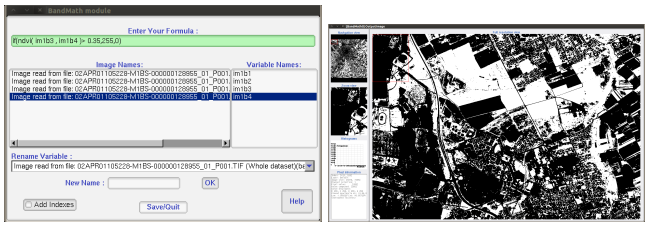


Figure 2.12: Conditional operators using the band math module (on the left) to process a NDVI image threshold and the resulting image (on the right).

Connected Component Segmentation module

The Connected Component Segmentation module allows segmentation and object analysis using user defined criteria at each step. This module uses muParser library using the the same scheme as

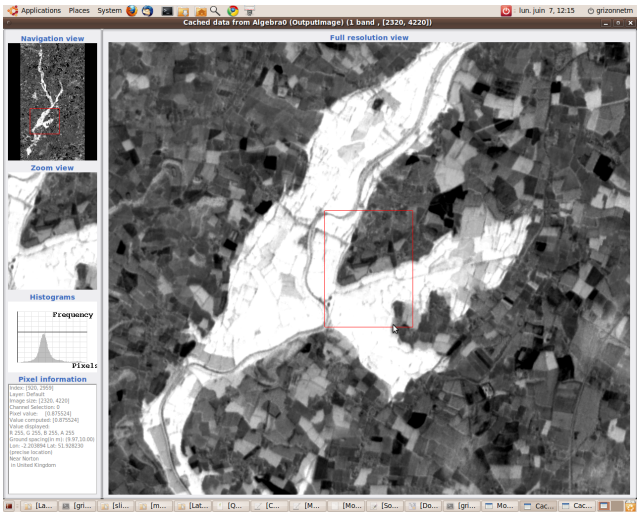


Figure 2.13: Subtraction of NDWI2.

it is done in Band math module (see 2.4.4 for a detailled explanation). It relies on three main steps process :

Mask definition : This mask is used as support of Connected Component segmentation (CC) . i.e zeros pixels are not taken into account by CC algorithm. Binarization criteria is defined by user, via muparser. This step is optional, if no Mask is given, entire image is processed. The following example creates a mask using *intensity* (mean of pixel values) parameter :

```
intensity > 200
```

Segmentation : Connected Component Segmentation driven by user defined criteria. Segmentation process can be followed by a small object rejection step. The following example use *distance* (pixel intensity value difference) parameter to define acception/rejection criteria between two adjacents pixels :

```
distance < 10
```

Object analysis post processing : This step consists in post processing on each detected area using shape and statistical object characterization. The following example use *elongation* parameter to test labeled objects :

```
SHAPE_elongation > 2
```

A detailled presentation of parameters and variables, can be found on the wiki.

Results are then exported in shape file format. Graphical user interface is presented on Figure 2.14. At each step intemerdiate output can be seen using *Display* item list. Viewing windows are updated by clicking on *Update* button. Available display outputs are :

Input image : input image.

Mask Output : mask image created using formula.

Masked Image : input image multiplied by mask image.

Segmentation Output : output of Connected Component segmentation filter.

Segmentation after small object rejection : output of Connected Component segmentation after relabeling and small object rejection.

Filter Output : final output after object based analysis opening post processing.

Available variables for each expression can be found using item list *variables names*. available functions can be found in help windows by clicking on *Help* button. The module also prevents error in the mathematical command by checking the expression as the user types it. Background value is set to green if formula is right, in red otherwise. If mask expression is left blank entire image is processed. If *Object Analysis* expression is left blank the whole set of label objects is considered.

After segmentation step, too small objects can be rejected using *Object min area* input. Eliminating too small objects at this step is needed to lighten further computation. min area is the pixel size of the label object.

When a first pass have been done, Specific label object properties can be displayed. Select the ”Filter Output” visualization mode, Update the visualization. Then use right click on selected object in image to display object properties.

Clicking on *Save and Quit* button export output to Monteverdi in vector data format.

A detailled presentation of this module, and examples can be found on the wiki.

A boat detection example is presented on Figure 2.15. Results can be seen on Figure 2.16.

Feature extraction

Under the term Feature Extraction, it include several techniques aiming to detect or extract informations of low level of abstraction from images. These features can be objects : points, lines, etc.They can also be measures : moments, textures, etc.

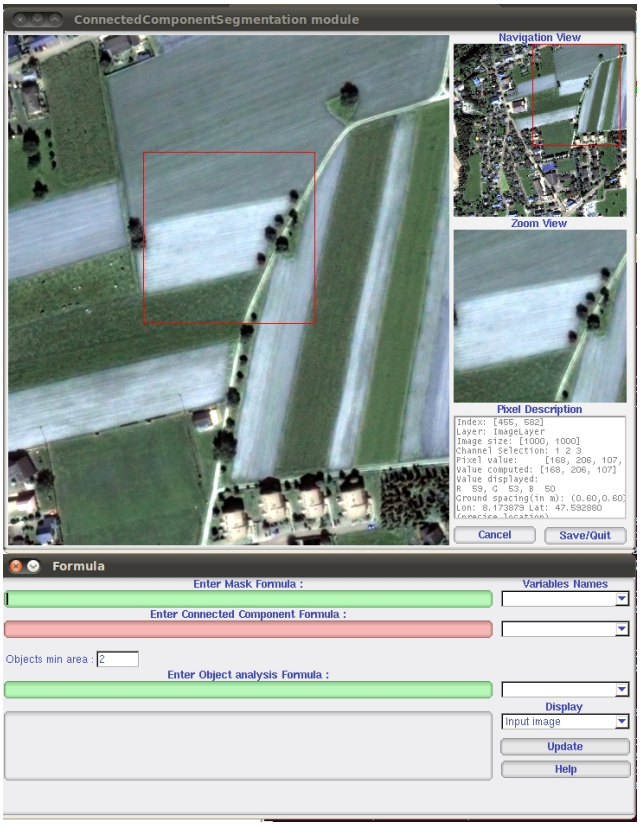


Figure 2.14: connected component segmentation module.

Figure 2.15: Connected Component Segmentation module example : boat detection.

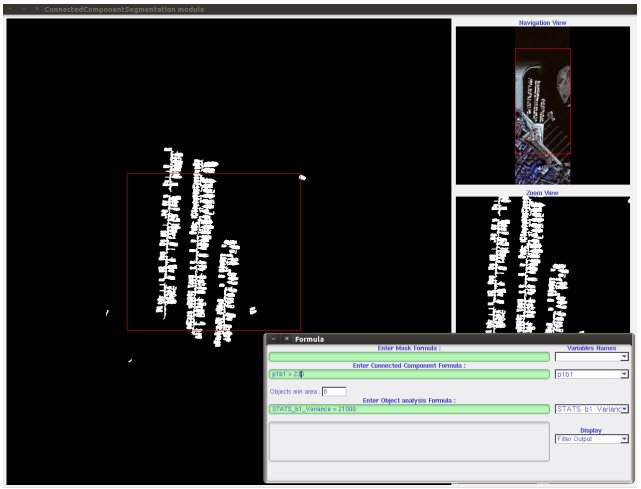


Figure 2.16: Boat detection example results.

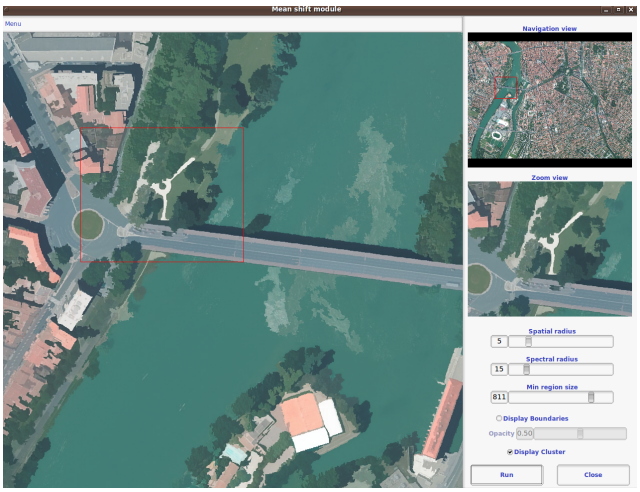


Figure 2.17: Mean-shift module.

Mean-shift segmentation

For a given pixel, the Mean-shift algorithm will build a set of neighboring pixels within a given spatial radius and a color range. The spatial and color center of this set is then computed and the algorithm iterates with this new spatial and color center. The Mean-shift can be used for edge-preserving smoothing, or for clustering.

2.4.5 Learning

Supervised classification

Supervised classification is a procedure in which individual items are placed into groups based on quantitative information on one or more characteristics inherent in the items and based on a training set of previously labeled items.

The supervised classification module is based on the Support Vector Machine method which consists in searching for the separating surface between 2 classes by the determination of the subset of training samples which best describes the boundary between the 2 classes. This method can be extended to be able to classify more than 2 classes.

The module allows to interactivelly describe learnings samples which corresponds to polygons samples on the input images.

Then a SVM model is derived from this learning sample which allows to classify each pixel of the input image in one of the defined class.

Non-supervised classification

The non supervised classification module is based on the Kmeans algorithm. The GUI allows to modify parameters of the algorithm and produce a label image.

2.4.6 Specific SAR functionnalities

This section give access to specific treatments related to the SAR (Synthetic Aperture Radar) functionnalities.

Despeckle

SAR images are generally corrupted by speckle noise. To suppress speckle and improve the radar image interpretability lots of filtering techniques have been proposed. The module implements to well-known despeckle methods: Frost and Lee.

Compute intensity and log-intensity

Compute the derived intensity and log-intensity from the input SAR imagery.

Polarimetry

In conventional imaging radar the measurement is a scalar which is proportional to the received backscattered power at a particular combination of linear polarization (HH, HV, VH or VV). Polarimetry is the measurement and interpretation of the polarization of this measurement which allows to measure various optical properties of a material. In polarimetry the basic measurement is a $2x2$ complex scattering matrix yielding an eight dimensional measurement space (Sinclair matrix). For reciprocal targets where $HV = VH$, this space is compressed to five dimensions: three amplitudes ($|HH|$, $|HV|$, and $|VV|$); and two phase measurements, (co-pol: HH-VV, and cross-pol: HH-HV). (see grss-ieee).

Synthesis Allow to construct an image that would be received from a polarimetric radar having selected transmit and receive polarizations. The Synthesis module waits for real and imaginary part (real images) of the HH, VV, VH and HV images. The reciprocal case where case $VH = HV$, is not properly handled yet, for now the user has to set the same input for the two HV and VH.

Conversion As we saw in the previous main section, the basic measurement is a $2x2$ complex scattering matrix yielding an eight dimensional measurement space. But other measurements exist:

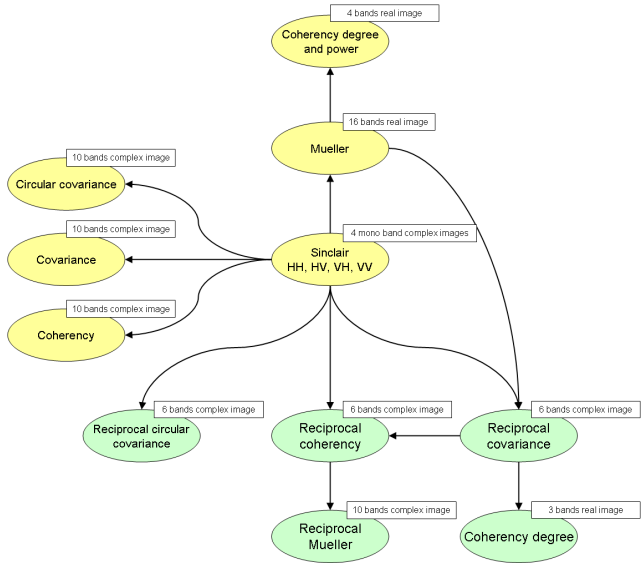


Figure 2.18: SAR polarimetry conversion module.

- covariance matrix and with its reciprocal specific case
- coherency matrix and with its reciprocal specific case
- circular coherency matrix and with its reciprocal specific case
- Mueller matrix and with its reciprocal specific case...

Modules in the Conversion subsection allow to proceed these conversions between matrix representations. Allowed conversion and input images types are described in the following figure 2.18.

Analysis This module allows to perform some of classical polarimetric analysis methods. It allows to compute:

- The polarimetric synthesis:
  - input: 4 bands complex image
  - output: mono channel real image
  - parameters: the synthesis parameters (incident and reflected ψ and χ angles)
- The reciprocal H alpha image:
  - input: 6 bands complex image
  - output: 3 bands real image

CHAPTER

THREE

A brief tour of Monteverdi2

3.1 Introduction

Monteverdi was developed 4 years ago in order to provide an integrated application for capacity building activities (teaching, simple image manipulation, etc.). Its success went far beyond this initial scope since it opened the OTB world to a wide range of users who needed a ready to use graphical tool more than a library of components to write their own processing chain. With this 4 years of lifetime, we have a lot of feedbacks regarding how useful the tool was, but also regarding what should be improved to move toward greater usability and operationnality. We therefore decided to rework the Monteverdi concept into a brand new software, enlightened by this experience.

Monteverdi2 offers a new interface based on Qt and a set of processing capibilities base on OTB-Applications.

3.2 Installation

Installation of **Monteverdi2** is very simple. Standard installer packages are available on the main platforms thanks to OTB-Developpers and external users. These packages are available few days after the release. Get the latest information on binary packages on the Orfeo ToolBox website in the section download.

We will discribe in the following sections the way to install **Monteverdi2** on:

- Windows platform (XP/Seven/8.1)
- Ubuntu 12.04 and higher
- MacOSX 10.8

If you want build from source or if we don't provide packages for your system, some informations are available into the OTB Software Guide, in the section (Building from Source)

3.2.1 Windows XP/Seven/8.1

For Windows XP/Seven/8.1 users, there is a classical standalone installation program for **Monteverdi2**, available from the OTB download page after each release of **Monteverdi2**.

3.2.2 MacOS X

A standard DMG package is available for **Monteverdi2** for MacOS X 10.8. Please go the OTB download page. Click on the file to launch **Monteverdi2**. We will provide in the next release a package for MacOSX 10.9.

3.2.3 Ubuntu 12.04 and higher

For Ubuntu 12.04 and higher, **Monteverdi2** package may be available as Debian package through APT repositories.

Since release 0.2, **Monteverdi2** packages are available in the ubuntugis-unstable repository.

You can add it by using these command-lines:

```
sudo aptitude install add-apt-repository
sudo apt-add-repository ppa:ubuntugis/ubuntugis-unstable
```

Now run:

```
sudo aptitude install monteverdi2
```

If you are using *Synaptic*, you can add the repository, update and install the package through the graphical interface.

apt-add-repository will try to retrieve the GPG keys of the repositories to certify the origin of the packages. If you are behind a http proxy, this step won't work and apt-add-repository will stall and eventually quit. You can temporarily ignore this error and proceed with the update step. Following this, aptitude update will issue a warning about a signature problem. This warning won't prevent you from installing the packages.

3.3 What does it look like?

You can find more information about **Monteverdi2** into the post of the OTB blog at `http://blog.orfeo-toolbox.org/`.

CHAPTER

FOUR

Recipes

This chapter presents guideline to perform various remote sensing and image processing tasks with either **OTB Applications**, **Monteverdi** or both. Its goal is not to be exhaustive, but rather to help the non-developper user to get familiar with these two packages, so that he can use and explore them for his future needs.

4.1 Using Pleiades images in **OTB Applications** and **Monteverdi**

The typical Pleiades product is a pansharpened image of 40 000 by 40 000 pixels large, with 4 spectral bands, but one can even order larger mosaics, whose size can be even larger, with hundreds of thousands of pixels in each dimension.

To allow easier storage and transfer of such products, the standard image file format is Jpeg2000, which allows to achieve high compression rates. The counterpart of these better storage and transfer performances is that the performance of pixels accesses within those images may be poorer than with an image format without compression, and even more important, the cost of accessing pixels is not uniform: it depends on where are the pixels you are trying to access, and how they are spatially arranged.

To be more specific, Pleiades images are internally encoded into 2048 per 2048 pixels tiles (within the Jpeg2000 file). These tiles represent the atomic decompression unit: if you need a single pixel from a given tile, you still have to decode the whole tile to get it. As a result, if you plan to access a large amount of pixels within the image, you should try to access them on a per tile basis, because anytime you ask for a given tile more than once, the performances of your processing chains drop.

What does it mean? In **Orfeo ToolBox**, the streaming (on the flow) pipeline execution will try to stay synchronised with the input image tiling scheme to avoid decoding the same tile several time. But you may know that in the **Orfeo ToolBox** world, one can easily chain numerous processing, some them enlarging the requested region to process the output - like neighbourhood based operators for instance - or even completely change the image geometry - like ortho-rectification for instance. And this chaining freedom is also at the heart of **Monteverdi**. In short, it is very easy to build a processing pipeline in **Orfeo ToolBox** or chain of modules in **Monteverdi** that will get incredibly bad performances, even if the **Orfeo ToolBox** back-end does its best to stay in tune with tiles. And here, we do not even speak of sub-sampling the whole dataset at some point in the pipeline, which will lead to even more incredibly poor performances, and is however done anytime a viewer is called on a module output in **Monteverdi**.

So, can **Monteverdi** or **OTB Applications** open and process Pleiades images? Fortunately yes. **Monteverdi** even takes advantage of Jpeg2000 ability to generate coarser scale images for quick-look generation for visualisation purposes. But to ease the use of Pleiades images in **Monteverdi**, we chose to open them in a separate data type, and to lock the use of most of modules for this data type. It can only be used in the Viewer module and a dedicated module allowing to uncompress a user-defined part of a Pleiades image to disk. One can still force the data type during the opening of the image, but this is not advised: the advised way to use the other modules with Pleiades data is to first uncompress to disk your area of interest, and then open it again in **Monteverdi** (careful, you may need a lot of disk space to do this). As for the applications, they will work fine even on Jpeg2000 Pleiades data, but keep in mind that a performance sink might show depending on the processing you are try to achieve. Again, the advised way of working would be to uncompress your area of interest first and then work with the uncompressed file, as you used to with other data.

A final word about metadata: **OTB Applications** and **Monteverdi** can read the Dimap V2 (note that we also read the less non-official Dimap V1.1 format) metadata file associated with the Jpeg2000

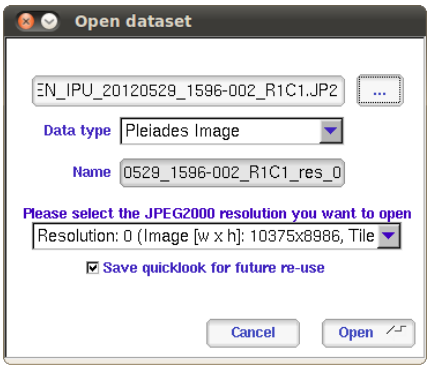


Figure 4.1: Dialog window when opening a Pleiades image in **Monteverdi**

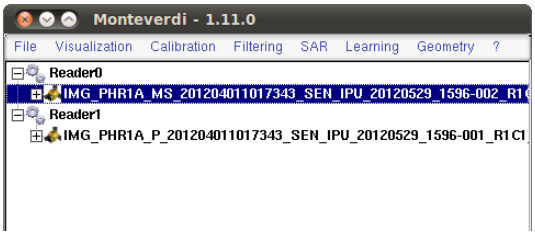


Figure 4.2: Pleiades images in the main **Monteverdi** window

file in the Pleiades product. It reads the RPC localisation model for geo-coding and the information needed to perform radiometric calibration. These metadata will be written in an associated geometry file (with a *.geom* extension) when uncompressing your area of interest to disk, so that both **Monteverdi** and **OTB Applications** will be able to retrieve them, even for images extracts.

4.1.1 Opening a Pleiades image in **Monteverdi**

Opening a Pleiades image in **Monteverdi** is not different from opening other kind of dataset: use the *Open Dataset* item from the *File* menu, and select the JP2 file corresponding to you image using the file browser.

Figure 4.1, page 39 shows the dialog box when opening a Pleiades image in **Monteverdi**. One can see some changes with respect to the classical dialog box for images opening.

The first novelty is a combo box allowing to choose the resolution of the Jpeg2000 file one wants to decode. As said in the introduction of this section, **Orfeo ToolBox** can take advantage of Jpeg2000 capability to access coarser resolution ver efficiently. If you select for instance the *Resolution: 1* item, you will end with an image half the size of the original image with pixels twice as big. For

instance, on a Pleiades panchromatic or pansharpened product, the *Resolution: 0* image has a ground samping distance of 0.5 meters while the *Resolution: 1* image has a ground samping distance of one meter. For a multispectral product, the *Resolution: 0* image has a ground samping distance of 2 meters while the *Resolution: 1* image has a ground samping distance of 4 meters.

The second novelty is a check-box called *Save quicklook for future re-use*. This option allows to speed-up the loading of a Pleiades image within **Monteverdi**. In fact, when loading a Pleiades image, **Monteverdi** generates a quicklook of this image to be used as a minimap in the *Viewer Module* as well as in the *Uncompress Jpeg2000 image* module. This quicklook is the coarser level of resolution from the Jpeg2000 file: it should decode easily, but can still take a while. This is why if the check-box is checked, **Monteverdi** will write this quicklook in uncompressed *Tiff* format next to the Jpeg2000 file. For instance, if the file name is:

```
IMG_PHR1A_MS_201204011017343_SEN_IPU_20120529_1596-002_R1C1.JP2
```

Monteverdi will write, if it can, the following files in the same directory:

```
IMG_PHR1A_MS_201204011017343_SEN_IPU_20120529_1596-002_R1C1.JP2_ql_by_otb.tif
IMG_PHR1A_MS_201204011017343_SEN_IPU_20120529_1596-002_R1C1.JP2_ql_by_otb.geom
```

Next time one will try to open this image in **Monteverdi**, the application will find these files and load directly the quicklook from them, instead of decoding it from the Jpeg2000 file, resulting in an instant loading of the image in **Monteverdi**. Since the wheight of these extra files is ususally of a few megaoctets, it is recommended to keep this option checked unless one has a very good reason not to. Now that the Pleiades image is loaded in **Monteverdi**, it appears in the main **Monteverdi** window, as shown in figure 4.2, page 39.

4.1.2 Viewing a Pleiades image in **Monteverdi**

You can open the Pleiades image in the viewer, either by using the contextual menu or by opening the *Viewer Module* through the menu bar.

You can notice that the viewer opens quickly without showing the traditional progress bar. This is because **Monteverdi** already loaded the quick-look upon opening, and we do not need to re-compute it each time the image is opened in the *Viewer Module*.

Figure 4.3, page 41 shows a Pleiades image displayed in the *Viewer Module*. One can notice that the navigation experience is rather smooth. If you navigate using arrows keys, you will notice that latency can occur now and then: this is due to the viewport switching to a new Jpeg2000 tile to decode. On can also observe that the latitude and longitude of the pixel under the mouse pointer is displayed, which means that the sensor modelling is handled (if you have an internet connection, you may even see the actual name of the place under mouse pointer). Last, as said in the foreword of this section, Pleiades image can be quite large, so it might be convenient to switch the viewer style from *Packed* to *Splitted*, in which case you will be able to maximize the *Scroll Window* for better localisation of the viewed area. To do so, one can go to the *Setup* tab of the *Viewer Control Window*.

Figure 4.3: A Pleiades image displayed in **Monteverdi** viewer. ©CNES 2012

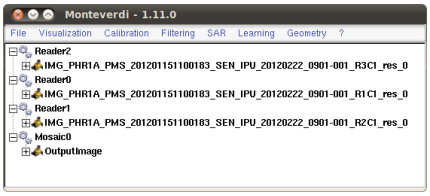


Figure 4.4: Pleiades mega-tiles and output mosaic in **Monteverdi**

4.1.3 Handling mega-tiles in **Monteverdi**

If the Pleiades product is very large, it might happen that the image is actually splitted into several Jpeg2000 files, also called mega-tiles. Since the area of interest might span two or more mega-tiles, it is convenient to stitch together these tiles so as to get the entire scene into one **Monteverdi** dataset. To do so, one must first open all mega-tiles in **Monteverdi**, as described in section 4.1.1, page 39. Once all mega-tiles are opened as shown in figure 4.4, page 41.

Once this is done, one can use the *Mosaic Images module* from the *File* menu. Simply append all mega-tiles into the module and run it: the module will look for the *RiCj* pattern to determine the mega-tiles layout, and will also check for consistency, e.g. missing tiles or mega-tiles size mismatch. Upon success, it generates a new Pleiades image dataset, which corresponding to the entire scene, as shown in figure 4.4, page 41. One can then use this dataset as a regular Pleiades dataset.

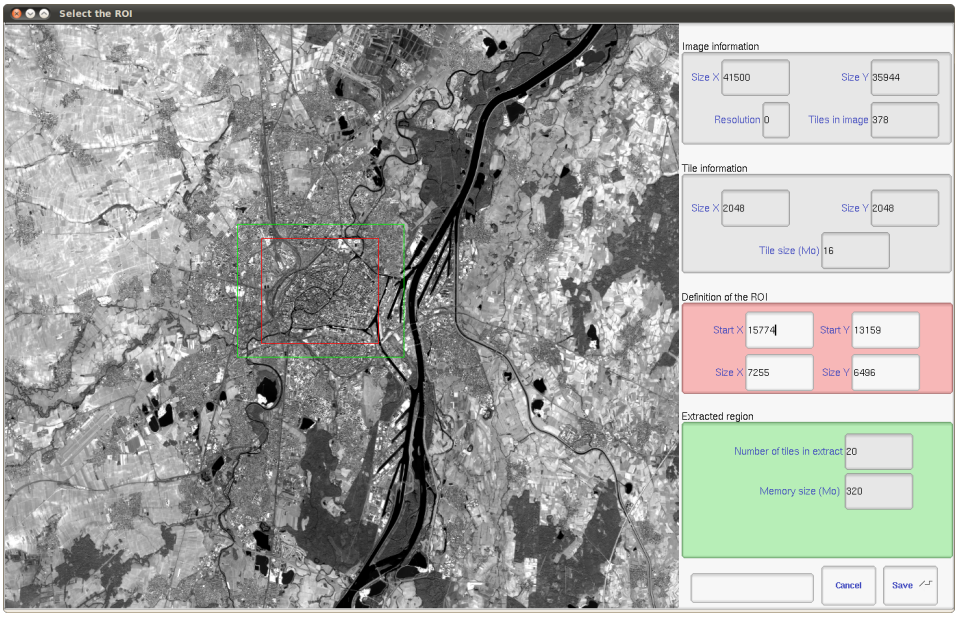


Figure 4.5: A Pleiades image in **Monteverdi** *Uncompress Jpeg2000 image module*. ©CNES 2012

4.1.4 Partial uncompressing of Pleiades images in **Monteverdi**

The next very important thing one can do with **Monteverdi** is to select an area of interest in the Pleiades image so as to uncompress it to disk. To do so, open the Pleiades dataset into the *Uncompress Jpeg2000 image module* from the *File* menu. Figure 4.5, page 42 shows what this module looks like. On the left, one can find informations about the images: dimensions, resolution level, and number of Jpeg2000 tiles in image, dimension of tiles, and size of tiles in mega-octets. The center part of the module is the most important one: it displays a quick-look of the Pleiades image. On this quick-look, one can select the area to be decoded by drawing a rectangle with the mouse. The red rectangle shown by the module corresponds to this user-defined area. On the left, in red, one can find the start index and size of corresponding region.

The module also displays a green rectangle, which shows the minimum set of tiles to be decoded to decode the red area: **this is the region that will actually be decoded to disk**. On the left, in green, one can find information about this region: how many tiles it contains, and what will be the size of the corresponding decoded output file.

Once one chose her area of interest, one can click on the *Save* button, and select an output file. The module will write a geometry file (with the *.geom* extension) with all useful metadata in it, so that when reading back the file in **Monteverdi** or in **OTB Applications**, geometry and radiometry based functionalities can still be used.

4.1.5 Other processing of Pleiades images with **Monteverdi**

For all the reasons exposed in the foreword of this section, we do not allow to use directly Pleiades images in the remaining of **Monteverdi** modules: the advised way of doing so is to first uncompress the area of interest to disk.

4.1.6 Processing of Pleiades images with **OTB Applications**

The **OTB Applications** are able to work directly with Pleiades images. However, keep in mind that performances may be limited due to the reasons exposed in the foreword of this section. If you experiment poor performances with some application, try to uncompress the area of interest from your image with **Monteverdi** first. One can also use the *ExtractROI* application for this purpose.

One thing that is interesting to know is that one can access the coarser resolution of the Jpeg2000 file by appending : *i* to the filename, where *i* is the resolution level starting at 0. For instance, one can use the following:

```
otbcli_ExtractROI -in IMG_PHR1A_PMS_201201151100183_SEN_IPU_20120222_0901-001_R2C1.JP2:5 -out test.tif uint16
```

4.2 Optical pre-processing

This section presents various pre-processing tasks that are presented in a classical order to obtain a calibrated, pan-sharpened image.

4.2.1 Optical radiometric calibration

In remote sensing imagery, pixel values are called DN (for Digital Numbers) and can not be physically interpreted and compared: they are influenced by various factors such as the amount of light flowing trough the sensor, the gain of the detectors and the analogic to numeric converter.

Depending on the season, the light and atmospheric conditions, the position of the sun or the sensor internal parameters, these DN can drastically change for a given pixel (apart from any ground change effects). Moreover, these effects are not uniform over the spectrum: for instance aerosol amount and type has usually more impact on the blue channel.

Therefore, it is necessary to calibrate the pixel values before any physical interpretation is made out of them. In particular, this processing is mandatory before any comparison of pixel spectrum between several images (from the same sensor), and to train a classifier without dependence to the atmospheric conditions at the acquisition time.

Calibrated values are called surface reflectivity, which is a ratio denoting the fraction of light that is reflected by the underlying surface in the given spectral range. As such, its values lie in the range

$[0,1]$. For convenience, images are often stored in thousandth of reflectivity, so that they can be encoded with an integer type. Two levels of calibration are usually distinguished:

- The first level is called *Top Of Atmosphere (TOA)* reflectivity. It takes into account the sensor gain, sensor spectral response and the solar illumination.
- The second level is called *Top Of Canopy (TOC)* reflectivity. In addition to sensor gain and solar illumination, it takes into account the optical thickness of the atmosphere, the atmospheric pressure, the water vapor amount, the ozone amount, as well as the composition and amount of aerosol gasses.

This transformation can be done either with **OTB Applications** or with **Monteverdi**. Sensor-related parameters such as gain, date, spectral sensitivity and sensor position are seamlessly read from the image metadata. Atmospheric parameters can be tuned by the user. Supported sensors are :

- Pleiades,
- SPOT5,
- QuickBird,
- Ikonos,
- WorldView-1,
- WorldView-2,
- Formosat.

Optical calibration with **OTB Applications**

The *OpticalCalibration* application allows to perform optical calibration. The mandatory parameters are the input and output images. All other parameters are optional. By default the level of calibration is set to TOA (Top Of Atmosphere). The output images are expressed in thousandth of reflectivity using a 16 bits unsigned integer type.

A basic TOA calibration task can be performed with the following command :

```
otbcli_OpticalCalibration -in  input_image -out output_image
```

A basic TOC calibration task can be performed with the following command :

```
otbcli_OpticalCalibration -in  input_image -out output_image -level toc
```

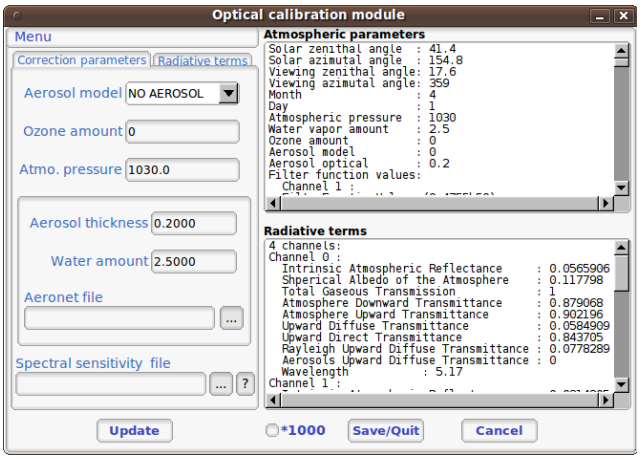


Figure 4.6: Optical calibration module.

Optical calibration with **Monteverdi**

These transformations can also be done in **Monteverdi**.

The 6S model needs atmospheric parameters to be able to compute radiative terms to estimate the atmospheric contributions on the input signal. Default parameters are available in the module. For atmospheric parameters, it is possible to indicate AERONET file. The AERONET (AErosol RObotic NETwork) program is a federation of ground-based remote sensing aerosol networks established by NASA and PHOTONS (Univ. of Lille 1, CNES, and CNRS-INSU) and is greatly expanded by collaborators from national agencies, institutes, universities, individual scientists, and partners. The program provides accessible public domain database of aerosol optical, mircrophysical and radiative properties.

The module produces four outputs:

- Luminance image.
- TOA reflectance image.
- TOC reflectance image.
- Difference TOA-TOC image, which allows to get the estimation of atmospheric contribution.

4.2.2 Pan-sharpening

Because of physical constrains on the sensor design, it is difficult to achieve high spatial and spectral resolution at the same time : a better spatial resolution means a smaller detector, which in turns

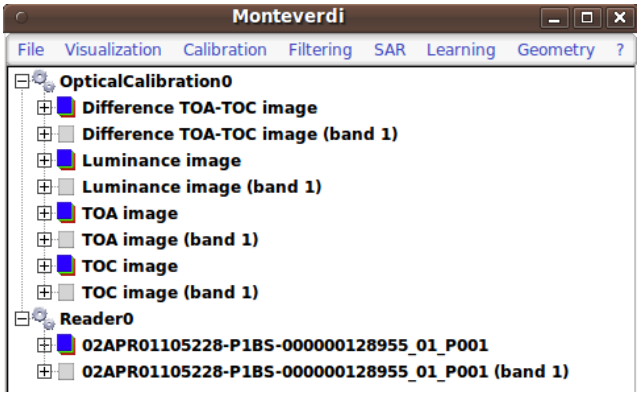


Figure 4.7: Optical calibration module's outputs.

means lesser optical flow on the detector surface. On the contrary, spectral bands are obtained through filters applied on the detector surface, that lowers the optical flow, so that it is necessary to increase the detector size to achieve an acceptable signal to noise ratio.

For these reasons, many high resolution satellite payload are composed of two sets of detectors, which in turns delivers two different kind of images :

- The multi-spectral (XS) image, composed of 3 to 8 spectral bands containing usually blue, green, red and near infra-red bands at a given resolution (usually from 2.8 meters to 2 meters).
- The panchromatic (PAN) image, which is a grayscale image acquired by a detector covering a wider part of the light spectrum, which allows to increase the optical flow and thus to reduce pixel size. Therefore, resolution of the panchromatic image is usually around 4 times lower than the resolution of the multi-spectral image (from 46 centimeters to 70 centimeters).

It is very frequent that those two images are delivered side by side by data providers. Such a dataset is called a bundle. A very common remote sensing processing is to fuse the panchromatic image with the multi-spectral one so as to get an image combining the spatial resolution of the panchromatic image with the spectral richness of the multi-spectral image. This operation is called pan-sharpening.

This fusion operation requires two different steps :

1. The multi-spectral (XS) image is zoomed and registered to the panchromatic image,
2. A pixel-by-pixel fusion operator is applied to the co-registered pixels of the multi-spectral and panchromatic image to obtain the fused pixels.

Using either **OTB Applications** or modules from **Monteverdi**, it is possible to perform both steps in a row, or step-by-step fusion, as described in the above sections.

Pan-sharpening with **OTB Applications**

The *BundleToPerfectSensor* application allows to perform both steps in a row. Seamless sensor modelling is used to perform zooming and registration of the multi-spectral image on the panchromatic image. Then, a simple pan-sharpening is applied, according to the following formula:

$$PXS(i,j) = \frac{PAN(i,j)}{PAN_{smooth}(i,j)} \cdot XS(i,j) \tag{4.1}$$

Where i and j are pixels indices, PAN is the panchromatic image, XS is the multi-spectral image and PAN_{smooth} is the panchromatic image smoothed with a kernel to fit the multi-spectral image scale.

Here is a simple example of how to use the *BundleToPerfectSensor* application:

```
otbcli_BundleToPerfectSensor -inp pan_image -inxs xs_image -out output_image
```

There are two more optional parameters that can be useful for this tool:

- The `-elev` option allows to specify the elevation, either with a DEM formatted for OTB (`-elev.dem` option, see section 4.2.3) or with an average elevation (`-elev.default` option). Since registration and zooming of the multi-spectral image is performed using sensor-models, it may happen that the registration is not perfect in case of landscape with high elevation variation. Using a DEM in this case allows to get better registration.
- The `-lmSpacing` option allows to specify the step of the registration grid between the multi-spectral image and panchromatic image. This is expressed in amount of panchromatic pixels. A lower value gives a more precise registration but implies more computation with the sensor models, and thus increase the computation time. Default value is 10 pixels, which gives sufficient precision in most of the cases.

Pan-sharpening is a quite heavy processing requiring a lot of system resource. The `-ram` option allows you to limit the amount of memory available for the computation, and to avoid overloading your computer. Increasing the available amount of RAM may also result in better computation time, seems it optimises the use of the system resources. Default value is 256 Mb.

Pan-sharpening with **Monteverdi**

Monteverdi allows to perform step-by-step fusion. The followings screenshots highlight operations needed to perform Pan-Sharpening.

- Open panchromatic and multispectral images in monteverdi using the *Open Dataset* module or using the `-il` option of the **Monteverdi** executable.

Figure 4.8: Panchromatic, Zoomed, and registered multispectral image.

Figure 4.9: Pan-sharpened image using the simple RCS module.

- The *Superimpose* module is used to zoomed and registered the multispectral on the panchromatic image. As a result, we get a multispectral dataset with the same geographic extension and the same resolution as the panchromatic image, cf 4.8.
- Now the *Simple RCS pan-sharpening* module can be used using the panchromatic and the multispectral images as inputs. It produces a multispectral image with the same resolution and geographic extension (cf 4.9).

Please also note that since registration and zooming of the multi-spectral image with the panchromatic image relies on sensor modelling, this tool will work only for images whose sensor models is available in **Orfeo ToolBox** (see section 4.2.4 for a detailed list). It will also work with ortho-ready products in cartographic projection.

4.2.3 Digital Elevation Model management

A Digital Elevation Model (DEM) is a georeferenced image (or collection of images) where each pixel corresponds to a local elevation. DEM are useful for tasks involving sensor to ground and

ground to sensor coordinate transforms, like during ortho-rectification (see section 4.2.4). These transforms need to find the intersection between the line of sight of the sensor and the earth geoid. If a simple spheroid is used as the earth model, potentially high localisation errors can be made in areas where elevation is high or perturbed. Of course, DEM accuracy and resolution have a great impact on the precision of these transforms.

Two main available DEM, free of charges, and with worldwide cover, are both delivered as 1 degree by 1 degree tiles:

- The Shuttle Radar topographic Mission (SRTM) is a 90 meters resolution DEM, obtained by radar interferometry during a campaign of the Endeavour space shuttle from NASA in 2000.
- The Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER) is a 30 meters resolution DEM obtained by stereoscopic processing of the archive of the ASTER instrument.

The **Orfeo ToolBox** relies on OSSIM capabilities for sensor modelling and DEM handling. Tiles of a given DEM are supposed to be located within a single directory. General elevation support is also supported from GeoTIFF files.

Whenever an application or **Monteverdi** module requires a DEM, the option **elev.dem** allows set the DEM directory. This directory must contains the DEM tiles, either in DTED or SRTM format, either as GeoTIFF files. Subdirectories are not supported.

Depending on the reference of the elevation, you also need to use a geoid to manage elevation accurately. For this, you need to specify a path to a file which contains the geoid. Geoid corresponds to the equipotential surface that would coincide with the mean ocean surface of the Earth (see). We provide one geoid in the OTB-Data repository available here.

In all applications, the option **elev.geoid** allows to manage the path to the geoid. Finally, it is also possible to use an average elevation in case no DEM is available by using the **elev.default** option.

4.2.4 Ortho-rectification and map projections

There are several level of products available on the remote sensing imagery market. The most basic level often provide the geometry of acquisition (sometimes called the raw geometry). In this case, pixel coordinates can not be directly used as geographical positions. For most sensors (but not for all), the different lines corresponds to different acquisition times and thus different sensor positions, and different rows correspond to different cells of the detector.

The mapping of a raw image so as to be registered to a cartographic grid is called ortho-rectification, and consist in inverting the following effects (at least):

- In most cases, lines are orthogonal to the sensor trajectory, which is not exactly (and in some case not at all) following a north-south axis,

- Depending on the sensor, the line of sight may be different from a Nadir (ground position of the sensor), and thus a projective warping may appear,
- The variation of height in the landscape may result in severe warping of the image.

Moreover, depending on the area of the world the image has been acquired on, different map projections should be used.

The ortho-rectification process is as follows: once an appropriate map projection has been defined, a localisation grid is computed to map pixels from the raw image to the ortho-rectified one. Pixels from the raw image are then interpolated according to this grid in order to fill the ortho-rectified pixels.

Ortho-rectification can be performed either with **OTB Applications** or **Monteverdi**. Sensor parameters and image meta-data are seamlessly read from the image files without needing any user interaction, provided that all auxiliary files are available. The sensor for which **Orfeo ToolBox** supports ortho-rectification of raw products are the following:

- Pleiades,
- SPOT5,
- Ikonos,
- Quickbird,
- GeoEye,
- WorldView.

In addition, GeoTiff and other file format with geographical information are seamlessly read by **Orfeo ToolBox**, and the ortho-rectification tools can be used to re-sample these images in another map projection.

Ortho-rectification with **OTB Applications**

The *OrthoRectification* application allows to perform ortho-rectification and map re-projection. The simplest way to use it is the following command:

```
otbcli_OrthoRectification -io.in input_image -io.out output_image
```

In this case, the tool will automatically estimates all the necessary parameters:

- The map projection is set to UTM (a worldwide map projection) and the UTM zone is automatically estimated,

- The ground sampling distance of the output image is computed to fit the image resolution,
- The region of interest (upper-left corner and size of the image) is estimated so as to contain the whole input image extent.

In order to use a Digital Elevation Model (see section 4.2.3) for better localisation performances, one can pass the directory containing the DEM tiles to the application:

```
otbcli_OrthoRectification -io.in input_image
                          -io.out output_image
                          -elev.dem dem_dir
```

If one wants to use a different map projection, the *-map* option may be used (example with *lambert93* map projection):

```
otbcli_OrthoRectification -io.in input_image
                          -io.out output_image
                          -elev.dem dem_dir
                          -map lambert93
```

Map projections handled by the application are the following (please note that the ellipsoid is always WGS84):

- UTM : `-map utm`
  The UTM zone and hemisphere can be set by the options `-map.utm.zone` and `-map.utm.northhem`.
- Lambert 2 etendu: `-map lambert2`
- Lambert 93: `-map lambert93`
- TransMercator: `-map transmercator`
  The related parameters (false easting, false northing and scale factor) can be set by the options `-map.transmercator.falseeasting`, `-map.transmercator.falsenorthing` and `-map.transmercator.scale`
- WGS : `-map wgs`
- Any map projection system with an EPSG code : `-map epsg`
  The EPSG code is set with the option `-map.epsg.code`

The group `outputs` contains parameters to set the origin, size and spacing of the output image. For instance, the ground spacing can be specified as follows:

```
otbcli_OrthoRectification -io.in input_image
                          -io.out output_image
                          -elev.dem dem_dir
                          -map lambert93
                          -outputs.spacingx spx
                          -outputs.spacingy spy
```

Please note that since the y axis of the image is bottom oriented, the y spacing should be negative to avoid switching north and south direction.

A user-defined region of interest to ortho-rectify can be specified as follows:

```
otbcli_OrthoRectification -io.in input_image
                          -io.out output_image
                          -elev.dem dem_dir
                          -map lambert93
                          -outputs.spacingx spx
                          -outputs.spacingy spy
                          -outputs.ulx ul_x_coord
                          -outputs.uly ul_y_coord
                          -outputs.sizex x_size
                          -outputs.sizey y_size
```

Where the `-outputs.ulx` and `-outputs.uly` options allow to specify the coordinates of the upper-left corner of the output image. The `-outputs.sizex` and `-outputs.sizey` options allow to specify the size of the output image.

A few more interesting options are available:

- The `-opt.rpc` option allows to use an estimated RPC model instead of the rigorous SPOT5 model, which speeds-up the processing,
- The `-opt.gridspacing` option allows to define the spacing of the localisation grid used for ortho-rectification. A coarser grid results in speeding-up the processing, but with potential loss of accuracy. A standard value would be 10 times the ground spacing of the output image.
- The `-interpolator` option allows to change the interpolation algorithm between nearest neighbor, linear and bicubic. Default is nearest neighbor interpolation, but bicubic should be fine in most cases.

- The `-opt.ram` option allows to specify the amount of memory available for the processing (in Mb). Default is 256 Mb. Increasing this value to fit the available memory on your computer might speed-up the processing.

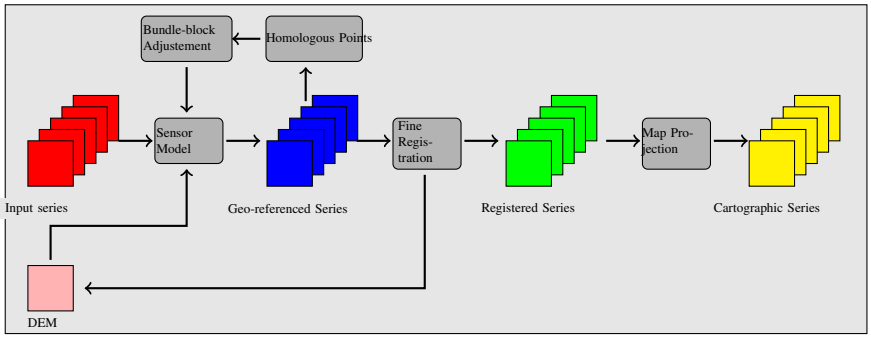
4.2.5 Residual registration

Image registration is a fundamental problem in image processing. The aim is to align two or more images of the same scene often taken at different times, from different viewpoints, or by different sensors. It is a basic step for orthorectification, image stitching, image fusion, change detection... But this process is also critical for stereo reconstruction process to be able to obtain an accurate estimation of epipolar geometry.

Sensor model is generally not sufficient to provide image registrations. Indeed, several sources of geometric distortion can be contained in optical remote sensing images including earth rotation, platform movement, non linearity...

They result in geometric errors on scene level, image level and pixel level. It is critical to rectify the errors before a thematic map is generated, especially when the remote sensing data need to be integrated together with other GIS data.

This figure illustrates the generic workflow in the case of image series registration:



We will now illustrate this process by applying this workflow to register two images. This process can be easily extended to perform image series registration.

The aim of this example is to describe how to register a Level 1 QuickBird image over an orthorectify Pleiades image over the area of Toulouse, France.

Extract metadata from the image reference

We first dump geometry metadata of the image we want to refine in a text file. In OTB, we use the extension *.geom* for this type of file. As you will see the application which will estimate a

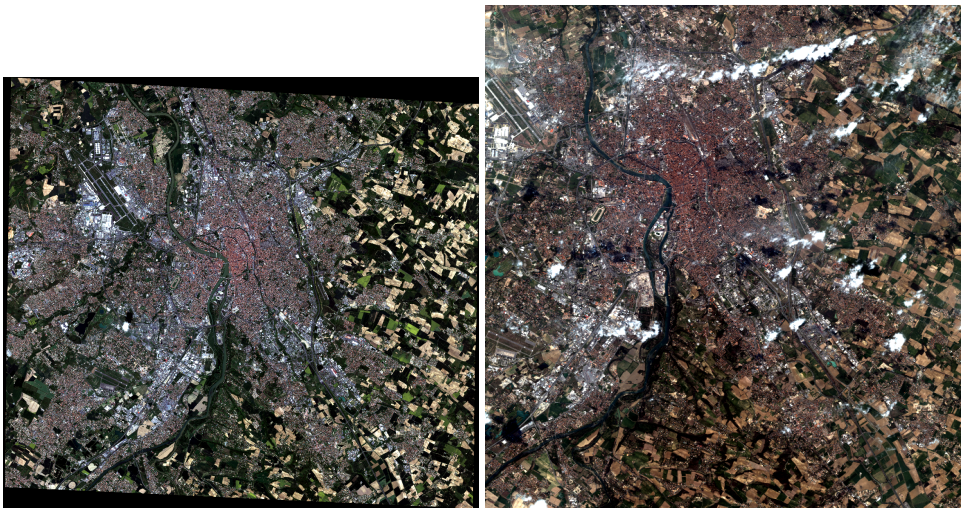


Figure 4.10: From left to right: Pleiades ortho-image, and original QuickBird image over Toulouse

refine geometry only needs as input this metadata and a set of homologous points. The refinement application will create a new *.geom* file containing refined geometry parameters which can be used after for reprojection for example.

The use of external *.geom* file is available in OTB since release 3.16. See here for more information.

```
otbcli_ReadImageInfo   -in slave_image
                       -outkwl TheGeom.geom
```

Extract homologous points from images

The main idea of the residual registration is to estimate an second transformation (after the application of sensors model).

The homologous point application use interest point detection method to get a set of point which match in both images.

The basic idea is to use this set of homologous points and estimate with them a residual transformation between the two images.

There is a wide variety of keypoint detector in the literature. They allow to detect and describe local features in images. These algorithms provide for each interesting point a “feature description”. This descriptor has the property to be invariant to image translation, scaling, and rotation, partially invari-

ant to illumination changes and robust to local geometric distortion. keypoints. Features extracted from the input images are then matched against each other. These correspondences are then used to create the homologous points.

SIFT or SURF keypoints can be computed in the application. The band on which keypoints are computed can be set independently for both images.

The application offers two modes :

- the first is the full mode where keypoints are extracted from the full extent of both images (please note that in this mode large image file are not supported).
- The second mode, called *geobins*, allows to set-up spatial binning so as to get fewer points spread across the entire image. In this mode, the corresponding spatial bin in the second image is estimated using geographical transform or sensor modeling, and is padded according to the user defined precision.

Moreover, in both modes the application can filter matches whose co-localization in the first image exceed this precision. Last, the elevation parameters allow to deal more precisely with sensor modelling in case of sensor geometry data. The *outvector* option allows to create a vector file with segments corresponding to the localization error between the matches.

Finally, with the *2wgs84* option, you can match two sensor geometry images or a sensor geometry image with an ortho-rectified reference. In all cases, you get a list of ground control points spread all over your image.

```
otbcli_HomologousPointsExtraction   -in1 slave_image
                                    -in2 reference_image
                                    -algorithm.surf
                                    -mode geobins
                                    -mode.geobins.binstep 512
                                    -mode.geobins.binsize 512
                                    -mfilter 1
                                    -precision 20
                                    -2wgs84 1
                                    -out homologous_points.txt
                                    -outvector points.shp
                                    -elev.dem dem_path/SRTM4-HGT/
                                    -elev.geoid OTB-Data/Input/DEM/egm96.grd
```

Note that for a proper use of the application, elevation must be correctly set (including DEM and geoid file).

Geometry refinement using homologous points

Now that we can use this set of tie points to estimate a residual transformation.For this we use the dedicated application called **RefineSensorModel**. This application make use of OSSIM capabilities to align the sensor model.

It reads the input geometry metadata file (*.geom*) which contains the sensor model information that we want to refine and the text file (homologous_points.txt) containing the list of ground control point. It performs a least-square fit of the sensor model adjustable parameters to these tie points and produces an updated geometry file as output (the extension which is always use is *.geom*)

The application can provide as well an optional ground control points based statistics file and a vector file containing residues that you can display in a GIS software.

Please note again that for a proper use of the application, elevation must be correctly set (including DEM and geoid file). The map parameters allows to choose a map projection in which the accuracy will be estimated (in meters).

Accuracy values are provided as output of the application (computed using tie points location) and allow also to control the precision of the estimated model.

```
otbcli_RefineSensorModel   -elev.dem dem_path/SRTM4-HGT/
                           -elev.geoid OTB-Data/Input/DEM/egm96.grd
                           -ingeom slave_image.geom
                           -outgeom refined_slave_image.geom
                           -inpoints homologous_points.txt
                           -outstat stats.txt
                           -outvector refined_slave_image.shp
```

Orthorecrtify image using the affine geometry

Now we will show how we can use this new sensor model. In our case we'll use this sensor model to orthorectify the image over the PlÃ©iades reference. **Orfeo ToolBox** offers since version 3.16 the possibility to use hrefhttp://wiki.orfeo-toolbox.org/index.php/ExtendedFileNameextend image path to use different metadata file as input. That's what we are going to use there to orthorectify the QuickBird image using the *.geom* file obtained by the **RefineSensorModel** applications. over the first one using for the second image estimated sensor model which take into account the original sensor model of the slave and which also fit to the set of tie points.

```
otbcli_OrthoRectification   -io.in slave_image?&geom=TheRefinedGeom.geom
                            -io.out ortho_slave_image
                            -elev.dem dem_path/SRTM4-HGT/
```

```
                                -elev.geoid OTB-Data/Input/DEM/egm96.grd
```

As a result, if you've got enough homologous points in images and control that the residual error between the set of tie points and the estimated sensor model is small, you must achieve a good registration now between the 2 rectified images. Normally far better than 'only' performing separate orthorectification over the 2 images.

This methodology can be adapt and apply in several cases, for example :

- register stereo pair of images and estimate accurate epipolar geometry
- registration prior to change detection

4.3 Image processing and information extraction

4.3.1 Simple calculus with channels

The *BandMath* application provides a simple and efficient way to perform band operations. The command line application and the corresponding Monteverdi module (shown in the section 2.4.4) are based on the same standards. It computes a band wise operation according to a user defined mathematical expression. The following code computes the absolute difference between first bands of two images:

```
otbcli_BandMath -il input_image_1 input_image_2
                -exp "abs(im1b1 - im2b1)"
                -out output_image
```

The naming convention "im[x]b[y]" designates the yth band of the xth input image.

The *BandMath* application embeds built-in operators and functions (listed here), allowing a vast choice of possible operations.

4.3.2 Segmentation

Segmenting objects across a very high resolution scene and with a controlled quality is a difficult task for which no method has reached a sufficient level of performance to be considered as operational.

Even if we leave aside the question of segmentation quality and consider that we have a method performing reasonably well on our data and objects of interest, the task of scaling up segmentation to real very high resolution data is itself challenging. First, we can not load the whole data into memory, and there is a need for on the flow processing which does not cope well with traditional

segmentation algorithms. Second, the result of the segmentation process itself is difficult to represent and manipulate efficiently.

The experience of segmenting large remote sensing images is packed into a single *Segmentation* in **OTB Applications**.

You can find more information about this application here.

4.3.3 Large-Scale Mean-Shift (LSMS) segmentation

LSMS is a segmentation workflow which allows to perform tile-wise segmentation of very large image with theoretical guarantees of getting identical results to those without tiling. It has been developed by David Youssefi and Julien Michel during David internship at CNES and is to be published soon.

The workflow consists in chaining 3 or 4 dedicated applications and produces a GIS vector file with artifact-free polygons corresponding to the segmented image, as well as mean and variance of the radiometry of each band for each polygon.

Step 1: Mean-Shift Smoothing

The first step of the workflow is to perform Mean-Shift smoothing with the *MeanShiftSmoothing* application:

```
otbcli_MeanShiftSmoothing -in input_image
                          -fout filtered_range.tif
                          -foutpos filtered_spat.tif
                          -ranger 30
                          -spatialr 5
                          -maxiter 10
                          -modesearch 0
```

Note that the *modesearch* option should be disabled, and that the *foutpos* parameter is optional: it can be activated if you want to perform the segmentation based on both spatial and range modes.

This application will smooth large images by streaming them, and deactivating the *modesearch* will guarantee that the results will not depend on the streaming scheme. Please also note that the *maxiter* is used to set the margin to ensure these identical results, and as such increasing the *maxiter* may have an additional impact on processing time.

Step 2: Segmentation

The next step is to produce an initial segmentation based on the smoothed images produced by the *MeanShiftSmoothing* application. To do so, the *LSMSSegmentation* will process them by tiles whose

dimensions are defined by the *tilesizex* and *tilesizey* parameters, and by writing intermediate images to disk, thus keeping the memory consumption very low throughout the process. The segmentation will group together adjacent pixels whose range distance is below the *ranger* parameter and (optionally) spatial distance is below the *spatialr* parameter.

```
otbcli_LSMSSegmentation -in filtered_range.tif
                        -inpos filtered_spatial.tif
                        -out  segmentation.tif uint32
                        -ranger 30
                        -spatialr 5
                        -minsize 0
                        -tilesizex 256
                        -tilesizey 256
```

Note that the final segmentation image may contains a very large number of segments, and the *uint32* image type should therefore be used to ensure that there will be enough labels to index those segments. The *minsize* parameter will filter segments whose size in pixels is below its value, and their labels will be set to 0 (nodata).

Please note that the output segmented image may look patchy, as if there were tiling artifacts: this is because segments are numbered sequentially with respect to the order in which tiles are processed. You will see after the result of the vectorization step that there are no artifacts in the results.

The *LSMSSegmentation* application will write as many intermediate files as tiles needed during processing. As such, it may require twice as free disk space as the final size of the final image. The *cleanup* option (active by default) will clear the intermediate files during the processing as soon as they are not needed anymore. By default, files will be written to the current directory. The *tmpdir* option allows to specify a different directory for these intermediate files.

Step 3 (optional): Merging small regions

The *LSMSSegmentation* application allows to filter out small segments. In the output segmented image, those segments will be removed and replaced by the background label (0). Another solution to deal with the small regions is to merge them with the closest big enough adjacent region in terms of radiometry. This is handled by the *LSMSSmallRegionsMerging* application, which will output a segmented image where small regions have been merged. Again, the *uint32* image type is advised for this output image.

```
otbcli_LSMSSmallRegionsMerging -in filtered_range.tif
                               -inseg segementation.tif
                               -out segmentation_merged.tif uint32
                               -minsize 10
                               -tilesizex 256
                               -tilesizey 256
```

The *minsize* parameter allows to specify the threshold on the size of the regions to be merged. Like the *LSMSSegmentation* application, this application will process the input images tile-wise to keep resources usage low, with the guarantee of identical results. You can set the tile size using the *tilesizex* and *tilesizey* parameters. However unlike the *LSMSSegmentation* application, it does not require to write any temporary file to disk.

Step 4: Vectorization

The last step of the LSMS workflow consists in the vectorization of the segmented image into a GIS vector file. This vector file will contain one polygon per segment, and each of these polygons will hold additional attributes denoting the label of the original segment, the size of the segment in pixels, and the mean and variance of each band over the segment. The projection of the output GIS vector file will be the same as the projection from the input image (if input image has no projection, so does the output GIS file).

```
otbcli_LSMSVectorization -in input_image
                         -inseg segmentation_merged.tif
                         -out segmentation_merged.shp
                         -tilesizex 256
                         -tilesizey 256
```

This application will process the input images tile-wise to keep resources usage low, with the guarantee of identical results. You can set the tile size using the *tilesizex* and *tilesizey* parameters. However unlike the *LSMSSegmentation* application, it does not require to write any temporary file to disk.

4.3.4 Dempster Shafer based Classifier Fusion

This framework is dedicated to perform cartographic validation starting from the result of a detection (for example a road extraction), enhance the results fiability by using a classifier fusion algorithm. Using a set of descriptor, the processing chain validates or invalidates the input geometrical features.

Fuzzy Model (requisite)

The *DSFuzzyModelEstimation* application performs the fuzzy model estimation (once by use case: descriptor set / Belief support / Plausibility support). It has the following input parameters :

- `-psin` a vector data of positive samples enriched according to the ”Compute Descriptors” part
- `-nsin` a vector data of negative samples enriched according to the ”Compute Descriptors” part
- `-belsup` a support for the Belief computation

- `-plasup` a support for the Plausibility computation
- `-desclist` an initialization model (xml file) or a descriptor name list (listing the descriptors to be included in the model)

The application can be used like this:

```
otbcli_DSFuzzyModelEstimation -psin     PosSamples.shp
                              -nsin     NegSamples.shp
                              -belsup   "ROADSA"
                              -plasup   "NONDVI" "ROADSA" "NOBUIL"
                              -desclist "NONDVI" "ROADSA" "NOBUIL"
                              -out      FuzzyModel.xml
```

The output file `FuzzyModel.xml` contains the optimal model to perform informations fusion.

First Step: Compute Descriptors

The first step in the classifier fusion based validation is to compute, for each studied polyline, the choosen descriptors. In this context, the *ComputePolylineFeatureFromImage* application can be used for a large range of descriptors. It has the following inputs :

- `-in` an image (of the sudied scene) corresponding to the choosen descriptor (NDVI, building Mask...)
- `-vd` a vector data containing polyline of interest
- `-expr` a formula ("b1 >0.4", "b1 == 0") where b1 is the standard name of input image first band
- `-field` a field name corresponding to the descriptor codename (NONDVI, ROADSA...)

The output is a vector data containing polylines with a new field containing the descriptor value. In order to add the "NONDVI" descriptor to an input vector data ("inVD.shp") corresponding to the percentage of pixels along a polyline that verifies the formula "NDVI >0.4" :

```
otbcli_ComputePolylineFeatureFromImage -in   NDVI.TIF
                                       -vd  inVD.shp
                                       -expr  "b1 > 0.4"
                                       -field "NONDVI"
                                       -out   VD_NONDVI.shp
```

`NDVI.TIF` is the NDVI mono band image of the studied scene. This step must be repeated for each choosen descriptor:

```
otbcli_ComputePolylineFeatureFromImage -in   roadSpectralAngle.TIF
                                       -vd  VD_NONDVI.shp
                                       -expr  "b1 > 0.24"
                                       -field "ROADSA"
                                       -out   VD_NONDVI_ROADSA.shp

otbcli_ComputePolylineFeatureFromImage -in   Buildings.TIF
                                       -vd  VD_NONDVI_ROADSA.shp
                                       -expr  "b1 == 0"
                                       -field "NOBUILDING"
                                       -out   VD_NONDVI_ROADSA_NOBUIL.shp
```

Both `NDVI.TIF` and `roadSpectralAngle.TIF` can be produced using **Monteverdi** feature extraction capabilities, and `Buildings.TIF` can be generated using **Monteverdi** rasterization module. From now on, `VD_NONDVI_ROADSA_NOBUIL.shp` contains three descriptor fields. It will be used in the following part.

Second Step: Feature Validation

The final application (*VectorDataDSValidation*) will validate or unvalidate the studied samples using the Dempster-Shafer theory . Its inputs are :

- `-in` an enriched vector data "VD_NONDVI_ROADSA_NOBUIL.shp"
- `-belsup` a support for the Belief computation
- `-plasup` a support for the Plausibility computation
- `-descmod` a fuzzy model FuzzyModel.xml

The output is a vector data containing only the validated samples.

```
otbcli_VectorDataDSValidation -in      extractedRoads_enriched.shp
                              -descmod FuzzyModel.xml
                              -out     validatedSamples.shp
```

4.4 Classification

4.4.1 Pixel based classification

The classification in the application framework provides a supervised pixel-wise classification chain based on learning from multiple images, and using one specified machine learning method like

SVM, Bayes, KNN, Random Forests, Artificial Neural Network, and others...(see application help of *TrainImagesClassifier* for further details about all the available classifiers). It supports huge images through streaming and multi-threading. The classification chain performs a training step based on the intensities of each pixel as features. Please note that all the input images must have the same number of bands to be comparable.

Statistics estimation

In order to make these features comparable between each training images, the first step consists in estimating the input images statistics. These statistics will be used to center and reduce the intensities (mean of 0 and standard deviation of 1) of samples based on the vector data produced by the user. To do so, the *ComputeImagesStatistics* tool can be used:

```
otbcli_ComputeImagesStatistics -il  im1.tif im2.tif im3.tif
                               -out images_statistics.xml
```

This tool will compute each band mean, compute the standard deviation based on pooled variance of each band and finally export them to an XML file. The features statistics XML file will be an input of the following tools.

Building the training data set

As the chain is supervised, we first need to build a training set with positive examples of different objects of interest. This can be done with Monteverdi Vectorization module (Fig.4.11). These polygons must be saved in OGR vector format supported by GDAL like ESRI shapefile for example.

This operation will be reproduced on each image used as input of the training function.

Please note that the positive examples in the vector data should have a “Class“ field with a label value higher than 1 and coherent in each images.

You can generate the vector data set with Quantum GIS software for example and save it in an OGR vector format supported by GDAL (ESRI sphapefile for example). **OTB Applications** should be able to transform the vector data into the image coordinate system.

Performing the learning scheme

Once images statistics have been estimated, the learning scheme is the following:

1. For each input image:
   (a) Read the region of interest (ROI) inside the shapefile,
   (b) Generate validation and training data within the ROI,

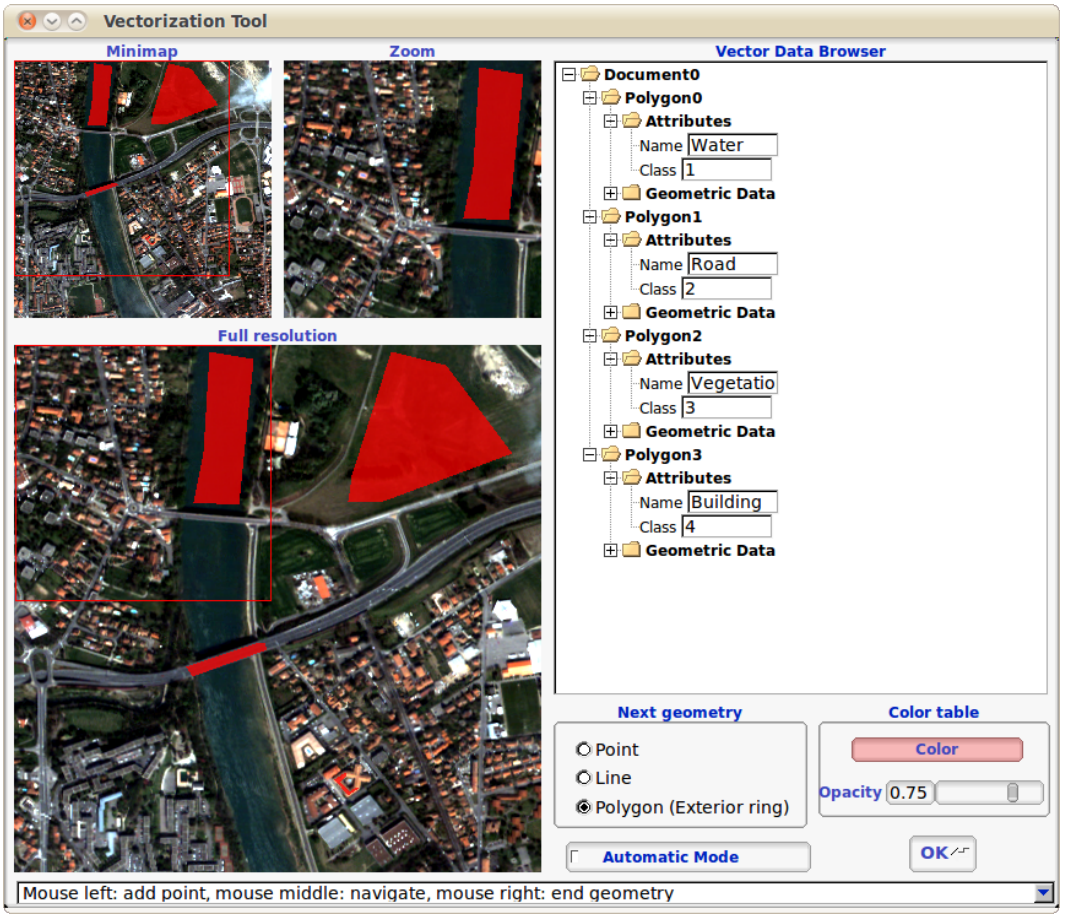


Figure 4.11: A training data set builded with the vectorization monteverdi module.

(c) Add vectors respectively to the training samples set and the validation samples set.

2. Increase the size of the training samples set and balance it by generating new noisy samples from the previous ones,
3. Perform the learning with this training set
4. Estimate performances of the classifier on the validation samples set (confusion matrix, precision, recall and F-Score).

Let us consider a SVM classification. These steps can be performed by the *TrainImagesClassifier* command-line using the following:

```
otbcli_TrainImagesClassifier -io.il      im1.tif im2.tif im3.tif
                             -io.vd      vd1.shp vd2.shp vd3.shp
                             -io.imstat  images_statistics.xml
                             -classifier svm (classifier_for_the_training)
                             -io.out     model.svm
```

Additional groups of parameters are also available (see application help for more details):

- `-elev` Handling of elevation (DEM or average elevation)
- `-sample` Group of parameters for sampling
- `-classifier` Classifiers to use for the training, and their corresponding groups of parameters

Using the classification model

Once the classifier has been trained, one can apply the model to classify pixel inside defined classes on a new image using the *ImageClassifier* application:

```
otbcli_ImageClassifier -in     image.tif
                       -imstat images_statistics.xml
                       -model  model.svm
                       -out    labeled_image.tif
```

You can set an input mask to limit the classification to the mask area with value >0.

Validating the classification model

The performance of the model generated by the *TrainImagesClassifier* application is directly estimated by the application itself, which displays the precision, recall and F-score of each class, and can generate the global confusion matrix as an output *.CSV file.

With the *ConputeConfusionMatrix* application, it is also possible to estimate the performance of a model from a classification map generated with the *ImageClassifier* application. This labeled image is compared to positive reference samples (either represented as a raster labeled image or as a vector data containing the reference classes). It will compute the confusion matrix and precision, recall and F-score of each class too, based on the ConfusionMatrixCalculator class.

```
otbcli_ComputeConfusionMatrix -in                labeled_image.tif
                              -ref               vector
                              -ref.vector.in     vectordata.shp
                              -ref.vector.field  Class (name_of_label_field)
                              -out               confusion_matrix.csv
```

Fancy classification results

Color mapping can be used to apply color transformations on the final graylevel label image. It allows to get an RGB classification map by re-mapping the image values to be suitable for display purposes. One can use the *ColorMapping* application. This tool will replace each label with an 8-bits RGB color specificied in a mapping file. The mapping file should look like this :

```
# Lines beginning with a # are ignored
1 255 0 0
```

In the previous example, 1 is the label and 255 0 0 is a RGB color (this one will be rendered as red). To use the mapping tool, enter the following :

```
otbcli_ColorMapping -in                labeled_image.tif
                    -method            custom
                    -method.custom.lut lut_mapping_file.txt
                    -out               RGB_color_image.tif
```

Other look-up tables (LUT) are available : standard continuous LUT, optimal LUT, and LUT computed over a support image.

Example

We consider 4 classes: water, roads, vegetation and buildings with red roofs. Data is available in the OTB-Data repository and this image is produced with the commands inside this file.

4.4.2 Fusion of classification maps

After having processed several classifications of the same input image but from different models or methods (SVM, KNN, Random Forest,...), it is possible to make a fusion of these classification maps

Figure 4.12: From left to right: Original image, result image with fusion (with monteverdi viewer) of original image and fancy classification and input image with fancy color classification from labeled image.

with the *FusionOfClassifications* application which uses either majority voting or the Demspter Shafer framework to handle this fusion. The Fusion of Classifications generates a single more robust and precise classification map which combines the information extracted from the input list of labeled images.

The *FusionOfClassifications* application has the following input parameters :

- `-il` list of input labeled classification images to fuse
- `-out` the output labeled image resulting from the fusion of the input classification images
- `-method` the fusion method (either by majority voting or by Dempster Shafer)
- `-nodatalabel` label for the no data class (default value = 0)
- `-undecidedlabel` label for the undecided class (default value = 0)

The input pixels with the nodata class label are simply ignored by the fusion process. Moreover, the output pixels for which the fusion process does not result in a unique class label, are set to the undecided value.

Majority voting for the fusion of classifications

In the Majority Voting method implemented in the *FusionOfClassifications* application, the value of each output pixel is equal to the more frequent class label of the same pixel in the input classification maps. However, it may happen that the more frequent class labels are not unique in individual pixels. In that case, the undecided label is attributed to the output pixels.

The application can be used like this:

```
otbcli_FusionOfClassifications  -il             cmap1.tif cmap2.tif cmap3.tif
                                -method         majorityvoting
                                -nodatalabel    0
                                -undecidedlabel 10
                                -out            MVFusedClassificationMap.tif
```

Let us consider 6 independent classification maps of the same input image (Cf. left image in Fig. 4.12) generated from 6 different SVM models. The Fig. 4.13 represents them after a color mapping by the same LUT. Thus, 4 classes (water: blue, roads: gray, vegetation: green, buildings with red roofs: red) are observable on each of them.

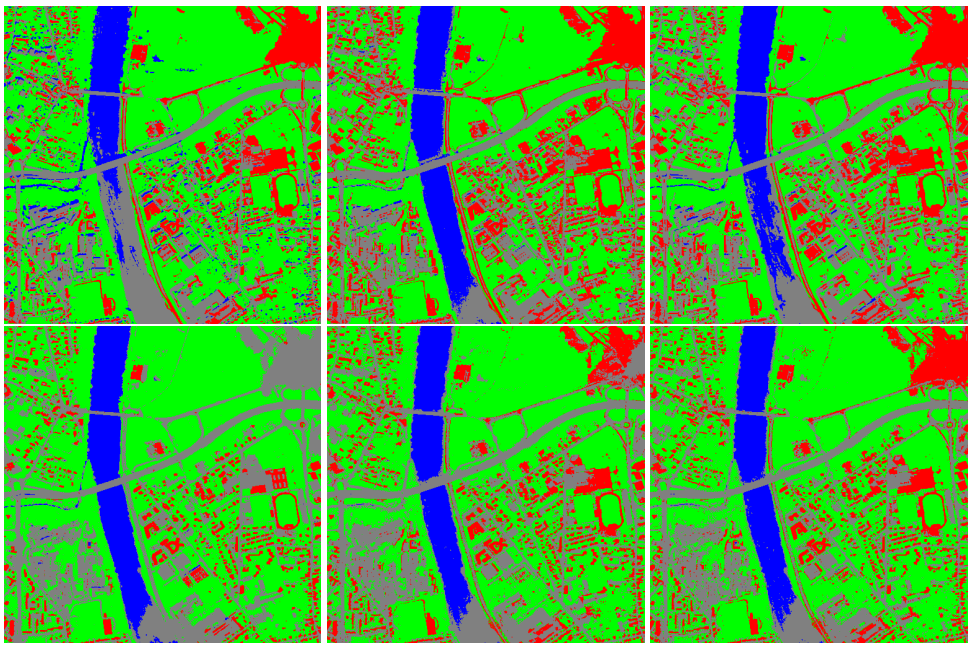


Figure 4.13: Six fancy colored classified images to be fused, generated from 6 different SVM models.

As an example of the *FusionOfClassifications* application by *majority voting*, the fusion of the six input classification maps represented in Fig. 4.13 leads to the classification map illustrated on the right in Fig. 4.14. Thus, it appears that this fusion highlights the more relevant classes among the six different input classifications. The white parts of the fused image correspond to the undecided class labels, i.e. to pixels for which there is not a unique majority voting.

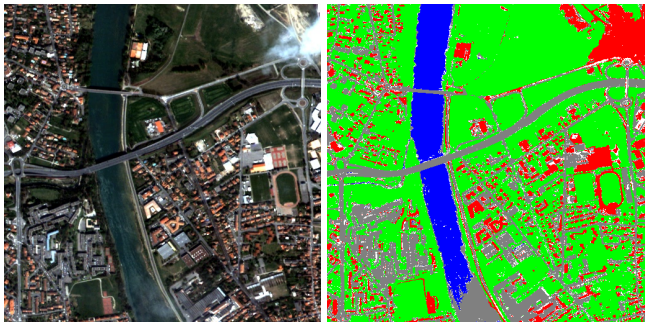


Figure 4.14: From left to right: Original image, and fancy colored classified image obtained by a majority voting fusion of the 6 classification maps represented in Fig. 4.13 (water: blue, roads: gray, vegetation: green, buildings with red roofs: red, undecided: white).

Dempster Shafer framework for the fusion of classifications

The *FusionOfClassifications* application, handles another method to compute the fusion: the Dempster Shafer framework. In the Dempster-Shafer theory, the performance of each classifier resulting in the classification maps to fuse are evaluated with the help of the so-called *belief function* of each class label, which measures the degree of belief that the corresponding label is correctly assigned to a pixel. For each classifier, and for each class label, these belief functions are estimated from another parameter called the *mass of belief* of each class label, which measures the confidence that the user can have in each classifier according to the resulting labels.

In the Dempster Shafer framework for the fusion of classification maps, the fused class label for each pixel is the one with the maximal belief function. In case of multiple class labels maximizing the belief functions, the output fused pixels are set to the undecided value.

In order to estimate the confidence level in each classification map, each of them should be confronted with a ground truth. For this purpose, the masses of belief of the class labels resulting from a classifier are estimated from its confusion matrix, which is itself exported as a *.CSV file with the help of the *ComputeConfusionMatrix* application. Thus, using the Dempster Shafer method to fuse classification maps needs an additional input list of such *.CSV files corresponding to their respective confusion matrices.

The application can be used like this:

```
otbcli_FusionOfClassifications  -il             cmap1.tif cmap2.tif cmap3.tif
                                -method         dempstershafer
                                -method.dempstershafer.cmfl
                                                cmat1.csv cmat2.csv cmat3.csv
                                -nodatalabel    0
                                -undecidedlabel 10
```

```
                    -out                DSFusedClassificationMap.tif
```

As an example of the *FusionOfClassifications* application by *Dempster Shafer*, the fusion of the six input classification maps represented in Fig. 4.13 leads to the classification map illustrated on the right in Fig. 4.15. Thus, it appears that this fusion gives access to a more precise and robust classification map based on the confidence level in each classifier.

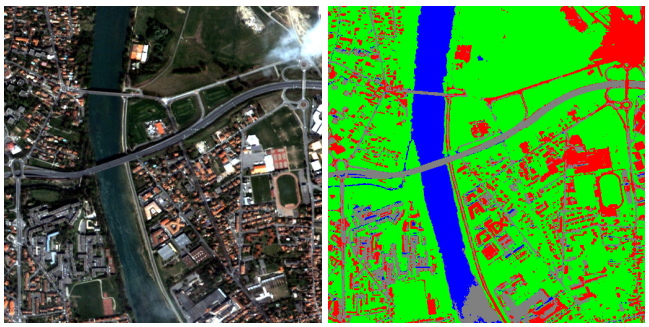


Figure 4.15: From left to right: Original image, and fancy colored classified image obtained by a Dempster Shafer fusion of the 6 classification maps represented in Fig. 4.13 (water: blue, roads: gray, vegetation: green, buildings with red roofs: red, undecided: white).

Recommandations to properly use the fusion of classification maps

In order to properly use the *FusionOfClassifications* application, some points should be considered. First, the `list_of_input_images` and `OutputFusedClassificationImage` are single band labeled images, which means that the value of each pixel corresponds to the class label it belongs to, and labels in each classification map must represent the same class. Secondly, the undecided label value must be different from existing labels in the input images in order to avoid any ambiguity in the interpretation of the `OutputFusedClassificationImage`.

4.4.3 Majority voting based classification map regularization

Resulting classification maps can be regularized in order to smoothen irregular classes. Such a regularization process improves classification results by making more homogeneous areas which are easier to handle.

Majority voting for the classification map regularization

The *ClassificationMapRegularization* application performs a regularization of a labeled input image based on the Majority Voting method in a specified ball shaped neighborhood. For each center pixel,

Majority Voting takes the more representative value of all the pixels identified by the structuring element and then sets the output center pixel to this majority label value. The ball shaped neighborhood is identified by its radius expressed in pixels.

Handling ambiguity and not classified pixels in the majority voting based regularization

Since, the Majority Voting regularization may lead to not unique majority labels in the neighborhood, it is important to define which behaviour the filter must have in this case. For this purpose, a Boolean parameter (called ip.suvbool) is used in the *ClassificationMapRegularization* application to choose whether pixels with more than one majority class are set to Undecided (true), or to their Original labels (false = default value).

Moreover, it may happen that pixels in the input image do not belong to any of the considered class. Such pixels are assumed to belong to the NoData class, the label of which is specified as an input parameter for the regularization. Therefore, those NoData input pixels are invariant and keep their NoData label in the output regularized image.

The *ClassificationMapRegularization* application has the following input parameters :

- `-io.in` a labeled input image resulting from a previous classification process
- `-io.out` the output labeled image corresponding to the regularization of the input image
- `-ip.radius` an integer corresponding to the radius of the ball shaped structuring element (default value = 1 pixel)
- `-ip.suvbool` a Boolean number used to choose whether pixels with more than one majority class are set to Undecided (true), or to their Original labels (false = default value). Please note that the Undecided value must be different from existing labels in the input image
- `-ip.nodatalabel` label for the NoData class. Such input pixels keep their NoData label in the output image (default value = 0)
- `-ip.undecidedlabel` label for the Undecided class (default value = 0).

The application can be used like this:

```
otbcli_ClassificationMapRegularization  -io.in              labeled_image.tif
                                        -ip.radius          3
                                        -ip.suvbool         true
                                        -ip.nodatalabel     10
                                        -ip.undecidedlabel  7
                                        -io.out             regularized.tif
```

Recommandations to properly use the majority voting based regularization

In order to properly use the *ClassificationMapRegularization* application, some points should be considered. First, both `InputLabeledImage` and `OutputLabeledImage` are single band labeled images, which means that the value of each pixel corresponds to the class label it belongs to. The `InputLabeledImage` is commonly an image generated with a classification algorithm such as the SVM classification. Remark: both `InputLabeledImage` and `OutputLabeledImage` are not necessarily of the same datatype. Secondly, if ip.suvbool == true, the Undecided label value must be different from existing labels in the input labeled image in order to avoid any ambiguity in the interpretation of the regularized `OutputLabeledImage`. Finally, the structuring element radius must have a minimum value equal to 1 pixel, which is its default value. Both NoData and Undecided labels have a default value equal to 0.

Example

Resulting from the *ColorMapping* application presented in section 4.4.1, and illustrated in Fig. 4.12, the Fig. 4.16 shows a regularization of a classification map composed of 4 classes: water, roads, vegetation and buildings with red roofs. The radius of the ball shaped structuring element is equal to 3 pixels, which corresponds to a ball included in a 7 x 7 pixels square. Pixels with more than one majority class keep their original labels.

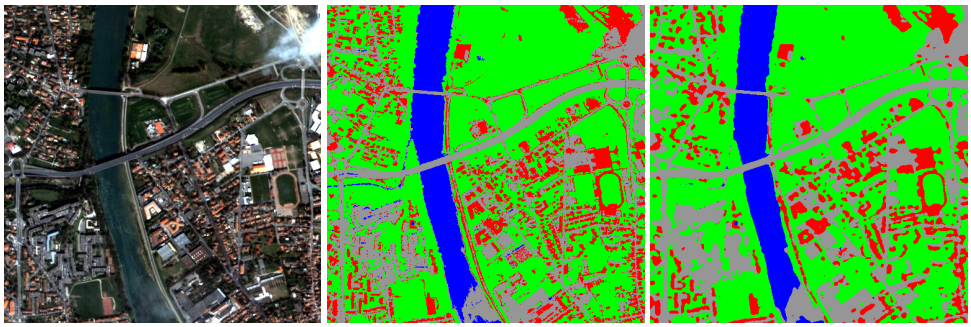


Figure 4.16: From left to right: Original image, fancy colored classified image and regularized classification map with radius equal to 3 pixels.

4.5 Feature extraction

As described in the OTB Software Guide, the term *Feature Extraction* refers to techniques aiming at extracting added value information from images. These extracted items named *features* can be local statistical moments, edges, radiometric indices, morphological and textural properties. For example,

such features can be used as input data for other image processing methods like *Segmentation* and *Classification*.

4.5.1 Local statistics extraction

This application computes the 4 local statistical moments on every pixel in the selected channel of the input image, over a specified neighborhood. The output image is multi band with one statistical moment (feature) per band. Thus, the 4 output features are the Mean, the Variance, the Skewness and the Kurtosis. They are provided in this exact order in the output image.

The *LocalStatisticExtraction* application has the following input parameters:

- `-in` the input image to compute the features on
- `-channel` the selected channel index in the input image to be processed (default value is 1)
- `-radius` the computational window radius (default value is 3 pixels)
- `-out` the output image containing the local statistical moments

The application can be used like this:

```
otbcli_LocalStatisticExtraction  -in        InputImage
                                 -channel   1
                                 -radius    3
                                 -out       OutputImage
```

4.5.2 Edge extraction

This application Computes edge features on every pixel in the selected channel of the input image.

The *EdgeExtraction* application has the following input parameters:

- `-in` the input image to compute the features on
- `-channel` the selected channel index in the input image to be processed (default value is 1)
- `-filter` the choice of edge detection method (gradient/sobel/touzi) (default value is gradient)
- `(-filter.touzi.xradius)` the X Radius of the Touzi processing neighborhood (only if filter==touzi) (default value is 1 pixel)

- (-filter.touzi.yradius) the Y Radius of the Touzi processing neighborhood (only if filter==touzi) (default value is 1 pixel)

- -out the output mono band image containing the edge features

The application can be used like this:

```
otbcli_EdgeExtraction  -in        InputImage
                       -channel   1
                       -filter    sobel
                       -out       OutputImage
```

or like this if filter==touzi:

```
otbcli_EdgeExtraction  -in                    InputImage
                       -channel               1
                       -filter                touzi
                       -filter.touzi.xradius  2
                       -filter.touzi.yradius  2
                       -out                   OutputImage
```

4.5.3 Radiometric indices extraction

This application computes radiometric indices using the channels of the input image. The output is a multi band image into which each channel is one of the selected indices.

The *RadiometricIndices* application has the following input parameters:

- -in the input image to compute the features on
- -out the output image containing the radiometric indices
- -channels.blue the Blue channel index in the input image (default value is 1)
- -channels.green the Green channel index in the input image (default value is 1)
- -channels.red the Red channel index in the input image (default value is 1)
- -channels.nir the Near Infrared channel index in the input image (default value is 1)
- -channels.mir the Mid-Infrared channel index in the input image (default value is 1)
- -list the list of available radiometric indices (default value is Vegetation:NDVI)

The available radiometric indices to be listed into -list with their relevant channels in brackets are:

```
Vegetation:NDVI - Normalized difference vegetation index (Red, NIR)
Vegetation:TNDVI - Transformed normalized difference vegetation index (Red, NIR)
Vegetation:RVI - Ratio vegetation index (Red, NIR)
Vegetation:SAVI - Soil adjusted vegetation index (Red, NIR)
Vegetation:TSAVI - Transformed soil adjusted vegetation index (Red, NIR)
Vegetation:MSAVI - Modified soil adjusted vegetation index (Red, NIR)
Vegetation:MSAVI2 - Modified soil adjusted vegetation index 2 (Red, NIR)
Vegetation:GEMI - Global environment monitoring index (Red, NIR)
Vegetation:IPVI - Infrared percentage vegetation index (Red, NIR)

Water:NDWI - Normalized difference water index (Gao 1996) (NIR, MIR)
Water:NDWI2 - Normalized difference water index (Mc Feeters 1996) (Green, NIR)
Water:MNDWI - Modified normalized difference water index (Xu 2006) (Green, MIR)
Water:NDPI - Normalized difference pond index (Lacaux et al.) (MIR, Green)
Water:NDTI - Normalized difference turbidity index (Lacaux et al.) (Red, Green)

Soil:RI - Redness index (Red, Green)
Soil:CI - Color index (Red, Green)
Soil:BI - Brightness index (Red, Green)
Soil:BI2 - Brightness index 2 (NIR, Red, Green)
```

The application can be used like this, which leads to an output image with 3 bands, respectively with the Vegetation:NDVI, Vegetation:RVI and Vegetation:IPVI radiometric indices in this exact order:

```
otbcli_RadiometricIndices -in             InputImage
                          -out            OutputImage
                          -channels.red   3
                          -channels.green 2
                          -channels.nir   4
                          -list           Vegetation:NDVI Vegetation:RVI
                                          Vegetation:IPVI
```

or like this, which leads to a single band output image with the Water:NDWI2 radiometric indice:

```
otbcli_RadiometricIndices -in             InputImage
                          -out            OutputImage
                          -channels.red   3
                          -channels.green 2
                          -channels.nir   4
                          -list           Water:NDWI2
```

4.5.4 Morphological features extraction

Morphological features can be highlighted by using image filters based on mathematical morphology either on binary or gray scale images.

Binary morphological operations

This application performs binary morphological operations (dilation, erosion, opening and closing) on a mono band image with a specific structuring element (a ball or a cross) having one radius along X and another one along Y. NB: the cross shaped structuring element has a fixed radius equal to 1 pixel in both X and Y directions.

The *BinaryMorphologicalOperation* application has the following input parameters:

- `-in` the input image to be filtered
- `-channel` the selected channel index in the input image to be processed (default value is 1)
- `-structype` the choice of the structuring element type (ball/cross) (default value is ball)
- `(-structype.ball.xradius)` the ball structuring element X Radius (only if structype==ball) (default value is 5 pixels)
- `(-structype.ball.yradius)` the ball structuring element Y Radius (only if structype==ball) (default value is 5 pixels)
- `-filter` the choice of the morphological operation (dilate/erode/opening/closing) (default value is dilate)
- `(-filter.dilate.foreval)` the foreground value for the dilation (idem for filter.erode/opening/closing) (default value is 1)
- `(-filter.dilate.backval)` the background value for the dilation (idem for filter.erode/opening/closing) (default value is 0)
- `-out` the output filtered image

The application can be used like this:

```
otbcli_BinaryMorphologicalOperation  -in                     InputImage
                                     -channel                1
                                     -structype              ball
                                     -structype.ball.xradius 10
                                     -structype.ball.yradius 5
                                     -filter                 opening
                                     -filter.opening.foreval 1.0
                                     -filter.opening.backval 0.0
                                     -out                    OutputImage
```

Gray scale morphological operations

This application performs morphological operations (dilation, erosion, opening and closing) on a gray scale mono band image with a specific structuring element (a ball or a cross) having one radius along X and another one along Y. NB: the cross shaped structuring element has a fixed radius equal to 1 pixel in both X and Y directions.

The *GrayScaleMorphologicalOperation* application has the following input parameters:

- `-in` the input image to be filtered
- `-channel` the selected channel index in the input image to be processed (default value is 1)
- `-structype` the choice of the structuring element type (ball/cross) (default value is ball)
- `(-structype.ball.xradius)` the ball structuring element X Radius (only if structype==ball) (default value is 5 pixels)
- `(-structype.ball.yradius)` the ball structuring element Y Radius (only if structype==ball) (default value is 5 pixels)
- `-filter` the choice of the morphological operation (dilate/erode/opening/closing) (default value is dilate)
- `-out` the output filtered image

The application can be used like this:

```
otbcli_GrayScaleMorphologicalOperation  -in                       InputImage
                                        -channel                  1
                                        -structype                ball
                                        -structype.ball.xradius   10
                                        -structype.ball.yradius   5
                                        -filter                   opening
                                        -out                      OutputImage
```

4.5.5 Textural features extraction

Texture features can be extracted with the help of image filters based on texture analysis methods like Haralick and structural feature set (SFS).

Haralick texture features

This application computes Haralick, advanced and higher order texture features on every pixel in the selected channel of the input image. The output image is multi band with a feature per band.

The *HaralickTextureExtraction* application has the following input parameters:

- `-in` the input image to compute the features on
- `-channel` the selected channel index in the input image to be processed (default value is 1)
- `-texture` the texture set selection [simple/advanced/higher] (default value is simple)
- `-parameters.min` the input image minimum (default value is 0)
- `-parameters.max` the input image maximum (default value is 255)
- `-parameters.xrad` the X Radius of the processing neighborhood (default value is 2 pixels)
- `-parameters.yrad` the Y Radius of the processing neighborhood (default value is 2 pixels)
- `-parameters.xoff` the ΔX Offset for the co-occurrence computation (default value is 1 pixel)
- `-parameters.yoff` the ΔY Offset for the co-occurrence computation (default value is 1 pixel)
- `-parameters.nbbin` the number of bin per axis for histogram generation (default value is 8)
- `-out` the output multi band image containing the selected texture features (one feature per band)

The available values for -texture with their relevant features are:

- `-texture=simple`: In this case, 8 local Haralick textures features will be processed. The 8 output image channels are: Energy, Entropy, Correlation, Inverse Difference Moment, Inertia, Cluster Shade, Cluster Prominence and Haralick Correlation. They are provided in this exact order in the output image. Thus, this application computes the following Haralick textures over a sliding windows with user defined radius: (where $g(i,j)$ is the element in cell i, j of a normalized Gray Level Co-occurrence Matrix (GLCM)):

  "Energy" $= f_1 = \sum_{i,j} g(i,j)^2$

  "Entropy" $= f_2 = -\sum_{i,j} g(i,j)\log_2 g(i,j)$, or 0 if $g(i,j) = 0$

  "Correlation" $= f_3 = \sum_{i,j} \frac{(i-\mu)(j-\mu)g(i,j)}{\sigma^2}$

  "Inverse Difference Moment" $= f_4 = \sum_{i,j} \frac{1}{1+(i-j)^2} g(i,j)$

  "Inertia" $= f_5 = \sum_{i,j} (i-j)^2 g(i,j)$ (sometimes called "contrast")

  "Cluster Shade" $= f_6 = \sum_{i,j} ((i-\mu)+(j-\mu))^3 g(i,j)$

  "Cluster Prominence" $= f_7 = \sum_{i,j} ((i-\mu)+(j-\mu))^4 g(i,j)$

  "Haralick's Correlation" $= f_8 = \frac{\sum_{i,j}(i,j)g(i,j)-\mu_t^2}{\sigma_t^2}$ where μ_t and σ_t are the mean and standard deviation of the row (or column, due to symmetry) sums.

Above, μ = (weighted pixel average) $= \sum_{i,j} i \cdot g(i,j) = \sum_{i,j} j \cdot g(i,j)$ (due to matrix symmetry), and σ = (weighted pixel variance) $= \sum_{i,j} (i-\mu)^2 \cdot g(i,j) = \sum_{i,j} (j-\mu)^2 \cdot g(i,j)$ (due to matrix symmetry).

- `-texture=advanced`: In this case, 9 advanced texture features will be processed. The 9 output image channels are: Mean, Variance, Sum Average, Sum Variance, Sum Entropy, Difference of Entropies, Difference of Variances, IC1 and IC2. They are provided in this exact order in the output image.

- `-texture=higher`: In this case, 11 local higher order statistics texture coefficients based on the grey level run-length matrix will be processed. The 11 output image channels are: Short Run Emphasis, Long Run Emphasis, Grey-Level Nonuniformity, Run Length Nonuniformity, Run Percentage, Low Grey-Level Run Emphasis, High Grey-Level Run Emphasis, Short Run Low Grey-Level Emphasis, Short Run High Grey-Level Emphasis, Long Run Low Grey-Level Emphasis and Long Run High Grey-Level Emphasis. They are provided in this exact order in the output image. Thus, this application computes the following Haralick textures over a sliding window with user defined radius: (where $p(i,j)$ is the element in cell i, j of a normalized Run Length Matrix, n_r is the total number of runs and n_p is the total number of pixels):

  "Short Run Emphasis" $= SRE = \frac{1}{n_r} \sum_{i,j} \frac{p(i,j)}{j^2}$

  "Long Run Emphasis" $= LRE = \frac{1}{n_r} \sum_{i,j} p(i,j) * j^2$

  "Grey-Level Nonuniformity" $= GLN = \frac{1}{n_r} \sum_i \left(\sum_j p(i,j) \right)^2$

  "Run Length Nonuniformity" $= RLN = \frac{1}{n_r} \sum_j \left(\sum_i p(i,j) \right)^2$

  "Run Percentage" $= RP = \frac{n_r}{n_p}$

  "Low Grey-Level Run Emphasis" $= LGRE = \frac{1}{n_r} \sum_{i,j} \frac{p(i,j)}{i^2}$

  "High Grey-Level Run Emphasis" $= HGRE = \frac{1}{n_r} \sum_{i,j} p(i,j) * i^2$

  "Short Run Low Grey-Level Emphasis" $= SRLGE = \frac{1}{n_r} \sum_{i,j} \frac{p(i,j)}{i^2 j^2}$

  "Short Run High Grey-Level Emphasis" $= SRHGE = \frac{1}{n_r} \sum_{i,j} \frac{p(i,j) * i^2}{j^2}$

  "Long Run Low Grey-Level Emphasis" $= LRLGE = \frac{1}{n_r} \sum_{i,j} \frac{p(i,j) * j^2}{i^2}$

  "Long Run High Grey-Level Emphasis" $= LRHGE = \frac{1}{n_r} \sum_{i,j} p(i,j) i^2 j^2$

The application can be used like this:

```
otbcli_HaralickTextureExtraction  -in             InputImage
                                  -channel        1
                                  -texture        simple
                                  -parameters.min 0
```

```
                                -parameters.max 255
                                -out            OutputImage
```

SFS texture extraction

This application computes Structural Feature Set textures on every pixel in the selected channel of the input image. The output image is multi band with a feature per band. The 6 output texture features are SFS'Length, SFS'Width, SFS'PSI, SFS'W-Mean, SFS'Ratio and SFS'SD. They are provided in this exact order in the output image.

It is based on line direction estimation and described in the following publication. Please refer to Xin Huang, Liangpei Zhang and Pingxiang Li publication, Classification and Extraction of Spatial Features in Urban Areas Using High-Resolution Multispectral Imagery. IEEE Geoscience and Remote Sensing Letters, vol. 4, n. 2, 2007, pp 260-264.

The texture is computed for each pixel using its neighborhood. User can set the spatial threshold that is the max line length, the spectral threshold that is the max difference authorized between a pixel of the line and the center pixel of the current neighborhood. The adjustement constant alpha and the ratio Maximum Consideration Number, which describes the shape contour around the central pixel, are used to compute the $w - mean$ value.

The *SFSTextureExtraction* application has the following input parameters:

- `-in` the input image to compute the features on
- `-channel` the selected channel index in the input image to be processed (default value is 1)
- `-parameters.spethre` the spectral threshold (default value is 50)
- `-parameters.spathre` the spatial threshold (default value is 100 pixels)
- `-parameters.nbdir` the number of directions (default value is 20)
- `-parameters.alpha` the alpha value (default value is 1)
- `-parameters.maxcons` the ratio Maximum Consideration Number (default value is 5)
- `-out` the output multi band image containing the selected texture features (one feature per band)

The application can be used like this:

```
otbcli_SFSTextureExtraction -in             InputImage
                            -channel        1
                            -out            OutputImage
```

4.6 Stereoscopic reconstruction from VHR optical images pair

This section describes how to convert pair of stereo images into elevation information.

The standard problem of terrain reconstruction with available **OTB Applications** contains the following steps:

- Estimation of displacements grids for epipolar geometry transformation
- Epipolar resampling of the image pair using those grids
- Dense disparity map estimation
- Projection of the disparities on a Digital Surface Model (DSM)

Let's go to the third dimension!

4.6.1 Estimate epipolar geometry transformation

The aim of this application is to generate resampled grids to transform images in epipolar geometry. Epipolar geometry is the geometry of stereo vision (see here). The operation of stereo rectification determines transformations to apply to each image such that pairs of conjugate epipolar lines become collinear, parallel to one of the image axes and aligned. In this geometry, the objects present on a given row of the left image are also located on the same line in the right image.

Applying this transformation reduces the problem of elevation (or stereo correspondences determination) to a 1-D problem. We have two images image1 and image2 over the same area (the stereo pair) and we assume that we know the localization functions (forward and inverse) associated for each of these images.

The forward function allows to go from the image referential to the geographic referential:

$$(long, lat) = f_{image1}^{forward}(i, j, h) \tag{4.2}$$

where h is the elevation hypothesis, (i, j) are the pixel coordinates in image1 and (long,lat) are geographic coordinates. As you can imagine, the inverse function allows to go from geographic coordinates to the image geometry.

For the second image, in that case, the expression of the inverse function is:

$$(long, lat, h) = f_{image2}^{inverse}(i, j) \tag{4.3}$$

Using jointly the forward and inverse functions from the image pair, we can construct a co-localization function $f_{image1 \to image2}$ between the position of a pixel in the first and its position in the second one:

$$(i_{image2}, j_{image2}) = f_{image1 \to image2}(i_{image1}, j_{image1}, h) \tag{4.4}$$

The expression of this function is:

$$f_{image1 \to image2}(i_{image1}, j_{image1}, h) = f^{inverse}_{image2} f^{forward}_{image1}((i_{image1}, j_{image1}), h) \tag{4.5}$$

The expression is not really important, what we need to understand is that if we are able to determine for a given pixel in image1 the corresponding pixel in image2, as we know the expression of the co-localization function between both images, we can determine by identification the information about the elevation (variable h in the equation)!

We now have the mathematical basis to understand how 3-D information can be extracted by examination of the relative positions of objects in the two 2-D epipolar images.

The construction of the two epipolar grids is a little bit more complicated in the case of VHR optical images.That is because most of passive remote sensing from space use a push-broom sensor, which corresponds to a line of sensors arranged perpendicularly to the flight direction of the spacecraft. This acquisition configuration implies a slightly different strategy for stereo-rectification (see here).

We will now explain how to use the *StereoRectificationGridGenerator* application to produce two images which are **deformation grids** to resample the two images in epipolar geometry.

```
otbcli_StereoRectificationGridGenerator -io.inleft image1.tif
                                        -io.inright image2.tif
                                        -epi.elevation.avg.value 50
                                        -epi.step 5
                                        -io.outimage1 outimage1_grid.tif
                                        -io.outright outimage1_grid.tif
```

The application estimates the displacement to apply to each pixel in both input images to obtain epipolar geometry.The application accept a ‘step’ parameter to estimate displacements on a coarser grid. Here we estimate the displacements every 10 pixels. This is because in most cases with a pair of VHR and a small angle between the two images, this grid is very smooth. Moreover, the implementation is not *streamable* and uses potentially a lot of memory. Therefore it is generally a good idea to estimate the displacement grid at a coarser resolution.

The application outputs the size of the output images in epipolar geometry. **Note these values**, we will use them in the next step to resample the two images in epipolar geometry.

In our case, we have:

```
Output parameters value:
epi.rectsizex: 4462
epi.rectsizey: 2951
epi.baseline:  0.2094
```

The epi.baseline parameter provides the mean value (in $pixels.meters^{-1}$) of the baseline to sensor altitude ratio. It can be used to convert disparities to physical elevation, since a disparity of this value will correspond to an elevation offset of one meter with respect to the mean elevation.

we can now move forward to the resampling in epipolar geometry.

4.6.2 Resample images in epipolar geometry

The former application generates two grids of displacements. The *GridBasedImageResampling* allows to resample the two input images in the epipolar geometry using these grids. These grids are intermediary results not really useful on their own in most cases. This second step *only* consists in applying the transformation and resample both images. This application can obviously be used in lot of other contexts.

The two commands to generate epipolar images are:

```
otbcli_GridBasedImageResampling -io.in image1.tif
                                -io.out image1_epipolar.tif
                                -grid.in outimage1_grid.tif
                                -out.sizex 4462
                                -out.sizey 2951

otbcli_GridBasedImageResampling -io.in image2.tif
                                -io.out image2_epipolar.tif
                                -grid.in outimage2_grid.tif
                                -out.sizex 4462
                                -out.sizey 2951
```

As you can see, we set *sizex* and *sizey* parameters using output values given by the *StereoRectificationGridGenerator* application to set the size of the output epipolar images.

We obtain two images in epipolar geometry, as shown in figure 4.17. Note that the application allows to resample only a part of the image using the *-out.ulx* and *-out.uly* parameters.

4.6.3 Disparity estimation: Block matching along epipolar lines

Finally, we can begin the stereo correspondences lookup process!

Things are becoming a little bit more complex but do not worry. First, we will describe the power of the *BlockMatching* application.

The resampling of our images in epipolar geometry allows us to constrain the search along a 1-dimensional line as opposed to both dimensions, but what is even more important is that the disparities along the lines, i.e. the offset along the lines measured by the block-matching process can be directly linked to the local elevation

Figure 4.17: Extract of resample image1 and image2 in epipolar geometry over Pyramids of Cheops. ©*CNES 2012*

An almost complete spectrum of stereo correspondence algorithms has been published and it is still augmented at a significant rate! See for example . The **Orfeo ToolBox** implements different strategies for block matching:

- Sum of Square Distances block-matching (SSD)
- Normalized Cross-Correlation (NCC)
- Lp pseudo-norm (LP)

An other important parameter (mandatory in the application!) is the range of disparities. In theory, the block matching can perform a blind exploration and search for a infinite range of disparities between the stereo pair. We need now to evaluate a range of disparities where the block matching will be performed (in the general case from the deepest point on Earth, the Challenger Deep. to the Everest summit!)

We deliberately exaggerated but you can imagine that without a smaller range the block matching algorithm can take a lot of time. That is why these parameters are mandatory for the application and as a consequence we need to estimate them manually. This is pretty simple using the two epipolar images.

In our case, we take one point on a *flat* area. The image coordinate in $image_1$ is $[1970, 1525]$ and in $image_2$ is $[1970, 1526]$. We then select a second point on a higher region (in our case a point near the top of the Pyramid of Cheops!). The image coordinate of this pixel in $image_1$ is $[1661, 1299]$ and in $image_2$ is $[1633, 1300]$. So you see for the horizontal exploration, we must set the minimum value lower than -30 (the convention for the sign of the disparity range is from image1 to image2).

Note that this estimation can be simplified using an external DEM in the *StereoRectificationGrid-Generator* application. Regarding the vertical disparity, in the first step we said that we reduced the problem of 3-D extraction to a 1-D problem, but this is not completely true in general cases. There might be small disparities in the vertical direction which are due to parallax errors (i.e. epipolar

lines exhibit a small shift in the vertical direction, around 1 pixel). In fact, the exploration is typically smaller along the vertical direction of disparities than along the horizontal one. You can also estimate them on the epipolar pair (in our case we use a range of -1 to 1).

One more time, take care of the sign of this minimum and this maximum for disparities (always from image1 to image2).

The command line for the *BlockMatching* application is :

```
otbcli_BlockMatching -io.inleft image1_epipolar.tif
                     -io.inright image2_epipolar.tif
                     -io.out disparity_map_ncc.tif
                     -bm.minhd -45
                     -bm.maxhd 5
                     -bm.minvd 1
                     -bm.maxvd 1
                     -mask.inleft image1_epipolar_mask.tif
                     -mask.inright image2_epipolar_mask.tif
                     -io.outmetric 1
                     -bm.metric ncc
                     -bm.subpixel dichotomy
                     -bm.medianfilter.radius 5
                     -bm.medianfilter.incoherence 2.0
```

The application creates by default a two bands image : the horizontal and vertical disparities.

The *BlockMatching* application gives access to a lot of other powerful functionalities to improve the quality of the output disparity map.

Here are a few of these functionalities:

- -io.outmetric: if the optimal metric values image is activated, it will be concatenated to the output image (which will then have three bands: horizontal disparity, vertical disparity and metric value)
- -bm.subpixel: Perform sub-pixel estimation of disparities
- -mask.inleft and -mask.inright: you can specify a no-data value which will discard pixels with this value (for example the epipolar geometry can generate large part of images with black pixels) This mask can be easily generated using the *BandMath* application:

  ```
  otbcli_BandMath -il image1_epipolar.tif
                  -out image1_epipolar_mask.tif
                  -exp "if(im1b1<=0,0,255)"

  otbcli_BandMath -il image2_epipolar.tif
  ```

```
-out image2_epipolar_mask.tif
-exp "if(im1b1<=0,0,255)"
```

- -mask.variancet : The block matching algorithm has difficulties to find matches on uniform areas. We can use the variance threshold to discard those regions and speed-up computation time.
- -bm.medianfilter.radius 5 and -bm.medianfilter.incoherence 2.0: Applies a median filter to the disparity map. The median filter belongs to the family of nonlinear filters. It is used to smooth an image without being biased by outliers or shot noise. The radius corresponds to the neighbourhood where the median value is computed. A detection of incoherence between the input disparity map and the median-filtered one is performed (a pixel corresponds to an incoherence if the absolute value of the difference between the pixel value in the disparity map and in the median image is higher than the incoherence threshold, whose default value is 1). Both parameters must be defined in the application to activate the filter.

Of course all these parameters can be combined to improve the disparity map.

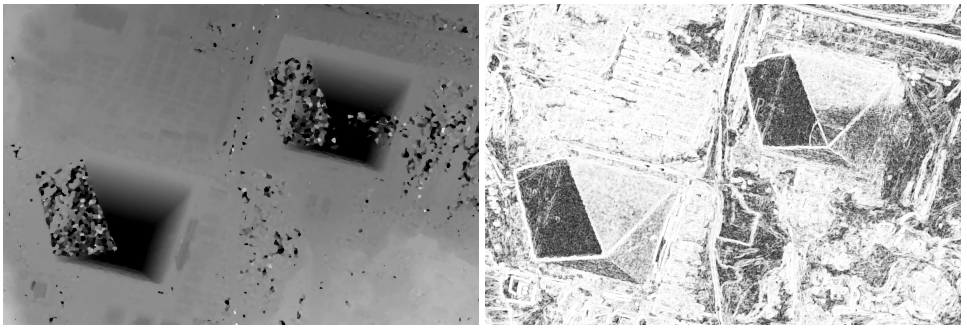


Figure 4.18: Horizontal disparity and optimal metric map

4.6.4 From disparity to Digital Surface Model

Using the previous application, we evaluated disparities between images. The next (and last!) step is now to transform the disparity map into an elevation information to produce an elevation map. It uses as input the disparity maps (horizontal and vertical) to produce a Digital Surface Model (DSM) with a regular sampling. The elevation values is computed from the triangulation of the ”left-right” pairs of matched pixels. When several elevations are available on a DSM cell, the highest one is kept.

First, an important point is that it is often a good idea to rework the disparity map given by the *BlockMatching* application to only keep relevant disparities. For this purpose, we can use the output optimal metric image and filter disparities with respect to this value.

For example, if we used Normalized Cross-Correlation (NCC), we can keep only disparities where optimal metric value is superior to 0.9. Disparities below this value can be consider as inaccurate and will not be used to compute elevation information (the *-io.mask* parameter can be used for this purpose).

This filtering can be easily done with **OTB Applications**.

We first use the *BandMath* application to filter disparities according to their optimal metric value:

```
otbcli_BandMath -il disparity_map_ncc.tif
                -out thres_hdisparity.tif uint8
                -exp "if(im1b3>0.9,255,0)"
```

Then, we concatenate thresholded disparities using the *ConcatenateImages*:

```
otbcli_ConcatenateImages -il thres_hdisparity.tif thres_vdisparity.tif
                         -out thres_hvdisparity.tif
```

Now, we can use the *DisparityMapToElevationMap* application to compute the elevation map from the filtered disparity maps.

```
otbcli_DisparityMapToElevationMap -io.in disparity_map_ncc.tif
                                  -io.left image1.tif
                                  -io.right image2.tif
                                  -io.lgrid outimage1_pyramid.tif
                                  -io.rgrid outimage2_pyramid.tif
                                  -io.mask thres_hdisparity.tif
                                  -io.out disparity_map_ssd_to_elevation.tif
                                  -hmin 10
                                  -hmax 400
                                  -elev.default 50
```

It produces the elevation map projected in WGS84 (EPSG code:4326) over the ground area covered by the stereo pair. Pixels values are expressed in meters.

This is it ! Figure 4.19 shows the output DEM from the Cheops pair.

4.6.5 One application to rule them all in multi stereo framework scheme

An application has been added to fuse one or multiple stereo reconstruction(s) using all in one approach : *StereoFramework*. It computes the DSM from one or several stereo pair. First of all the user have to choose his input data and defines stereo couples using *-input.co* string parameter. This parameter use the following formatting convention " $index_0$ $index_1$, $index_2$ $index_3$, ...", which will

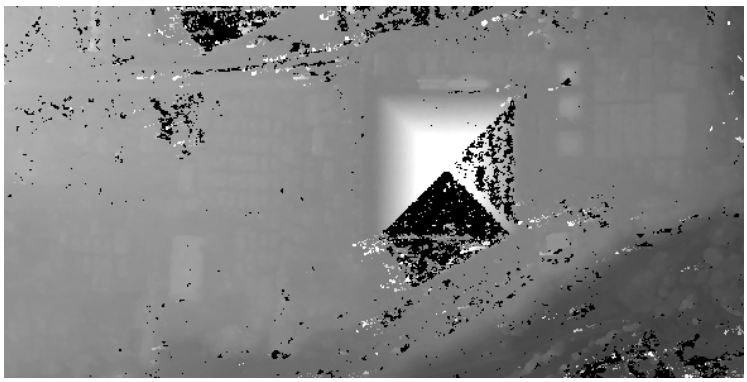


Figure 4.19: Extract of the elevation map over Pyramids of Cheops.

create a first couple with image $index_0$ and $index_1$, a second with image $index_1$ and $index_2$, and so on. If left blank images are processed by pairs (which is equivalent as using " 0 1,2 3,4 5 " ...). In addition to the usual elevation and projection parameters, main parameters have been splitted in groups detailled below:

Output : output parameters : DSM resolution, NoData value, Cell Fusion method,

- `-output.map` : output projection map selection.
- `-output.res` : Spatial Sampling Distance of the output DSM in meters
- `-output.nodata` : DSM empty cells are filled with this float value (-32768 by default)
- `-output.fusionmethod` : Choice of fusion strategy in each DSM cell (max, min, mean, acc)
- `-output.out` : Output DSM
- `-output.mode` : Output DSM extent choice

Stereorect : Direct and inverse stereorectification grid subsampling parameters

- `-stereorect.fwdgridstep` : Step of the direct deformation grid (in pixels)
- `-stereorect.invgridssrate` : Sub-sampling of the inverse epipolar grid

BM : Block Matching parameters.

- `-bm.metric` : Block-matching metric choice (robust SSD, SSD, NCC, Lp Norm)
- `-bm.radius` : Radius of blocks for matching filter (in pixels, 2 by default)
- `-bm.minhoffset` : Minimum altitude below the selected elevation source (in meters, -20.0 by default)
- `-bm.maxhoffset` : Maximum altitude above the selected elevation source (in meters, 20.0 by default)

Postproc : Post-Processing parameters

- `-postproc.bij` : use bijection consistency. Right to Left correlation is computed to validate Left to Right disparities. If bijection is not found pixel is rejected
- `-postproc.med` : use median disparities filtering (disabled by default)
- `-postproc.metrict` : use block matching metric output to discard pixels with low correlation value (disabled by default, float value)");

Mask : Compute optional intermediate masks.

- `-mask.left` : Mask for left input image (must have the same size for all couples)
- `-mask.right` : Mask for right input image (must have the same size for all couples)
- `-mask.variancet` : This parameter allows to discard pixels whose local variance is too small. The size of the neighborhood is given by the radius parameter. (disabledd by default)

4.6.6 Stereo reconstruction good practices

The parameters `-bm.minhoffset` and `-bm.maxhoffset` are used inside the application to derive the minimum and maximum horizontal disparity exploration, so they have a critical impact on computation time. It is advised to choose an elevation source that is not too far from the DSM you want to produce (for instance, an SRTM elevation model). Therefore, the altitude from your elevation source will be already taken into account in the epipolar geometry and the disparities will reveal the elevation offsets (such as buildings). It allows you to use a smaller exploration range along the elevation axis, causing a smaller exploration along horizontal disparities and faster computation.

`-stereorect.fwdgridstep` and `-stereorect.invgridssrate` have also a deep impact in time consumption, thus they have to be carefully chosen in case of large image processing.

To reduce time consumption it would be useful to crop all sensor images to the same extent. The easiest way to do that is to choose an image as reference, and then apply *ExtractROI* application on the other sensor images using the fit mode `-mode.fit` option.

4.6.7 Algorithm outline

The following algorithms are used in the application: For each sensor pair

- Compute the epipolar deformation grids from the stereo pair (direct and inverse)
- Resample into epipolar geometry with BCO interpolator
- Create masks for each epipolar image : remove black borders and resample input masks
- Compute horizontal disparities with a block matching algorithm

- Refing Disparities to sub-pixel precision with a dichotomy algorithm
- Apply an optional Median filter
- Filter disparities based on the correlation score (optional) and exploration bounds
- Translate disparities in sensor geometry
- Convert disparity map to 3D map

Then fuse all 3D maps to produce DSM with desired geographic or cartographic projection and parametrizable extent.

CHAPTER

FIVE

Applications Reference Documentation

This chapter is the reference documentation for applications delivered with **Orfeo ToolBox**. It provides detailed description of the application behaviour and parameters, as well as python and bash snippets to use them applications. For a general introduction on applications delivered with **Orfeo ToolBox**, please read chapter1, page 1.

5.1 Image Manipulation

5.1.1 Color Mapping

Maps an input label image to 8-bits RGB using look-up tables.

Detailed description

This application allows to map a label image to a 8-bits RGB image (in both ways) using different methods.
-The custom method allows to use a custom look-up table. The look-up table is loaded from a text file where each line describes an entry. The typical use of this method is to colorise a classification map.
-The continuous method allows to map a range of values in a scalar input image to a colored image using continuous look-up table, in order to enhance image interpretation. Several look-up tables can been chosen with different color ranges.
-The optimal method computes an optimal look-up table. When processing a segmentation label image (label to color), the color difference between adjacent segmented regions is maximized. When processing an unknown color image (color to label), all the present colors are mapped to a continuous label list.
- The support image method uses a color support image to associate an average color to each region.

Parameters

This section describes in details the parameters available for this application. Table 5.1, page 93 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `ColorMapping`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`out`	Output image	Output Image
`ram`	Int	Available RAM (Mb)
`op`	Choices	Operation
`op labeltocolor`	*Choice*	Label to color
`op colortolabel`	*Choice*	Color to label
`op.colortolabel.notfound`	Int	Not Found Label
`method`	Choices	Color mapping method
`method custom`	*Choice*	Color mapping with custom labeled look-up table
`method continuous`	*Choice*	Color mapping with continuous look-up table
`method optimal`	*Choice*	Compute an optimized look-up table
`method image`	*Choice*	Color mapping with look-up table calculated on support image
`method.custom.lut`	Input File name	Look-up table file
`method.continuous.lut`	Choices	Look-up tables
`method.continuous.lut red`	*Choice*	Red
`method.continuous.lut green`	*Choice*	Green
`method.continuous.lut blue`	*Choice*	Blue
`method.continuous.lut grey`	*Choice*	Grey
`method.continuous.lut hot`	*Choice*	Hot
`method.continuous.lut cool`	*Choice*	Cool
`method.continuous.lut spring`	*Choice*	Spring
`method.continuous.lut summer`	*Choice*	Summer
`method.continuous.lut autumn`	*Choice*	Autumn
`method.continuous.lut winter`	*Choice*	Winter
`method.continuous.lut copper`	*Choice*	Copper
`method.continuous.lut jet`	*Choice*	Jet
`method.continuous.lut hsv`	*Choice*	HSV
`method.continuous.lut overunder`	*Choice*	OverUnder
`method.continuous.lut relief`	*Choice*	Relief
`method.continuous.min`	Float	Mapping range lower value
`method.continuous.max`	Float	Mapping range higher value
`method.optimal.background`	Int	Background label
`method.image.in`	Input image	Support Image
`method.image.nodatavalue`	Float	NoData value
`method.image.low`	Int	lower quantile
`method.image.up`	Int	upper quantile
.../...		

Parameter key	Parameter type	Parameter description
.../...		
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.1: Parameters table for Color Mapping.

Input Image Input image filename

Output Image Output image filename

Available RAM (Mb) Available memory for processing (in MB)

Operation Selection of the operation to execute (default is : label to color). Available choices are:

- **Label to color**
- **Color to label**
  - **Not Found Label:** Label to use for unknown colors.

Color mapping method Selection of color mapping methods and their parameters. Available choices are:

- **Color mapping with custom labeled look-up table**: Apply a user-defined look-up table to a labeled image. Look-up table is loaded from a text file.
  - **Look-up table file:** An ASCII file containing the look-up table
    with one color per line
    (for instance the line '1 255 0 0' means that all pixels with label 1 will be replaced by RGB color 255 0 0)
    Lines beginning with a # are ignored
- **Color mapping with continuous look-up table**: Apply a continuous look-up table to a range of input values.
  - **Look-up tables:** Available look-up tables.
  - **Mapping range lower value:** Set the lower input value of the mapping range.

  - **Mapping range higher value:** Set the higher input value of the mapping range.

- **Compute an optimized look-up table**: [label to color] Compute an optimal look-up table such that neighboring labels in a segmentation are mapped to highly contrasted colors. [color to label] Searching all the colors present in the image to compute a continuous label list

  - **Background label:** Value of the background label

- **Color mapping with look-up table calculated on support image**

  - **Support Image:** Support image filename. For each label, the LUT is calculated from the mean pixel value in the support image, over the corresponding labeled areas. First of all, the support image is normalized with extrema rejection
  - **NoData value:** NoData value for each channel of the support image, which will not be handled in the LUT estimation. If NOT checked, ALL the pixel values of the support image will be handled in the LUT estimation.
  - **lower quantile:** lower quantile for image normalization
  - **upper quantile:** upper quantile for image normalization

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_ColorMapping -in ROI_QB_MUL_1_SVN_CLASS_MULTI.png -method custom -method.custom.lut
     ROI_QB_MUL_1_SVN_CLASS_MULTI_PNG_ColorTable.txt -out
     Colorized_ROI_QB_MUL_1_SVN_CLASS_MULTI.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the ColorMapping application
ColorMapping = otbApplication.Registry.CreateApplication("ColorMapping")

# The following lines set all the application parameters:
ColorMapping.SetParameterString("in", "ROI_QB_MUL_1_SVN_CLASS_MULTI.png")

ColorMapping.SetParameterString("method","custom")

ColorMapping.SetParameterString("method.custom.lut",
     "ROI_QB_MUL_1_SVN_CLASS_MULTI_PNG_ColorTable.txt")
```

```
ColorMapping.SetParameterString("out", "Colorized_ROI_QB_MUL_1_SVN_CLASS_MULTI.tif")

# The following line execute the application
ColorMapping.ExecuteAndWriteOutput()
```

Limitations

The segmentation optimal method does not support streaming, and thus large images. The operation color to label is not implemented for the methods continuous LUT and support image LUT. ColorMapping using support image is not threaded.

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- ImageSVMClassifier

5.1.2 Images Concatenation

Concatenate a list of images of the same size into a single multi-channel one.

Detailed description

This application performs images channels concatenation. It will walk the input image list (single or multi-channel) and generates a single multi-channel image. The channel order is the one of the list.

Parameters

This section describes in details the parameters available for this application. Table 5.2, page 96 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `ConcatenateImages`.

Parameter key	Parameter type	Parameter description
`il`	Input image list	Input images list
`out`	Output image	Output Image
`ram`	Int	Available RAM (Mb)
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.2: Parameters table for Images Concatenation.

- **Input images list:** The list of images to concatenate
- **Output Image:** The concatenated output image
- **Available RAM (Mb):** Available memory for processing (in MB)
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_ConcatenateImages -il GomaAvant.png GomaApres.png -out otbConcatenateImages.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the ConcatenateImages application
ConcatenateImages = otbApplication.Registry.CreateApplication("ConcatenateImages")

# The following lines set all the application parameters:
ConcatenateImages.SetParameterStringList("il", ['GomaAvant.png', 'GomaApres.png'])

ConcatenateImages.SetParameterString("out", "otbConcatenateImages.tif")

# The following line execute the application
ConcatenateImages.ExecuteAndWriteOutput()
```

Limitations

All input images must have the same size.

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- Rescale application, Convert

5.1.3 Image Conversion

Convert an image to a different format, eventually rescaling the data and/or changing the pixel type.

Detailed description

This application performs an image pixel type conversion (short, ushort, uchar, int, uint, float and double types are handled). The output image is written in the specified format (ie. that corresponds to the given extension).
The convertion can include a rescale using the image 2 percent minimum and maximum values. The rescale can be linear or log2.

Parameters

This section describes in details the parameters available for this application. Table 5.3, page 98 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `Convert`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input image
`type`	Choices	Rescale type
`type none`	*Choice*	None
`type linear`	*Choice*	Linear
`type log2`	*Choice*	Log2
`type.linear.gamma`	Float	Gamma correction factor
`mask`	Input image	Input mask
`hcp`	Group	Histogram Cutting Parameters
`hcp.high`	Float	High Cut Quantile
`hcp.low`	Float	Low Cut Quantile
`out`	Output image	Output Image
.../...		

Parameter key	Parameter type	Parameter description
.../...		
ram	Int	Available RAM (Mb)
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.3: Parameters table for Image Conversion.

Input image Input image

Rescale type Transfer function for the rescaling Available choices are:

- **None**
- **Linear**
  - **Gamma correction factor:** Gamma correction factor
- **Log2**

Input mask The masked pixels won't be used to adapt the dynamic (the mask must have the same dimensions as the input image)

Histogram Cutting Parameters Parameters to cut the histogram edges before rescaling

- **High Cut Quantile:** Quantiles to cut from histogram high values before computing min/max rescaling (in percent, 2 by default)
- **Low Cut Quantile:** Quantiles to cut from histogram low values before computing min/max rescaling (in percent, 2 by default)

Output Image Output image

Available RAM (Mb) Available memory for processing (in MB)

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_Convert -in QB_Toulouse_Ortho_XS.tif -out otbConvertWithScalingOutput.png uint8 -type
    linear
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the Convert application
Convert = otbApplication.Registry.CreateApplication("Convert")

# The following lines set all the application parameters:
Convert.SetParameterString("in", "QB_Toulouse_Ortho_XS.tif")

Convert.SetParameterString("out", "otbConvertWithScalingOutput.png")
Convert.SetParameterOutputImagePixelType("out", 1)

Convert.SetParameterString("type","linear")

# The following line execute the application
Convert.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- Rescale

5.1.4 Download or list SRTM tiles related to a set of images

Download or list SRTM tiles related to a set of images

Detailed description

This application allows to select the appropriate SRTM tiles that covers a list of images. It builds a list of the required tiles. Two modes are available: the first one download those tiles from the USGS SRTM3 website (http://dds.cr.usgs.gov/srtm/version2_1/SRTM3/), the second one list those tiles in a local directory. In both cases, you need to indicate the directory in which directory tiles will be download or the location of local SRTM files.

Parameters

This section describes in details the parameters available for this application. Table 5.4, page 100 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `DownloadSRTMTiles`.

Parameter key	Parameter type	Parameter description
`il`	Input image list	Input images list
`mode`	Choices	Download/List corresponding SRTM tiles.
`mode download`	*Choice*	Download
`mode list`	*Choice*	List tiles
`mode.download.outdir`	Directory	Output directory
`mode.list.indir`	Directory	Input directory
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.4: Parameters table for Download or list SRTM tiles related to a set of images.

Input images list The list of images on which you want to determine corresponding SRTM tiles.

Download/List corresponding SRTM tiles. Available choices are:

- **Download**: Download corresponding tiles on USGE server.
  - **Output directory:** Directory where zipped tiles will be save. You'll need to unzip all tile files before using them in your application.
- **List tiles**: List tiles in an existing local directory.
  - **Input directory:** Input directory where SRTM tiles can are located.

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_DownloadSRTMTiles -il QB_Toulouse_Ortho_XS.tif -mode list -mode.list.indir
    /home/user/srtm_dir/
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the DownloadSRTMTiles application
DownloadSRTMTiles = otbApplication.Registry.CreateApplication("DownloadSRTMTiles")

# The following lines set all the application parameters:
DownloadSRTMTiles.SetParameterStringList("il", ['QB_Toulouse_Ortho_XS.tif'])

DownloadSRTMTiles.SetParameterString("mode","list")

DownloadSRTMTiles.SetParameterString("mode.list.indir", "/home/user/srtm_dir/")

# The following line execute the application
DownloadSRTMTiles.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.1.5 Extract ROI

Extract a ROI defined by the user.

Detailed description

This application extracts a Region Of Interest with user defined size, or reference image.

Parameters

This section describes in details the parameters available for this application. Table 5.5, page 102 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `ExtractROI`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`out`	Output image	Output Image
`ram`	Int	Available RAM (Mb)
`mode`	Choices	Extraction mode
`mode standard`	*Choice*	Standard
`mode fit`	*Choice*	Fit
`mode.fit.ref`	Input image	Reference image
`mode.fit.elev`	Group	Elevation management
`mode.fit.elev.dem`	Directory	DEM directory
`mode.fit.elev.geoid`	Input File name	Geoid File
`mode.fit.elev.default`	Float	Default elevation
`startx`	Int	Start X
`starty`	Int	Start Y
`sizex`	Int	Size X
`sizey`	Int	Size Y
`cl`	List	Output Image channels
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.5: Parameters table for Extract ROI.

Input Image Input image.

Output Image Output image.

Available RAM (Mb) Available memory for processing (in MB)

Extraction mode Available choices are:

- **Standard**: In standard mode, extract is done according the coordinates entered by the user

- **Fit**: In fit mode, extract is made to best fit a reference image.
  - **Reference image:** Reference image to define the ROI
  - **Elevation management:** This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.
  - **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles
  - **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).
  - **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

Start X ROI start x position.

Start Y ROI start y position.

Size X size along x in pixels.

Size Y size along y in pixels.

Output Image channels Channels to write in the output image.

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_ExtractROI -in VegetationIndex.hd -startx 40 -starty 250 -sizex 150 -sizey 150 -out
     ExtractROI.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the ExtractROI application
ExtractROI = otbApplication.Registry.CreateApplication("ExtractROI")

# The following lines set all the application parameters:
ExtractROI.SetParameterString("in", "VegetationIndex.hd")

ExtractROI.SetParameterInt("startx", 40)

ExtractROI.SetParameterInt("starty", 250)

ExtractROI.SetParameterInt("sizex", 150)

ExtractROI.SetParameterInt("sizey", 150)

ExtractROI.SetParameterString("out", "ExtractROI.tif")

# The following line execute the application
ExtractROI.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.1.6 Multi Resolution Pyramid

Build a multi-resolution pyramid of the image.

Detailed description

This application builds a multi-resolution pyramid of the input image. User can specified the number of levels of the pyramid and the subsampling factor. To speed up the process, you can use the fast scheme option

Parameters

This section describes in details the parameters available for this application. Table 5.6, page 105 presents a summary of these parameters and the parameters keys to be used in command-line and

programming languages. Application key is `MultiResolutionPyramid`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`out`	Output image	Output Image
`ram`	Int	Available RAM (Mb)
`level`	Int	Number Of Levels
`sfactor`	Int	Subsampling factor
`vfactor`	Float	Variance factor
`fast`	Boolean	Use Fast Scheme
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.6: Parameters table for Multi Resolution Pyramid.

- **Input Image:**
- **Output Image:** will be used to get the prefix and the extension of the images to write
- **Available RAM (Mb):** Available memory for processing (in MB)
- **Number Of Levels:** Number of levels in the pyramid (default is 1).
- **Subsampling factor:** Subsampling factor between each level of the pyramid (default is 2).
- **Variance factor:** Variance factor use in smoothing. It is multiplied by the subsampling factor of each level in the pyramid (default is 0.6).
- **Use Fast Scheme:** If used, this option allows to speed-up computation by iteratively subsampling previous level of pyramid instead of processing the full input.
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_MultiResolutionPyramid -in QB_Toulouse_Ortho_XS.tif -out multiResolutionImage.tif
     -level 1 -sfactor 2 -vfactor 0.6 -fast false
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the MultiResolutionPyramid application
MultiResolutionPyramid = otbApplication.Registry.CreateApplication("MultiResolutionPyramid")

# The following lines set all the application parameters:
MultiResolutionPyramid.SetParameterString("in", "QB_Toulouse_Ortho_XS.tif")

MultiResolutionPyramid.SetParameterString("out", "multiResolutionImage.tif")

MultiResolutionPyramid.SetParameterInt("level", 1)

MultiResolutionPyramid.SetParameterInt("sfactor", 2)

MultiResolutionPyramid.SetParameterFloat("vfactor", 0.6)

MultiResolutionPyramid.SetParameterString("fast","1")

# The following line execute the application
MultiResolutionPyramid.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.1.7 Quick Look

Generates a subsampled version of an image extract

Detailed description

Generates a subsampled version of an extract of an image defined by ROIStart and ROISize.
This extract is subsampled using the ratio OR the output image Size.

Parameters

This section describes in details the parameters available for this application. Table 5.7, page 107 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `Quicklook`.

Parameter key	Parameter type	Parameter description
in	Input image	Input Image
out	Output image	Output Image
cl	List	Channel List
rox	Int	ROI Origin X
roy	Int	ROI Origin Y
rsx	Int	ROI Size X
rsy	Int	ROI Size Y
sr	Int	Sampling ratio
sx	Int	Size X
sy	Int	Size Y
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.7: Parameters table for Quick Look.

- **Input Image:** The image to read
- **Output Image:** The subsampled image
- **Channel List:** Selected channels
- **ROI Origin X:** first point of ROI in x-direction
- **ROI Origin Y:** first point of ROI in y-direction
- **ROI Size X:** size of ROI in x-direction
- **ROI Size Y:** size of ROI in y-direction
- **Sampling ratio:** Sampling Ratio, default is 2
- **Size X:** quicklook size in x-direction (used if no sampling ration is given)
- **Size Y:** quicklook size in y-direction (used if no sampling ration is given)
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_Quicklook -in qb_RoadExtract.tif -out quicklookImage.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the Quicklook application
Quicklook = otbApplication.Registry.CreateApplication("Quicklook")

# The following lines set all the application parameters:
Quicklook.SetParameterString("in", "qb_RoadExtract.tif")

Quicklook.SetParameterString("out", "quicklookImage.tif")

# The following line execute the application
Quicklook.ExecuteAndWriteOutput()
```

Limitations

This application does not provide yet the optimal way to decode coarser level of resolution from JPEG2000 images (like in Monteverdi).
Trying to subsampled huge JPEG200 image with the application will lead to poor performances for now.

Authors

This application has been written by OTB-Team.

5.1.8 Read image information

Get information about the image

Detailed description

Display information about the input image like: image size, origin, spacing, metadata, projections...

Parameters

This section describes in details the parameters available for this application. Table 5.8, page 110 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `ReadImageInfo`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`keywordlist`	Boolean	Display the OSSIM keywordlist
`outkwl`	Output File name	Write the OSSIM keywordlist to a geom file
`indexx`	Int	Start index X
`indexy`	Int	Start index Y
`sizex`	Int	Size X
`sizey`	Int	Size Y
`spacingx`	Float	Pixel Size X
`spacingy`	Float	Pixel Size Y
`originx`	Float	Image Origin X
`originy`	Float	Image Origin Y
`estimatedgroundspacingx`	Float	Estimated ground spacing X
`estimatedgroundspacingy`	Float	Estimated ground spacing Y
`numberbands`	Int	Number Of Bands
`sensor`	String	Sensor id
`id`	String	Image id
`time`	String	Acquisition time
`ullat`	Float	Upper left lattitude
`ullon`	Float	Upper left longitude
`urlat`	Float	Upper right lattitude
`urlon`	Float	Upper right longitude
`lrlat`	Float	Lower right lattitude
`lrlon`	Float	Lower right longitude
`lllat`	Float	Lower left lattitude
`lllon`	Float	Lower left longitude
`town`	String	Nearest town
`country`	String	Country
`rgb`	Group	Default RGB Display
`rgb.r`	Int	Red Band
`rgb.g`	Int	Green Band
`rgb.b`	Int	Blue Band
`projectionref`	String	Projection
`keyword`	String	Keywordlist
`gcp`	Group	Ground Control Points informations
`gcp.count`	Int	GCPs Number
`gcp.proj`	String	GCP Projection
`gcp.ids`	String list	GCPs Id
`gcp.info`	String list	GCPs Info
`gcp.imcoord`	String list	GCPs Image Coordinates
.../...		

Parameter key	Parameter type	Parameter description
.../...		
gcp.geocoord	String list	GCPs Geographic Coordinates
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.8: Parameters table for Read image information.

Input Image Input image to analyse

Display the OSSIM keywordlist Output the OSSIM keyword list. It contains metadata information (sensor model, geometry). Informations are stored in keyword list (pairs of key/value)

Write the OSSIM keywordlist to a geom file This option allows to extract the OSSIM keywordlist of the image into a geom file.

Start index X X start index

Start index Y Y start index

Size X X size (in pixels)

Size Y Y size (in pixels)

Pixel Size X Pixel size along X (in physical units)

Pixel Size Y Pixel size along Y (in physical units)

Image Origin X Origin along X

Image Origin Y Origin along Y

Estimated ground spacing X Estimated ground spacing along X (in meters).

Estimated ground spacing Y Estimated ground spacing along Y (in meters).

Number Of Bands Number of bands

Sensor id Sensor identifier

Image id Image identifier

Acquisition time Acquisition time.

Upper left lattitude Lattitude of the upper left corner.

Upper left longitude Longitude of the upper left corner.

Upper right lattitude Lattitude of the upper right corner.

Upper right longitude Longitude of the upper right corner.

Lower right lattitude Lattitude of the lower right corner.

Lower right longitude Longitude of the lower right corner.

Lower left lattitude Lattitude of the lower left corner.

Lower left longitude Longitude of the lower left corner.

Nearest town Main town near center of image

Country Country of the image

Default RGB Display This group of parameters allows to access to the default rgb composition.

- **Red Band:** Red band Number
- **Green Band:** Green band Number
- **Blue Band:** Blue band Number

Projection Projection Coordinate System

Keywordlist Image keyword list

Ground Control Points informations This group of parameters allows to access to the GCPs informations.

- **GCPs Number:** Number of GCPs
- **GCP Projection:** Projection Coordinate System for GCPs
- **GCPs Id:** GCPs identifier
- **GCPs Info:** GCPs Information
- **GCPs Image Coordinates:** GCPs Image coordinates
- **GCPs Geographic Coordinates:** GCPs Geographic Coordinates

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_ReadImageInfo -in QB_Toulouse_Ortho_XS.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the ReadImageInfo application
ReadImageInfo = otbApplication.Registry.CreateApplication("ReadImageInfo")

# The following lines set all the application parameters:
ReadImageInfo.SetParameterString("in", "QB_Toulouse_Ortho_XS.tif")

# The following line execute the application
ReadImageInfo.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.1.9 Rescale Image

Rescale the image between two given values.

Detailed description

This application scales the given image pixel intensity between two given values. By default min (resp. max) value is set to 0 (resp. 255).

Parameters

This section describes in details the parameters available for this application. Table 5.9, page 114 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `Rescale`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`out`	Output image	Output Image
`ram`	Int	Available RAM (Mb)
.../...		

Parameter key	Parameter type	Parameter description
.../...		
outmin	Float	Output min value
outmax	Float	Output max value
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.9: Parameters table for Rescale Image.

- **Input Image:** The image to scale.
- **Output Image:** The rescaled image filename.
- **Available RAM (Mb):** Available memory for processing (in MB)
- **Output min value:** Minimum value of the output image.
- **Output max value:** Maximum value of the output image.
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_Rescale -in QB_Toulouse_Ortho_PAN.tif -out rescaledImage.png uchar -outmin 0 -outmax
    255
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the Rescale application
Rescale = otbApplication.Registry.CreateApplication("Rescale")

# The following lines set all the application parameters:
Rescale.SetParameterString("in", "QB_Toulouse_Ortho_PAN.tif")

Rescale.SetParameterString("out", "rescaledImage.png")
Rescale.SetParameterOutputImagePixelType("out", 1)

Rescale.SetParameterFloat("outmin", 0)
```

```
Rescale.SetParameterFloat("outmax", 255)

# The following line execute the application
Rescale.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.1.10 Split Image

Split a N multiband image into N images

Detailed description

This application splits a N-bands image into N mono-band images. The output images filename will be generated from the output parameter. Thus if the input image has 2 channels, and the user has set an output outimage.tif, the generated images will be outimage_0.tif and outimage_1.tif

Parameters

This section describes in details the parameters available for this application. Table 5.10, page 116 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `SplitImage`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`out`	Output image	Output Image
`ram`	Int	Available RAM (Mb)
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.10: Parameters table for Split Image.

- **Input Image:** Input multiband image filename.
- **Output Image:** Output filename that will be used to get the prefix and the extension of the output images to write
- **Available RAM (Mb):** Available memory for processing (in MB)
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_SplitImage -in VegetationIndex.hd -out splittedImage.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the SplitImage application
SplitImage = otbApplication.Registry.CreateApplication("SplitImage")

# The following lines set all the application parameters:
SplitImage.SetParameterString("in", "VegetationIndex.hd")

SplitImage.SetParameterString("out", "splittedImage.tif")

# The following line execute the application
SplitImage.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.1.11 Image Tile Fusion

Fusion of an image made of several tile files.

Detailed description

Concatenate several tile files into a single image file.

Parameters

This section describes in details the parameters available for this application. Table 5.11, page 117 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `TileFusion`.

Parameter key	Parameter type	Parameter description
`il`	Input image list	Input Tile Images
`cols`	Int	Number of tile columns
`rows`	Int	Number of tile rows
`out`	Output image	Output Image
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.11: Parameters table for Image Tile Fusion.

- **Input Tile Images:** Input tiles to concatenate (in lexicographic order : (0,0) (1,0) (0,1) (1,1)).
- **Number of tile columns:** Number of columns in the tile array
- **Number of tile rows:** Number of rows in the tile array
- **Output Image:** Output entire image
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_TileFusion -il Scene_R1C1.tif Scene_R1C2.tif Scene_R2C1.tif Scene_R2C2.tif -cols 2
    -rows 2 -out EntireImage.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the TileFusion application
TileFusion = otbApplication.Registry.CreateApplication("TileFusion")

# The following lines set all the application parameters:
TileFusion.SetParameterStringList("il", ['Scene_R1C1.tif', 'Scene_R1C2.tif',
     'Scene_R2C1.tif', 'Scene_R2C2.tif'])

TileFusion.SetParameterInt("cols", 2)

TileFusion.SetParameterInt("rows", 2)

TileFusion.SetParameterString("out", "EntireImage.tif")

# The following line execute the application
TileFusion.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.2 Vector Data Manipulation

5.2.1 Concatenate

Concatenate VectorDatas

Detailed description

This application concatenates a list of VectorData to produce a unique VectorData as output.Note that the VectorDatas must be of the same type (Storing polygons only, lines only, or points only)

Parameters

This section describes in details the parameters available for this application. Table 5.12, page 119 presents a summary of these parameters and the parameters keys to be used in command-line and

programming languages. Application key is `ConcatenateVectorData`.

Parameter key	Parameter type	Parameter description
vd	Input vector data list	Input VectorDatas to concatenate
out	Output vector data	Concatenated VectorData
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.12: Parameters table for Concatenate.

- **Input VectorDatas to concatenate:** VectorData files to be concatenated in an unique VectorData
- **Concatenated VectorData:** Output conctenated VectorData
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_ConcatenateVectorData -vd ToulousePoints-examples.shp ToulouseRoad-examples.shp -out
    ConcatenatedVectorData.shp
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the ConcatenateVectorData application
ConcatenateVectorData = otbApplication.Registry.CreateApplication("ConcatenateVectorData")

# The following lines set all the application parameters:
ConcatenateVectorData.SetParameterStringList("vd", ['ToulousePoints-examples.shp',
    'ToulouseRoad-examples.shp'])

ConcatenateVectorData.SetParameterString("out", "ConcatenatedVectorData.shp")

# The following line execute the application
ConcatenateVectorData.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.2.2 Rasterization

Rasterize a vector dataset.

Detailed description

This application allows to reproject and rasterize a vector dataset. The grid of the rasterized output can be set by using a reference image, or by setting all parmeters (origin, size, spacing) by hand. In the latter case, at least the spacing (ground sampling distance) is needed (other parameters are computed automatically). The rasterized output can also be in a different projection reference system than the input dataset.
There are two rasterize mode available in the application. The first is the binary mode: it allows to render all pixels belonging to a geometry of the input dataset in the foreground color, while rendering the other in background color. The second one allows to render pixels belonging to a geometry woth respect to an attribute of this geometry. The field of the attribute to render can be set by the user. In the second mode, the background value is still used for unassociated pixels.

Parameters

This section describes in details the parameters available for this application. Table 5.13, page 121 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `Rasterization`.

Parameter key	Parameter type	Parameter description
`in`	Input File name	Input vector dataset
`out`	Output image	Ouptut image
`im`	Input image	Input reference image
`szx`	Int	Output size x
`szy`	Int	Output size y
`epsg`	Int	Output EPSG code
`orx`	Float	Output Upper-left x
`ory`	Float	Output Upper-left y
.../...		

Parameter key	Parameter type	Parameter description
.../...		
spx	Float	Spacing (GSD) x
spy	Float	Spacing (GSD) y
background	Float	Background value
mode	Choices	Rasterization mode
mode binary	*Choice*	Binary mode
mode attribute	*Choice*	Attribute burning mode
mode.binary.foreground	Float	Foreground value
mode.attribute.field	String	The attribute field to burn
ram	Int	Available RAM (Mb)
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.13: Parameters table for Rasterization.

Input vector dataset The input vector dataset to be rasterized

Ouptut image An output image containing the rasterized vector dataset

Input reference image A reference image from which to import output grid and projection reference system information.

Output size x Output size along x axis (useless if support image is given)

Output size y Output size along y axis (useless if support image is given)

Output EPSG code EPSG code for the output projection reference system (EPSG 4326 for WGS84, 32631 for UTM31N...,useless if support image is given)

Output Upper-left x Output upper-left x coordinate (useless if support image is given)

Output Upper-left y Output upper-left y coordinate (useless if support image is given)

Spacing (GSD) x Spacing (ground sampling distance) along x axis (useless if support image is given)

Spacing (GSD) y Spacing (ground sampling distance) along y axis (useless if support image is given)

Background value Default value for pixels not belonging to any geometry

Rasterization mode Choice of rasterization modes Available choices are:

- **Binary mode**: In this mode, pixels within a geometry will hold the user-defined foreground value
  - **Foreground value:** Value for pixels inside a geometry
- **Attribute burning mode**: In this mode, pixels within a geometry will hold the value of a user-defined field extracted from this geometry.
  - **The attribute field to burn:** Name of the attribute field to burn

Available RAM (Mb) Available memory for processing (in MB)

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_Rasterization -in qb_RoadExtract_classification.shp -out rasterImage.tif -spx 1. -spy
    1.
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the Rasterization application
Rasterization = otbApplication.Registry.CreateApplication("Rasterization")

# The following lines set all the application parameters:
```

```
Rasterization.SetParameterString("in", "qb_RoadExtract_classification.shp")

Rasterization.SetParameterString("out", "rasterImage.tif")

Rasterization.SetParameterFloat("spx", 1.)

Rasterization.SetParameterFloat("spy", 1.)

# The following line execute the application
Rasterization.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- For now, support of input dataset with multiple layers having different projection reference system is limited.

5.2.3 VectorData Extract ROI

Perform an extract ROI on the input vector data according to the input image extent

Detailed description

This application extracts the vector data features belonging to a region specified by the support image envelope

Parameters

This section describes in details the parameters available for this application. Table 5.14, page 124 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `VectorDataExtractROIApplication`.

Parameter key	Parameter type	Parameter description
io	Group	Input and output data
io.vd	Input vector data	Input Vector data
io.in	Input image	Support image
io.out	Output vector data	Output Vector data
elev	Group	Elevation management
elev.dem	Directory	DEM directory
elev.geoid	Input File name	Geoid File
elev.default	Float	Default elevation
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.14: Parameters table for VectorData Extract ROI.

Input and output data Group containing input and output parameters

- **Input Vector data:** Input vector data
- **Support image:** Support image that specifies the extracted region
- **Output Vector data:** Output extracted vector data

Elevation management This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).
- **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_VectorDataExtractROIApplication -io.in qb_RoadExtract.tif -io.vd
    qb_RoadExtract_classification.shp -io.out apTvUtVectorDataExtractROIApplicationTest.shp
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the VectorDataExtractROIApplication application
VectorDataExtractROIApplication =
    otbApplication.Registry.CreateApplication("VectorDataExtractROIApplication")

# The following lines set all the application parameters:
VectorDataExtractROIApplication.SetParameterString("io.in", "qb_RoadExtract.tif")

VectorDataExtractROIApplication.SetParameterString("io.vd",
    "qb_RoadExtract_classification.shp")

VectorDataExtractROIApplication.SetParameterString("io.out",
    "apTvUtVectorDataExtractROIApplicationTest.shp")

# The following line execute the application
VectorDataExtractROIApplication.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.2.4 Vector Data reprojection

This application allows to reproject a vector data using support image projection reference, or a user specified map projection

Detailed description

This application allows to reproject a vector data using support image projection reference, or a user given map projection.
If given, image keywordlist can be added to reprojected vectordata.

Parameters

This section describes in details the parameters available for this application. Table 5.15, page 126 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `VectorDataReprojection`.

Parameter key	Parameter type	Parameter description
`in`	Group	Input data
`in.vd`	Input File name	Input vector data
`in.kwl`	Input image	Use image keywords list
`out`	Group	Output data
`out.vd`	Output File name	Output vector data
`out.proj`	Choices	Output Projection choice
`out.proj image`	*Choice*	Use image projection ref
`out.proj user`	*Choice*	User defined projection
`out.proj.image.in`	Input image	Image used to get projection map
`out.proj.user.map`	Choices	Output Cartographic Map Projection
`out.proj.user.map utm`	*Choice*	Universal Trans-Mercator (UTM)
`out.proj.user.map lambert2`	*Choice*	Lambert II Etendu
`out.proj.user.map lambert93`	*Choice*	Lambert93
`out.proj.user.map wgs`	*Choice*	WGS 84
`out.proj.user.map epsg`	*Choice*	EPSG Code
`out.proj.user.map.utm.zone`	Int	Zone number
`out.proj.user.map.utm.northhem`	Boolean	Northern Hemisphere
`out.proj.user.map.epsg.code`	Int	EPSG Code
`elev`	Group	Elevation management
`elev.dem`	Directory	DEM directory
`elev.geoid`	Input File name	Geoid File
`elev.default`	Float	Default elevation
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.15: Parameters table for Vector Data reprojection.

Input data

- **Input vector data:** The input vector data to reproject
- **Use image keywords list:** Optional input image to fill vector data with image kwl.

Output data

- **Output vector data:** The reprojected vector data
- **Output Projection choice:**
  Available choices are:
  - **Use image projection ref**: Vector data will be reprojected in image projection ref.
    - **Image used to get projection map:** Projection map will be found using image metadata
  - **User defined projection**
    - **Output Cartographic Map Projection:** Parameters of the output map projection to be used.
    - **Zone number:** The zone number ranges from 1 to 60 and allows to define the transverse mercator projection (along with the hemisphere)
    - **Northern Hemisphere:** The transverse mercator projections are defined by their zone number as well as the hemisphere. Activate this parameter if your image is in the northern hemisphere.
    - **EPSG Code:** See www.spatialreference.org to find which EPSG code is associated to your projection

Elevation management This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).
- **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_VectorDataReprojection -in.vd VectorData_QB1.shp -out.proj image -out.proj.image.in
     ROI_QB_MUL_1.tif -out.vd reprojected_vd.shp
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the VectorDataReprojection application
VectorDataReprojection = otbApplication.Registry.CreateApplication("VectorDataReprojection")

# The following lines set all the application parameters:
VectorDataReprojection.SetParameterString("in.vd", "VectorData_QB1.shp")

VectorDataReprojection.SetParameterString("out.proj","image")

VectorDataReprojection.SetParameterString("out.proj.image.in", "ROI_QB_MUL_1.tif")

VectorDataReprojection.SetParameterString("out.vd", "reprojected_vd.shp")

# The following line execute the application
VectorDataReprojection.ExecuteAndWriteOutput()
```

Authors

This application has been written by OTB-Team.

5.2.5 Vector data set field

Set a field in vector data.

Detailed description

Set a specified field to a specified value on all features of a vector data.

Parameters

This section describes in details the parameters available for this application. Table 5.16, page 129 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `VectorDataSetField`.

Parameter key	Parameter type	Parameter description
`in`	Input vector data	Input
`out`	Output vector data	Output
`fn`	String	Field
`fv`	String	Value
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.16: Parameters table for Vector data set field.

- **Input:** Input Vector Data
- **Output:** Output Vector Data
- **Field:** Field name
- **Value:** Field value
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_VectorDataSetField -in qb_RoadExtract_classification.shp -out VectorDataSetField.shp
     -fn Info -fv Sample polygon
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the VectorDataSetField application
VectorDataSetField = otbApplication.Registry.CreateApplication("VectorDataSetField")
```

```
# The following lines set all the application parameters:
VectorDataSetField.SetParameterString("in", "qb_RoadExtract_classification.shp")

VectorDataSetField.SetParameterString("out", "VectorDataSetField.shp")

VectorDataSetField.SetParameterString("fn", "Info")

VectorDataSetField.SetParameterString("fv", "Sample polygon")

# The following line execute the application
VectorDataSetField.ExecuteAndWriteOutput()
```

Limitations

Doesn't work with KML files yet

Authors

This application has been written by OTB-Team.

5.2.6 Vector Data Transformation

Apply a transform to each vertex of the input VectorData

Detailed description

This application performs a transformation of an input vector data transforming each vertex in the vector data. The applied transformation manages translation, rotation and scale, and can be centered or not.

Parameters

This section describes in details the parameters available for this application. Table 5.17, page 131 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `VectorDataTransform`.

Parameter key	Parameter type	Parameter description
`vd`	Input vector data	Input Vector data
`out`	Output vector data	Output Vector data
`in`	Input image	Support image
`transform`	Group	Transform parameters
.../...		

Parameter key	Parameter type	Parameter description
.../...		
transform.tx	Float	Translation X
transform.ty	Float	Translation Y
transform.ro	Float	Rotation Angle
transform.centerx	Float	Center X
transform.centery	Float	Center Y
transform.scale	Float	Scale
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.17: Parameters table for Vector Data Transformation.

Input Vector data Input vector data to transform

Output Vector data Output transformed vector data

Support image Image needed as a support to the vector data

Transform parameters Group of parameters to define the transform

- **Translation X:** Translation in the X direction (in pixels)
- **Translation Y:** Translation in the Y direction (in pixels)
- **Rotation Angle:** Angle of the rotation to apply in degrees
- **Center X:** X coordinate of the rotation center (in physical units)
- **Center Y:** Y coordinate of the rotation center (in physical units)
- **Scale:** The scale to apply

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_VectorDataTransform -vd qb_RoadExtract_easyClassification.shp -in qb_RoadExtract.tif
     -out VectorDataTransform.shp -transform.ro 5
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the VectorDataTransform application
VectorDataTransform = otbApplication.Registry.CreateApplication("VectorDataTransform")

# The following lines set all the application parameters:
VectorDataTransform.SetParameterString("vd", "qb_RoadExtract_easyClassification.shp")

VectorDataTransform.SetParameterString("in", "qb_RoadExtract.tif")

VectorDataTransform.SetParameterString("out", "VectorDataTransform.shp")

VectorDataTransform.SetParameterFloat("transform.ro", 5)

# The following line execute the application
VectorDataTransform.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.3 Calibration

5.3.1 Optical calibration

Perform optical calibration TOA/TOC (Top Of Atmosphere/Top Of Canopy). Supported sensors: QuickBird, Ikonos, WorldView2, Formosat, Spot5, Pleiades

Detailed description

The application allows to convert pixel values from DN (for Digital Numbers) to physically interpretable and comparable values. Calibrated values are called surface reflectivity and its values lie in the range [0, 1].
The first level is called Top Of Atmosphere (TOA) reflectivity. It takes into account the sensor gain, sensor spectral response and the solar illumination.
The second level is called Top Of Canopy (TOC) reflectivity. In addition to sensor gain and solar illumination, it takes into account the optical thickness of the atmosphere, the atmospheric pressure, the water vapor amount, the ozone amount, as well as the composition and amount of aerosol gasses. It is also possible to indicate an AERONET file which contains atmospheric parameters (version 1 and version 2 of Aeronet file are supported.

Parameters

This section describes in details the parameters available for this application. Table 5.18, page 134 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `OpticalCalibration`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input
`out`	Output image	Output
`ram`	Int	Available RAM (Mb)
`level`	Choices	Calibration Level
`level toa`	*Choice*	TOA : Top Of Atmosphere
`level toc`	*Choice*	TOC : Top Of Canopy (EXPERIMENTAL)
`milli`	Boolean	Convert to milli reflectance
`clamp`	Boolean	Clamp of reflectivity values between [0, 100]
`rsr`	Input File name	Relative Spectral Response File
`atmo`	Group	Atmospheric parameters
`atmo.aerosol`	Choices	Aerosol Model
`atmo.aerosol noaersol`	*Choice*	No Aerosol Model
`atmo.aerosol continental`	*Choice*	Continental
`atmo.aerosol maritime`	*Choice*	Maritime
`atmo.aerosol urban`	*Choice*	Urban
`atmo.aerosol desertic`	*Choice*	Desertic
`atmo.oz`	Float	Ozone Amount
`atmo.wa`	Float	Water Vapor Amount
`atmo.pressure`	Float	Atmospheric Pressure
`atmo.opt`	Float	Aerosol Optical Thickness
`atmo.aeronet`	Input File name	Aeronet File
`radius`	Int	Window radius
.../...		

Parameter key	Parameter type	Parameter description
.../...		
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.18: Parameters table for Optical calibration.

Input Input image filename (values in DN)

Output Output calibrated image filename

Available RAM (Mb) Available memory for processing (in MB)

Calibration Level Available choices are:

- **TOA : Top Of Atmosphere**
- **TOC : Top Of Canopy (EXPERIMENTAL)**

Convert to milli reflectance Flag to use milli-reflectance instead of reflectance.
This allows to save the image with integer pixel type (in the range [0, 1000] instead of floating point in the range [0, 1]. In order to do that, use this option and set the output pixel type (-out filename uint16 for example)

Clamp of reflectivity values between [0, 100] Clamping in the range [0, 100]. It can be useful to preserve area with specular reflectance.

Relative Spectral Response File Sensor relative spectral response file
By default the application gets these informations in the metadata

Atmospheric parameters This group allows to set the atmospheric parameters.

- **Aerosol Model:**
  Available choices are:

  - **No Aerosol Model**
  - **Continental**
  - **Maritime**
  - **Urban**
  - **Desertic**

- **Ozone Amount:** Ozone Amount
- **Water Vapor Amount:** Water Vapor Amount (in saturation fraction of water)
- **Atmospheric Pressure:** Atmospheric Pressure (in hPa)
- **Aerosol Optical Thickness:** Aerosol Optical Thickness
- **Aeronet File:** Aeronet file containing atmospheric parameters

Window radius Window radius for adjacency effects corrections

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_OpticalCalibration -in QB_1_ortho.tif -level toa -out OpticalCalibration.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the OpticalCalibration application
OpticalCalibration = otbApplication.Registry.CreateApplication("OpticalCalibration")

# The following lines set all the application parameters:
OpticalCalibration.SetParameterString("in", "QB_1_ortho.tif")

OpticalCalibration.SetParameterString("level","toa")

OpticalCalibration.SetParameterString("out", "OpticalCalibration.tif")

# The following line execute the application
OpticalCalibration.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- The OTB CookBook

5.3.2 SAR Radiometric calibration

Perform SAR calibration on input complex images

Detailed description

This application performs SAR calibration on input complex images.

Parameters

This section describes in details the parameters available for this application. Table 5.19, page 137 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `SarRadiometricCalibration`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Complex Image
`out`	Output image	Output Image
`ram`	Int	Available RAM (Mb)
`noise`	Boolean	Disable Noise
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.19: Parameters table for SAR Radiometric calibration.

- **Input Complex Image:** Input complex image
- **Output Image:** Output calibrated complex image
- **Available RAM (Mb):** Available memory for processing (in MB)
- **Disable Noise:** Flag to disable noise
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_SarRadiometricCalibration -in RSAT_imagery_HH.tif -out SarRadiometricCalibration.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the SarRadiometricCalibration application
SarRadiometricCalibration =
     otbApplication.Registry.CreateApplication("SarRadiometricCalibration")

# The following lines set all the application parameters:
SarRadiometricCalibration.SetParameterString("in", "RSAT_imagery_HH.tif")

SarRadiometricCalibration.SetParameterString("out", "SarRadiometricCalibration.tif")

# The following line execute the application
SarRadiometricCalibration.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.4 Geometry

5.4.1 Bundle to perfect sensor

Perform P+XS pansharpening

Detailed description

This application performs P+XS pansharpening.

Parameters

This section describes in details the parameters available for this application. Table 5.20, page 138 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `BundleToPerfectSensor`.

Parameter key	Parameter type	Parameter description
`inp`	Input image	Input PAN Image
`inxs`	Input image	Input XS Image
`elev`	Group	Elevation management
`elev.dem`	Directory	DEM directory
`elev.geoid`	Input File name	Geoid File
`elev.default`	Float	Default elevation
`lms`	Float	Spacing of the deformation field
`out`	Output image	Output image
`ram`	Int	Available RAM (Mb)
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.20: Parameters table for Bundle to perfect sensor.

Input PAN Image Input panchromatic image.

Input XS Image Input XS image.

Elevation management This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).
- **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

Spacing of the deformation field Spacing of the deformation field. Default is 10 times the PAN image spacing.

Output image Output image.

Available RAM (Mb) Available memory for processing (in MB)

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_BundleToPerfectSensor -inp QB_Toulouse_Ortho_PAN.tif -inxs QB_Toulouse_Ortho_XS.tif
     -out BundleToPerfectSensor.png uchar
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the BundleToPerfectSensor application
```

```
BundleToPerfectSensor = otbApplication.Registry.CreateApplication("BundleToPerfectSensor")

# The following lines set all the application parameters:
BundleToPerfectSensor.SetParameterString("inp", "QB_Toulouse_Ortho_PAN.tif")

BundleToPerfectSensor.SetParameterString("inxs", "QB_Toulouse_Ortho_XS.tif")

BundleToPerfectSensor.SetParameterString("out", "BundleToPerfectSensor.png")
BundleToPerfectSensor.SetParameterOutputImagePixelType("out", 1)

# The following line execute the application
BundleToPerfectSensor.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.4.2 Cartographic to geographic coordinates conversion

Convert cartographic coordinates to geographic one.

Detailed description

This application computes the geographic coordinates from a cartographic one. User has to give the X and Y coordinate and the cartographic projection (UTM/LAMBERT/LAMBERT2/LAMBERT93/SINUS/ECKERT4/TRANSMERCATOR/MOLLWEID/SVY21).

Parameters

This section describes in details the parameters available for this application. Table 5.21, page 141 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `ConvertCartoToGeoPoint`.

Parameter key	Parameter type	Parameter description
`carto`	Group	Input cartographic coordinates
`carto.x`	Float	X cartographic coordinates
`carto.y`	Float	Y cartographic coordinates
`mapproj`	Choices	Output Cartographic Map Projection
.../...		

Parameter key	Parameter type	Parameter description
.../...		
mapproj utm	*Choice*	Universal Trans-Mercator (UTM)
mapproj lambert2	*Choice*	Lambert II Etendu
mapproj lambert93	*Choice*	Lambert93
mapproj wgs	*Choice*	WGS 84
mapproj epsg	*Choice*	EPSG Code
mapproj.utm.zone	Int	Zone number
mapproj.utm.northhem	Boolean	Northern Hemisphere
mapproj.epsg.code	Int	EPSG Code
long	Float	Output long
lat	Float	Output lat
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.21: Parameters table for Cartographic to geographic coordinates conversion.

Input cartographic coordinates

- **X cartographic coordinates:** X cartographic coordinates in the specified projection.
- **Y cartographic coordinates:** Y cartographic coordinates in the specified projection.

Output Cartographic Map Projection Parameters of the output map projection to be used. Available choices are:

- **Universal Trans-Mercator (UTM)**: A system of transverse mercator projections dividing the surface of Earth between 80S and 84N latitude.
  - **Zone number:** The zone number ranges from 1 to 60 and allows to define the transverse mercator projection (along with the hemisphere)
  - **Northern Hemisphere:** The transverse mercator projections are defined by their zone number as well as the hemisphere. Activate this parameter if your image is in the northern hemisphere.
- **Lambert II Etendu**: This is a Lambert Conformal Conic projection mainly used in France.
- **Lambert93**: This is a Lambert 93 projection mainly used in France.
- **WGS 84**: This is a Geographical projection

- **EPSG Code**: This code is a generic way of identifying map projections, and allows to specify a large amount of them. See www.spatialreference.org to find which EPSG code is associated to your projection;
  - **EPSG Code:** See www.spatialreference.org to find which EPSG code is associated to your projection

Output long Point longitude coordinates.

Output lat Point latitude coordinates.

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_ConvertCartoToGeoPoint -carto.x 367074.625 -carto.y 4835740 -mapproj utm
     -mapproj.utm.northhem true -mapproj.utm.zone 31
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the ConvertCartoToGeoPoint application
ConvertCartoToGeoPoint = otbApplication.Registry.CreateApplication("ConvertCartoToGeoPoint")

# The following lines set all the application parameters:
ConvertCartoToGeoPoint.SetParameterFloat("carto.x", 367074.625)

ConvertCartoToGeoPoint.SetParameterFloat("carto.y", 4835740)

ConvertCartoToGeoPoint.SetParameterString("mapproj","utm")

ConvertCartoToGeoPoint.SetParameterString("mapproj.utm.northhem","1")

ConvertCartoToGeoPoint.SetParameterInt("mapproj.utm.zone", 31)

# The following line execute the application
ConvertCartoToGeoPoint.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.4.3 Convert Sensor Point To Geographic Point

Sensor to geographic coordinates conversion.

Detailed description

This Application converts a sensor point of an input image to a geographic point using the Forward Sensor Model of the input image.

Parameters

This section describes in details the parameters available for this application. Table 5.22, page 143 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `ConvertSensorToGeoPoint`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Sensor image
`input`	Group	Point Coordinates
`input.idx`	Float	X value of desired point
`input.idy`	Float	Y value of desired point
`output`	Group	Geographic Coordinates
`output.idx`	Float	Output Point Longitude
`output.idy`	Float	Output Point Latitude
`output.town`	String	Main town near the coordinates computed
`output.country`	String	Country of the image
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.22: Parameters table for Convert Sensor Point To Geographic Point.

Sensor image Input sensor image.

Point Coordinates

- **X value of desired point:** X coordinate of the point to transform.
- **Y value of desired point:** Y coordinate of the point to transform.

Geographic Coordinates

- **Output Point Longitude:** Output point longitude coordinate.
- **Output Point Latitude:** Output point latitude coordinate.
- **Main town near the coordinates computed:** Nearest main town of the computed geographic point.
- **Country of the image:** Country of the input image

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_ConvertSensorToGeoPoint -in QB_TOULOUSE_MUL_Extract_500_500.tif -input.idx 200
     -input.idy 200
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the ConvertSensorToGeoPoint application
ConvertSensorToGeoPoint = otbApplication.Registry.CreateApplication("ConvertSensorToGeoPoint")

# The following lines set all the application parameters:
ConvertSensorToGeoPoint.SetParameterString("in", "QB_TOULOUSE_MUL_Extract_500_500.tif")

ConvertSensorToGeoPoint.SetParameterFloat("input.idx", 200)

ConvertSensorToGeoPoint.SetParameterFloat("input.idy", 200)

# The following line execute the application
ConvertSensorToGeoPoint.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- ConvertCartoToGeoPoint application, otbObtainUTMZoneFromGeoPoint

5.4.4 Ply 3D files generation

Generate a 3D Ply file from a DEM and a color image.

Detailed description

Generate a 3D Ply file from a DEM and a color image.

Parameters

This section describes in details the parameters available for this application. Table 5.23, page 146 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `GeneratePlyFile`.

Parameter key	Parameter type	Parameter description
`indem`	Input image	The input DEM
`mode`	Choices	Conversion Mode
`mode dem`	*Choice*	DEM
`mode 3dgrid`	*Choice*	3D grid
`map`	Choices	Output Cartographic Map Projection
`map utm`	*Choice*	Universal Trans-Mercator (UTM)
`map lambert2`	*Choice*	Lambert II Etendu
`map lambert93`	*Choice*	Lambert93
`map wgs`	*Choice*	WGS 84
`map epsg`	*Choice*	EPSG Code
.../...		

Parameter key	Parameter type	Parameter description
.../...		
map.utm.zone	Int	Zone number
map.utm.northhem	Boolean	Northern Hemisphere
map.epsg.code	Int	EPSG Code
incolor	Input image	The input color image
out	Output File name	The output Ply file
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.23: Parameters table for Ply 3D files generation.

The input DEM The input DEM

Conversion Mode Available choices are:

- **DEM**: DEM conversion mode
- **3D grid**: 3D grid conversion mode

Output Cartographic Map Projection Parameters of the output map projection to be used. Available choices are:

- **Universal Trans-Mercator (UTM)**: A system of transverse mercator projections dividing the surface of Earth between 80S and 84N latitude.
  - **Zone number:** The zone number ranges from 1 to 60 and allows to define the transverse mercator projection (along with the hemisphere)
  - **Northern Hemisphere:** The transverse mercator projections are defined by their zone number as well as the hemisphere. Activate this parameter if your image is in the northern hemisphere.
- **Lambert II Etendu**: This is a Lambert Conformal Conic projection mainly used in France.
- **Lambert93**: This is a Lambert 93 projection mainly used in France.
- **WGS 84**: This is a Geographical projection
- **EPSG Code**: This code is a generic way of identifying map projections, and allows to specify a large amount of them. See www.spatialreference.org to find which EPSG code is associated to your projection;

– **EPSG Code:** See www.spatialreference.org to find which EPSG code is associated to your projection

The input color image The input color image

The output Ply file The output Ply file

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_GeneratePlyFile -indem image_dem.tif -out out.ply -incolor image_color.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the GeneratePlyFile application
GeneratePlyFile = otbApplication.Registry.CreateApplication("GeneratePlyFile")

# The following lines set all the application parameters:
GeneratePlyFile.SetParameterString("indem", "image_dem.tif")

GeneratePlyFile.SetParameterString("out", "out.ply")

GeneratePlyFile.SetParameterString("incolor", "image_color.tif")

# The following line execute the application
GeneratePlyFile.ExecuteAndWriteOutput()
```

Authors

This application has been written by OTB-Team.

5.4.5 Generate a RPC sensor model

Generate a RPC sensor model from a list of Ground Control Points.

Detailed description

This application generates a RPC sensor model from a list of Ground Control Points. At least 20 points are required for estimation wihtout elevation support, and 40 points for estimation with elevation support. Elevation support will be automatically deactivated if an insufficient amount of points is provided. The application can optionnaly output a file containing accuracy statistics for each point, and a vector file containing segments represening points residues. The map projection parameter allows to define a map projection in which the accuracy is evaluated.

Parameters

This section describes in details the parameters available for this application. Table 5.24, page 148 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `GenerateRPCSensorModel`.

Parameter key	Parameter type	Parameter description
`outgeom`	Output File name	Output geom file
`inpoints`	Input File name	Input file containing tie points
`outstat`	Output File name	Output file containing output precision statistics
`outvector`	Output File name	Output vector file with residues
`map`	Choices	Output Cartographic Map Projection
`map utm`	*Choice*	Universal Trans-Mercator (UTM)
`map lambert2`	*Choice*	Lambert II Etendu
`map lambert93`	*Choice*	Lambert93
`map wgs`	*Choice*	WGS 84
`map epsg`	*Choice*	EPSG Code
`map.utm.zone`	Int	Zone number
`map.utm.northhem`	Boolean	Northern Hemisphere
`map.epsg.code`	Int	EPSG Code
`elev`	Group	Elevation management
`elev.dem`	Directory	DEM directory
`elev.geoid`	Input File name	Geoid File
`elev.default`	Float	Default elevation
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.24: Parameters table for Generate a RPC sensor model.

Output geom file Geom file containing the generated RPC sensor model

Input file containing tie points Input file containing tie points. Points are stored in following format: row col lon lat. Line beginning with # are ignored.

Output file containing output precision statistics Output file containing the following info: ref_lon ref_lat elevation predicted_lon predicted_lat x_error_ref(meters) y_error_ref(meters) global_error_ref(meters) x_error(meters) y_error(meters) overall_error(meters)

Output vector file with residues File containing segments representing residues

Output Cartographic Map Projection Parameters of the output map projection to be used. Available choices are:

- **Universal Trans-Mercator (UTM)**: A system of transverse mercator projections dividing the surface of Earth between 80S and 84N latitude.
  - **Zone number:** The zone number ranges from 1 to 60 and allows to define the transverse mercator projection (along with the hemisphere)
  - **Northern Hemisphere:** The transverse mercator projections are defined by their zone number as well as the hemisphere. Activate this parameter if your image is in the northern hemisphere.
- **Lambert II Etendu**: This is a Lambert Conformal Conic projection mainly used in France.
- **Lambert93**: This is a Lambert 93 projection mainly used in France.
- **WGS 84**: This is a Geographical projection
- **EPSG Code**: This code is a generic way of identifying map projections, and allows to specify a large amount of them. See www.spatialreference.org to find which EPSG code is associated to your projection;
  - **EPSG Code:** See www.spatialreference.org to find which EPSG code is associated to your projection

Elevation management This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles

- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).
- **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_GenerateRPCSensorModel -outgeom output.geom -inpoints points.txt -map epsg
     -map.epsg.code 32631
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the GenerateRPCSensorModel application
GenerateRPCSensorModel = otbApplication.Registry.CreateApplication("GenerateRPCSensorModel")

# The following lines set all the application parameters:
GenerateRPCSensorModel.SetParameterString("outgeom", "output.geom")

GenerateRPCSensorModel.SetParameterString("inpoints", "points.txt")

GenerateRPCSensorModel.SetParameterString("map","epsg")

GenerateRPCSensorModel.SetParameterInt("map.epsg.code", 32631)

# The following line execute the application
GenerateRPCSensorModel.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- OrthoRectication,HomologousPointsExtraction,RefineSensorModel

5.4.6 Grid Based Image Resampling

Resamples an image according to a resampling grid

Detailed description

This application allows to perform image resampling from an input resampling grid.

Parameters

This section describes in details the parameters available for this application. Table 5.25, page 152 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `GridBasedImageResampling`.

Parameter key	Parameter type	Parameter description
`io`	Group	Input and output data
`io.in`	Input image	Input image
`io.out`	Output image	Output Image
`grid`	Group	Resampling grid parameters
`grid.in`	Input image	Input resampling grid
`grid.type`	Choices	Grid Type
`grid.type def`	*Choice*	Displacement grid: $G(x_out, y_out) = (x_in - x_out, y_in - y_out)$
`grid.type loc`	*Choice*	Localisation grid: $G(x_out, y_out) = (x_in, y_in)$
`out`	Group	Output Image parameters
`out.ulx`	Float	Upper Left X
`out.uly`	Float	Upper Left Y
`out.sizex`	Int	Size X
`out.sizey`	Int	Size Y
`out.spacingx`	Float	Pixel Size X
.../...		

Parameter key	Parameter type	Parameter description
.../...		
out.spacingy	Float	Pixel Size Y
out.default	Float	Default value
interpolator	Choices	Interpolation
interpolator nn	*Choice*	Nearest Neighbor interpolation
interpolator linear	*Choice*	Linear interpolation
interpolator bco	*Choice*	Bicubic interpolation
interpolator.bco.radius	Int	Radius for bicubic interpolation
ram	Int	Available RAM (Mb)
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.25: Parameters table for Grid Based Image Resampling.

Input and output data This group of parameters allows to set the input and output images.

- **Input image:** The input image to resample
- **Output Image:** The resampled output image

Resampling grid parameters

- **Input resampling grid:** The resampling grid
- **Grid Type:** Allows to choose between two grid types

  Available choices are:

  - **Displacement grid:** $G(x\_out, y\_out) = (x\_in - x\_out, y\_in - y\_out)$: A deformation grid contains at each grid position the offset to apply to this position in order to get to the corresponding point in the input image to resample
  - **Localisation grid:** $G(x\_out, y\_out) = (x\_in, y\_in)$: A localisation grid contains at each grid position the corresponding position in the input image to resample

Output Image parameters Parameters of the output image

- **Upper Left X:** X Coordinate of the upper-left pixel of the output resampled image
- **Upper Left Y:** Y Coordinate of the upper-left pixel of the output resampled image

- **Size X:** Size of the output resampled image along X (in pixels)
- **Size Y:** Size of the output resampled image along Y (in pixels)
- **Pixel Size X:** Size of each pixel along X axis
- **Pixel Size Y:** Size of each pixel along Y axis
- **Default value:** The default value to give to pixel that falls outside of the input image.

Interpolation This group of parameters allows to define how the input image will be interpolated during resampling. Available choices are:

- **Nearest Neighbor interpolation**: Nearest neighbor interpolation leads to poor image quality, but it is very fast.
- **Linear interpolation**: Linear interpolation leads to average image quality but is quite fast
- **Bicubic interpolation**
  - **Radius for bicubic interpolation:** This parameter allows to control the size of the bicubic interpolation filter. If the target pixel size is higher than the input pixel size, increasing this parameter will reduce aliasing artefacts.

Available RAM (Mb) Available memory for processing (in MB)

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_GridBasedImageResampling -io.in ROI_IKO_PAN_LesHalles_sub.tif -io.out
     ROI_IKO_PAN_LesHalles_sub_resampled.tif uint8 -grid.in
     ROI_IKO_PAN_LesHalles_sub_deformation_field.tif -out.sizex 256 -out.sizey 256 -grid.type
     def
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication
```

```
# The following line creates an instance of the GridBasedImageResampling application
GridBasedImageResampling =
     otbApplication.Registry.CreateApplication("GridBasedImageResampling")

# The following lines set all the application parameters:
GridBasedImageResampling.SetParameterString("io.in", "ROI_IKO_PAN_LesHalles_sub.tif")

GridBasedImageResampling.SetParameterString("io.out",
     "ROI_IKO_PAN_LesHalles_sub_resampled.tif")
GridBasedImageResampling.SetParameterOutputImagePixelType("io.out", 1)

GridBasedImageResampling.SetParameterString("grid.in",
     "ROI_IKO_PAN_LesHalles_sub_deformation_field.tif")

GridBasedImageResampling.SetParameterInt("out.sizex", 256)

GridBasedImageResampling.SetParameterInt("out.sizey", 256)

GridBasedImageResampling.SetParameterString("grid.type","def")

# The following line execute the application
GridBasedImageResampling.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- otbStereorecificationGridGeneration

5.4.7 Image Envelope

Extracts an image envelope.

Detailed description

Build a vector data containing the polygon of the image envelope.

Parameters

This section describes in details the parameters available for this application. Table 5.26, page 155 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `ImageEnvelope`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`out`	Output vector data	Output Vector Data
`sr`	Int	Sampling Rate
`elev`	Group	Elevation management
`elev.dem`	Directory	DEM directory
`elev.geoid`	Input File name	Geoid File
`elev.default`	Float	Default elevation
`proj`	String	Projection
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.26: Parameters table for Image Envelope.

Input Image Input image.

Output Vector Data Vector data file containing the envelope

Sampling Rate Sampling rate for image edges (in pixel)

Elevation management This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).

- **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

Projection Projection to be used to compute the envelope (default is WGS84)

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_ImageEnvelope -in QB_TOULOUSE_MUL_Extract_500_500.tif -out ImageEnvelope.shp
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the ImageEnvelope application
ImageEnvelope = otbApplication.Registry.CreateApplication("ImageEnvelope")

# The following lines set all the application parameters:
ImageEnvelope.SetParameterString("in", "QB_TOULOUSE_MUL_Extract_500_500.tif")

ImageEnvelope.SetParameterString("out", "ImageEnvelope.shp")

# The following line execute the application
ImageEnvelope.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.4.8 Ortho-rectification

This application allows to ortho-rectify optical images from supported sensors.

Detailed description

An inverse sensor model is built from the input image metadata to convert geographical to raw geometry coordinates. This inverse sensor model is then combined with the chosen map projection to build a global coordinate mapping grid. Last, this grid is used to resample using the chosen interpolation algorithm. A Digital Elevation Model can be specified to account for terrain deformations.
In case of SPOT5 images, the sensor model can be approximated by an RPC model in order to speed-up computation.

Parameters

This section describes in details the parameters available for this application. Table 5.27, page 158 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `OrthoRectification`.

Parameter key	Parameter type	Parameter description
`io`	Group	Input and output data
`io.in`	Input image	Input Image
`io.out`	Output image	Output Image
`map`	Choices	Output Cartographic Map Projection
`map utm`	*Choice*	Universal Trans-Mercator (UTM)
`map lambert2`	*Choice*	Lambert II Etendu
`map lambert93`	*Choice*	Lambert93
`map wgs`	*Choice*	WGS 84
`map epsg`	*Choice*	EPSG Code
`map.utm.zone`	Int	Zone number
`map.utm.northhem`	Boolean	Northern Hemisphere
`map.epsg.code`	Int	EPSG Code
`outputs`	Group	Output Image Grid
`outputs.mode`	Choices	Parameters estimation modes
`outputs.mode auto`	*Choice*	User Defined
`outputs.mode autosize`	*Choice*	Automatic Size from Spacing
`outputs.mode autospacing`	*Choice*	Automatic Spacing from Size
`outputs.mode outputroi`	*Choice*	Automatic Size from Spacing and output corners
`outputs.mode orthofit`	*Choice*	Fit to ortho
`outputs.ulx`	Float	Upper Left X
.../...		

Parameter key	Parameter type	Parameter description
.../...		
outputs.uly	Float	Upper Left Y
outputs.sizex	Int	Size X
outputs.sizey	Int	Size Y
outputs.spacingx	Float	Pixel Size X
outputs.spacingy	Float	Pixel Size Y
outputs.lrx	Float	Lower right X
outputs.lry	Float	Lower right Y
outputs.ortho	Input image	Model ortho-image
outputs.isotropic	Boolean	Force isotropic spacing by default
outputs.default	Float	Default pixel value
elev	Group	Elevation management
elev.dem	Directory	DEM directory
elev.geoid	Input File name	Geoid File
elev.default	Float	Default elevation
interpolator	Choices	Interpolation
interpolator bco	*Choice*	Bicubic interpolation
interpolator nn	*Choice*	Nearest Neighbor interpolation
interpolator linear	*Choice*	Linear interpolation
interpolator.bco.radius	Int	Radius for bicubic interpolation
opt	Group	Speed optimization parameters
opt.rpc	Int	RPC modeling (points per axis)
opt.ram	Int	Available RAM (Mb)
opt.gridspacing	Float	Resampling grid spacing
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.27: Parameters table for Ortho-rectification.

Input and output data This group of parameters allows to set the input and output images.

- **Input Image:** The input image to ortho-rectify
- **Output Image:** The ortho-rectified output image

Output Cartographic Map Projection Parameters of the output map projection to be used. Available choices are:

- **Universal Trans-Mercator (UTM)**: A system of transverse mercator projections dividing the surface of Earth between 80S and 84N latitude.

  - **Zone number:** The zone number ranges from 1 to 60 and allows to define the transverse mercator projection (along with the hemisphere)
  - **Northern Hemisphere:** The transverse mercator projections are defined by their zone number as well as the hemisphere. Activate this parameter if your image is in the northern hemisphere.

- **Lambert II Etendu**: This is a Lambert Conformal Conic projection mainly used in France.
- **Lambert93**: This is a Lambert 93 projection mainly used in France.
- **WGS 84**: This is a Geographical projection
- **EPSG Code**: This code is a generic way of identifying map projections, and allows to specify a large amount of them. See www.spatialreference.org to find which EPSG code is associated to your projection;
  - **EPSG Code:** See www.spatialreference.org to find which EPSG code is associated to your projection

Output Image Grid This group of parameters allows to define the grid on which the input image will be resampled.

- **Parameters estimation modes:**

  Available choices are:
  - **User Defined**: This mode allows you to fully modify default values.
  - **Automatic Size from Spacing**: This mode allows you to automatically compute the optimal image size from given spacing (pixel size) values
  - **Automatic Spacing from Size**: This mode allows you to automatically compute the optimal image spacing (pixel size) from the given size
  - **Automatic Size from Spacing and output corners**: This mode allows you to automatically compute the optimal image size from spacing (pixel size) and output corners
  - **Fit to ortho**: Fit the size, origin and spacing to an existing ortho image (uses the value of outputs.ortho)
- **Upper Left X:** Cartographic X coordinate of upper-left corner (meters for cartographic projections, degrees for geographic ones)
- **Upper Left Y:** Cartographic Y coordinate of the upper-left corner (meters for cartographic projections, degrees for geographic ones)
- **Size X:** Size of projected image along X (in pixels)
- **Size Y:** Size of projected image along Y (in pixels)

- **Pixel Size X:** Size of each pixel along X axis (meters for cartographic projections, degrees for geographic ones)
- **Pixel Size Y:** Size of each pixel along Y axis (meters for cartographic projections, degrees for geographic ones)
- **Lower right X:** Cartographic X coordinate of the lower-right corner (meters for cartographic projections, degrees for geographic ones)
- **Lower right Y:** Cartographic Y coordinate of the lower-right corner (meters for cartographic projections, degrees for geographic ones)
- **Model ortho-image:** A model ortho-image that can be used to compute size, origin and spacing of the output
- **Force isotropic spacing by default:** Default spacing (pixel size) values are estimated from the sensor modeling of the image. It can therefore result in a non-isotropic spacing. This option allows you to force default values to be isotropic (in this case, the minimum of spacing in both direction is applied. Values overriden by user are not affected by this option.
- **Default pixel value:** Default value to write when outside of input image.

Elevation management This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).
- **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

Interpolation This group of parameters allows to define how the input image will be interpolated during resampling. Available choices are:

- **Bicubic interpolation**

  - **Radius for bicubic interpolation:** This parameter allows to control the size of the bicubic interpolation filter. If the target pixel size is higher than the input pixel size, increasing this parameter will reduce aliasing artefacts.

- **Nearest Neighbor interpolation**: Nearest neighbor interpolation leads to poor image quality, but it is very fast.

- **Linear interpolation**: Linear interpolation leads to average image quality but is quite fast

Speed optimization parameters This group of parameters allows to optimize processing time.

- **RPC modeling (points per axis):** Enabling RPC modeling allows to speed-up SPOT5 ortho-rectification. Value is the number of control points per axis for RPC estimation

- **Available RAM (Mb):** This allows to set the maximum amount of RAM available for processing. As the writing task is time consuming, it is better to write large pieces of data, which can be achieved by increasing this parameter (pay attention to your system capabilities)

- **Resampling grid spacing:** Resampling is done according to a coordinate mapping deformation grid, whose pixel size is set by this parameter, and expressed in the coordinate system of the output image The closer to the output spacing this parameter is, the more precise will be the ortho-rectified image,but increasing this parameter will reduce processing time.

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_OrthoRectification -io.in QB_TOULOUSE_MUL_Extract_500_500.tif -io.out
    QB_Toulouse_ortho.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the OrthoRectification application
OrthoRectification = otbApplication.Registry.CreateApplication("OrthoRectification")

# The following lines set all the application parameters:
OrthoRectification.SetParameterString("io.in", "QB_TOULOUSE_MUL_Extract_500_500.tif")
```

```
OrthoRectification.SetParameterString("io.out", "QB_Toulouse_ortho.tif")

# The following line execute the application
OrthoRectification.ExecuteAndWriteOutput()
```

Limitations

Supported sensors are Pleiades, SPOT5 (TIF format), Ikonos, Quickbird, Worldview2, GeoEye.

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- Ortho-rectification chapter from the OTB Software Guide

5.4.9 Pansharpening

Perform P+XS pansharpening

Detailed description

This application performs P+XS pansharpening. Pansharpening is a process of merging high-resolution panchromatic and lower resolution multispectral imagery to create a single high-resolution color image. Algorithms available in the applications are: RCS, bayesian fusion and Local Mean and Variance Matching(LMVM).

Parameters

This section describes in details the parameters available for this application. Table 5.28, page 163 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `Pansharpening`.

Parameter key	Parameter type	Parameter description
.../...		

Parameter key	Parameter type	Parameter description
.../...		
inp	Input image	Input PAN Image
inxs	Input image	Input XS Image
out	Output image	Output image
method	Choices	Algorithm
method rcs	*Choice*	RCS
method lmvm	*Choice*	LMVM
method bayes	*Choice*	Bayesian
method.lmvm.radiusx	Int	X radius
method.lmvm.radiusy	Int	Y radius
method.bayes.lambda	Float	Weight
method.bayes.s	Float	S coefficient
ram	Int	Available RAM (Mb)
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.28: Parameters table for Pansharpening.

Input PAN Image Input panchromatic image.

Input XS Image Input XS image.

Output image Output image.

Algorithm Selection of the pan-sharpening method. Available choices are:

- **RCS**: Simple RCS Pan sharpening operation.
- **LMVM**: Local Mean and Variance Matching (LMVM) Pan sharpening.
  - **X radius:** Set the x radius of the sliding window.
  - **Y radius:** Set the y radius of the sliding window.
- **Bayesian**: Bayesian fusion.
  - **Weight:** Set the weighting value.
  - **S coefficient:** Set the S coefficient.

Available RAM (Mb) Available memory for processing (in MB)

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_Pansharpening -inp QB_Toulouse_Ortho_PAN.tif -inxs QB_Toulouse_Ortho_XS.tif -out
     Pansharpening.tif uint16
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the Pansharpening application
Pansharpening = otbApplication.Registry.CreateApplication("Pansharpening")

# The following lines set all the application parameters:
Pansharpening.SetParameterString("inp", "QB_Toulouse_Ortho_PAN.tif")

Pansharpening.SetParameterString("inxs", "QB_Toulouse_Ortho_XS.tif")

Pansharpening.SetParameterString("out", "Pansharpening.tif")
Pansharpening.SetParameterOutputImagePixelType("out", 3)

# The following line execute the application
Pansharpening.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.4.10 Refine Sensor Model

Perform least-square fit of a sensor model to a set of tie points

Detailed description

This application reads a geom file containing a sensor model and a text file containing a list of ground control point, and performs a least-square fit of the sensor model adjustable parameters to these tie points. It produces an updated geom file as output, as well as an optional ground control points based statistics file and a vector file containing residues. The output geom file can then be used to ortho-rectify the data more accurately. Plaease note that for a proper use of the application, elevation must be correctly set (including DEM and geoid file). The map parameters allows to choose a map projection in which the accuracy will be estimated in meters.

Parameters

This section describes in details the parameters available for this application. Table 5.29, page 165 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `RefineSensorModel`.

Parameter key	Parameter type	Parameter description
`ingeom`	Input File name	Input geom file
`outgeom`	Output File name	Output geom file
`inpoints`	Input File name	Input file containing tie points
`outstat`	Output File name	Output file containing output precision statistics
`outvector`	Output File name	Output vector file with residues
`map`	Choices	Output Cartographic Map Projection
`map utm`	*Choice*	Universal Trans-Mercator (UTM)
`map lambert2`	*Choice*	Lambert II Etendu
`map lambert93`	*Choice*	Lambert93
`map wgs`	*Choice*	WGS 84
`map epsg`	*Choice*	EPSG Code
`map.utm.zone`	Int	Zone number
`map.utm.northhem`	Boolean	Northern Hemisphere
`map.epsg.code`	Int	EPSG Code
`elev`	Group	Elevation management
`elev.dem`	Directory	DEM directory
`elev.geoid`	Input File name	Geoid File
`elev.default`	Float	Default elevation
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.29: Parameters table for Refine Sensor Model.

Input geom file Geom file containing the sensor model to refine

Output geom file Geom file containing the refined sensor model

Input file containing tie points Input file containing tie points. Points are stored in following format: row col lon lat. Line beginning with # are ignored.

Output file containing output precision statistics Output file containing the following info: ref_lon ref_lat elevation predicted_lon predicted_lat x_error_ref(meters) y_error_ref(meters) global_error_ref(meters) x_error(meters) y_error(meters) overall_error(meters)

Output vector file with residues File containing segments representing residues

Output Cartographic Map Projection Parameters of the output map projection to be used. Available choices are:

- **Universal Trans-Mercator (UTM)**: A system of transverse mercator projections dividing the surface of Earth between 80S and 84N latitude.
  - **Zone number:** The zone number ranges from 1 to 60 and allows to define the transverse mercator projection (along with the hemisphere)
  - **Northern Hemisphere:** The transverse mercator projections are defined by their zone number as well as the hemisphere. Activate this parameter if your image is in the northern hemisphere.
- **Lambert II Etendu**: This is a Lambert Conformal Conic projection mainly used in France.
- **Lambert93**: This is a Lambert 93 projection mainly used in France.
- **WGS 84**: This is a Geographical projection
- **EPSG Code**: This code is a generic way of identifying map projections, and allows to specify a large amount of them. See www.spatialreference.org to find which EPSG code is associated to your projection;
  - **EPSG Code:** See www.spatialreference.org to find which EPSG code is associated to your projection

Elevation management This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).
- **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_RefineSensorModel -ingeom input.geom -outgeom output.geom -inpoints points.txt -map
    epsg -map.epsg.code 32631
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the RefineSensorModel application
RefineSensorModel = otbApplication.Registry.CreateApplication("RefineSensorModel")

# The following lines set all the application parameters:
RefineSensorModel.SetParameterString("ingeom", "input.geom")

RefineSensorModel.SetParameterString("outgeom", "output.geom")

RefineSensorModel.SetParameterString("inpoints", "points.txt")

RefineSensorModel.SetParameterString("map","epsg")

RefineSensorModel.SetParameterInt("map.epsg.code", 32631)

# The following line execute the application
RefineSensorModel.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- OrthoRectification,HomologousPointsExtraction

5.4.11 Image resampling with a rigid transform

Resample an image with a rigid transform

Detailed description

This application performs a parametric transform on the input image. Scaling, translation and rotation with scaling factor are handled. Parameters of the transform is expressed in physical units, thus particular attention must be paid on pixel size (value, and sign). Moreover transform is expressed from input space to output space (on the contrary ITK Transforms are expressed form output space to input space).

Parameters

This section describes in details the parameters available for this application. Table 5.30, page 169 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `RigidTransformResample`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input image
`out`	Output image	Output image
`transform`	Group	Transform parameters
`transform.type`	Choices	Type of transformation
`transform.type id`	*Choice*	id
`transform.type translation`	*Choice*	translation
.../...		

Parameter key	Parameter type	Parameter description
.../...		
transform.type rotation	*Choice*	rotation
transform.type.id.scalex	Float	X scaling
transform.type.id.scaley	Float	Y scaling
transform.type.translation.tx	Float	The X translation (in physical units)
transform.type.translation.ty	Float	The Y translation (in physical units)
transform.type.translation.scalex	Float	X scaling
transform.type.translation.scaley	Float	Y scaling
transform.type.rotation.angle	Float	Rotation angle
transform.type.rotation.scalex	Float	X scaling
transform.type.rotation.scaley	Float	Y scaling
interpolator	Choices	Interpolation
interpolator nn	*Choice*	Nearest Neighbor interpolation
interpolator linear	*Choice*	Linear interpolation
interpolator bco	*Choice*	Bicubic interpolation
interpolator.bco.radius	Int	Radius for bicubic interpolation
ram	Int	Available RAM (Mb)
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.30: Parameters table for Image resampling with a rigid transform.

Input image The input image to translate.

Output image The transformed output image.

Transform parameters This group of parameters allows to set the transformation to apply.

- **Type of transformation:** Type of transformation. Available transformations are spatial scaling, translation and rotation with scaling factor

  Available choices are:

  - **id**: Spatial scaling
    - **X scaling:** Scaling factor between the output X spacing and the input X spacing
    - **Y scaling:** Scaling factor between the output Y spacing and the input Y spacing
  - **translation**: translation
    - **The X translation (in physical units):** The translation value along X axis (in physical units).

    - **The Y translation (in physical units):** The translation value along Y axis (in physical units)
    - **X scaling:** Scaling factor between the output X spacing and the input X spacing
    - **Y scaling:** Scaling factor between the output Y spacing and the input Y spacing
  - **rotation**: rotation
    - **Rotation angle:** The rotation angle in degree (values between -180 and 180)
    - **X scaling:** Scale factor between the X spacing of the rotated output image and the X spacing of the unrotated image
    - **Y scaling:** Scale factor between the Y spacing of the rotated output image and the Y spacing of the unrotated image

Interpolation This group of parameters allows to define how the input image will be interpolated during resampling. Available choices are:

- **Nearest Neighbor interpolation**: Nearest neighbor interpolation leads to poor image quality, but it is very fast.
- **Linear interpolation**: Linear interpolation leads to average image quality but is quite fast
- **Bicubic interpolation**
  - **Radius for bicubic interpolation:** This parameter allows to control the size of the bicubic interpolation filter. If the target pixel size is higher than the input pixel size, increasing this parameter will reduce aliasing artefacts.

Available RAM (Mb) This allows to set the maximum amount of RAM available for processing. As the writing task is time consuming, it is better to write large pieces of data, which can be achieved by increasing this parameter (pay attention to your system capabilities)

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_RigidTransformResample -in qb_toulouse_sub.tif -out rigitTransformImage.tif
     -transform.type rotation -transform.type.rotation.angle 20
     -transform.type.rotation.scalex 2. -transform.type.rotation.scaley 2.
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the RigidTransformResample application
RigidTransformResample = otbApplication.Registry.CreateApplication("RigidTransformResample")

# The following lines set all the application parameters:
RigidTransformResample.SetParameterString("in", "qb_toulouse_sub.tif")

RigidTransformResample.SetParameterString("out", "rigitTransformImage.tif")

RigidTransformResample.SetParameterString("transform.type","rotation")

RigidTransformResample.SetParameterFloat("transform.type.rotation.angle", 20)

RigidTransformResample.SetParameterFloat("transform.type.rotation.scalex", 2.)

RigidTransformResample.SetParameterFloat("transform.type.rotation.scaley", 2.)

# The following line execute the application
RigidTransformResample.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- Translation

5.4.12 Superimpose sensor

Using available image metadata, project one image onto another one

Detailed description

This application performs the projection of an image into the geometry of another one.

Parameters

This section describes in details the parameters available for this application. Table 5.31, page 172 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `Superimpose`.

Parameter key	Parameter type	Parameter description
`inr`	Input image	Reference input
`inm`	Input image	The image to reproject
`elev`	Group	Elevation management
`elev.dem`	Directory	DEM directory
`elev.geoid`	Input File name	Geoid File
`elev.default`	Float	Default elevation
`lms`	Float	Spacing of the deformation field
`out`	Output image	Output image
`interpolator`	Choices	Interpolation
`interpolator bco`	*Choice*	Bicubic interpolation
`interpolator nn`	*Choice*	Nearest Neighbor interpolation
`interpolator linear`	*Choice*	Linear interpolation
`interpolator.bco.radius`	Int	Radius for bicubic interpolation
`ram`	Int	Available RAM (Mb)
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.31: Parameters table for Superimpose sensor.

Reference input The input reference image.

The image to reproject The image to reproject into the geometry of the reference input.

Elevation management This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version

of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).

- **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

Spacing of the deformation field Generate a coarser deformation field with the given spacing

Output image Output reprojected image.

Interpolation This group of parameters allows to define how the input image will be interpolated during resampling. Available choices are:

- **Bicubic interpolation**: Bicubic interpolation leads to very good image quality but is slow.
  - **Radius for bicubic interpolation:** This parameter allows to control the size of the bicubic interpolation filter. If the target pixel size is higher than the input pixel size, increasing this parameter will reduce aliasing artefacts.
- **Nearest Neighbor interpolation**: Nearest neighbor interpolation leads to poor image quality, but it is very fast.
- **Linear interpolation**: Linear interpolation leads to average image quality but is quite fast

Available RAM (Mb) Available memory for processing (in MB)

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_Superimpose -inr QB_Toulouse_Ortho_PAN.tif -inm QB_Toulouse_Ortho_XS.tif -out
     SuperimposedXS_to_PAN.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the Superimpose application
Superimpose = otbApplication.Registry.CreateApplication("Superimpose")

# The following lines set all the application parameters:
Superimpose.SetParameterString("inr", "QB_Toulouse_Ortho_PAN.tif")

Superimpose.SetParameterString("inm", "QB_Toulouse_Ortho_XS.tif")

Superimpose.SetParameterString("out", "SuperimposedXS_to_PAN.tif")

# The following line execute the application
Superimpose.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.5 Image Filtering

5.5.1 Dimensionality reduction

Perform Dimension reduction of the input image.

Detailed description

Performs dimensionality reduction on input image. PCA,NA-PCA,MAF,ICA methods are available. It is also possible to compute the inverse transform to reconstruct the image. It is also possible to optionnaly export the transformation matrix to a text file.

Parameters

This section describes in details the parameters available for this application. Table 5.32, page 175 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `DimensionalityReduction`.

Parameter key	Parameter type	Parameter description
in	Input image	Input Image
out	Output image	Output Image
rescale	Group	Rescale Output.
rescale.outmin	Float	Output min value
rescale.outmax	Float	Output max value
outinv	Output image	Inverse Output Image
method	Choices	Algorithm
method pca	*Choice*	PCA
method napca	*Choice*	NA-PCA
method maf	*Choice*	MAF
method ica	*Choice*	ICA
method.napca.radiusx	Int	Set the x radius of the sliding window.
method.napca.radiusy	Int	Set the y radius of the sliding window.
method.ica.iter	Int	number of iterations
method.ica.mu	Float	Give the increment weight of W in [0, 1]
nbcomp	Int	Number of Components.
normalize	Boolean	Normalize.
outmatrix	Output File name	Transformation matrix output (text format)
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.32: Parameters table for Dimensionality reduction.

Input Image The input image to apply dimensionality reduction.

Output Image output image. Components are ordered by decreasing eigenvalues.

Rescale Output.

- **Output min value:** Minimum value of the output image.
- **Output max value:** Maximum value of the output image.

Inverse Output Image reconstruct output image.

Algorithm Selection of the reduction dimension method. Available choices are:

- **PCA**: Principal Component Analysis.
- **NA-PCA**: Noise Adjusted Principal Component Analysis.
  - **Set the x radius of the sliding window.:**
  - **Set the y radius of the sliding window.:**
- **MAF**: Maximum Autocorrelation Factor.
- **ICA**: Independant Component Analysis.
  - **number of iterations :**
  - **Give the increment weight of W in [0, 1]:**

Number of Components. Number of relevant components kept. By default all components are kept.

Normalize. center AND reduce data before Dimensionality reduction.

Transformation matrix output (text format) Filename to store the transformation matrix (csv format)

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_DimensionalityReduction -in cupriteSubHsi.tif -out FilterOutput.tif -method pca
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the DimensionalityReduction application
DimensionalityReduction = otbApplication.Registry.CreateApplication("DimensionalityReduction")

# The following lines set all the application parameters:
DimensionalityReduction.SetParameterString("in", "cupriteSubHsi.tif")

DimensionalityReduction.SetParameterString("out", "FilterOutput.tif")
```

```
DimensionalityReduction.SetParameterString("method","pca")

# The following line execute the application
DimensionalityReduction.ExecuteAndWriteOutput()
```

Limitations

This application does not provide the inverse transform and the transformation matrix export for the MAF.

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- "Kernel maximum autocorrelation factor and minimum noise fraction transformations," IEEE Transactions on Image Processing, vol. 20, no. 3, pp. 612-624, (2011)

5.5.2 Mean Shift filtering (can be used as Exact Large-Scale Mean-Shift segmentation, step 1)

Perform mean shift filtering

Detailed description

This application performs mean shift fitlering (multi-threaded).

Parameters

This section describes in details the parameters available for this application. Table 5.33, page 178 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `MeanShiftSmoothing`.

Parameter key	Parameter type	Parameter description
.../...		

Parameter key	Parameter type	Parameter description
.../...		
in	Input image	Input Image
fout	Output image	Filtered output
foutpos	Output image	Spatial image
spatialr	Int	Spatial radius
ranger	Float	Range radius
thres	Float	Mode convergence threshold
maxiter	Int	Maximum number of iterations
rangeramp	Float	Range radius coefficient
modesearch	Boolean	Mode search.
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.33: Parameters table for Mean Shift filtering (can be used as Exact Large-Scale Mean-Shift segmentation, step 1).

- **Input Image:** The input image.
- **Filtered output:** The filtered output image.
- **Spatial image:** The spatial image output. Spatial image output is a displacement map (pixel position after convergence).
- **Spatial radius:** Spatial radius of the neighborhood.
- **Range radius:** Range radius defining the radius (expressed in radiometry unit) in the multi-spectral space.
- **Mode convergence threshold:** Algorithm iterative scheme will stop if mean-shift vector is below this threshold or if iteration number reached maximum number of iterations.
- **Maximum number of iterations:** Algorithm iterative scheme will stop if convergence hasn't been reached after the maximum number of iterations.
- **Range radius coefficient:** This coefficient makes dependent the ranger of the colorimetry of the filtered pixel : y = rangeramp*x+ranger.
- **Mode search.:** If activated pixel iterative convergence is stopped if the path . Be careful, with this option, the result will slightly depend on thread number
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_MeanShiftSmoothing -in maur_rgb.png -fout MeanShift_FilterOutput.tif -spatialr 16
     -ranger 16 -thres 0.1 -maxiter 100
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the MeanShiftSmoothing application
MeanShiftSmoothing = otbApplication.Registry.CreateApplication("MeanShiftSmoothing")

# The following lines set all the application parameters:
MeanShiftSmoothing.SetParameterString("in", "maur_rgb.png")

MeanShiftSmoothing.SetParameterString("fout", "MeanShift_FilterOutput.tif")

MeanShiftSmoothing.SetParameterInt("spatialr", 16)

MeanShiftSmoothing.SetParameterFloat("ranger", 16)

MeanShiftSmoothing.SetParameterFloat("thres", 0.1)

MeanShiftSmoothing.SetParameterInt("maxiter", 100)

# The following line execute the application
MeanShiftSmoothing.ExecuteAndWriteOutput()
```

Limitations

With mode search option, the result will slightly depend on thread number.

Authors

This application has been written by OTB-Team.

5.5.3 Smoothing

Apply a smoothing filter to an image

Detailed description

This application applies smoothing filter to an image. Either gaussian, mean, or anisotropic diffusion are available.

Parameters

This section describes in details the parameters available for this application. Table 5.34, page 180 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `Smoothing`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`out`	Output image	Output Image
`ram`	Int	Available RAM (Mb)
`type`	Choices	Smoothing Type
`type mean`	*Choice*	Mean
`type gaussian`	*Choice*	Gaussian
`type anidif`	*Choice*	Anisotropic Diffusion
`type.mean.radius`	Int	Radius
`type.gaussian.radius`	Float	Radius
`type.anidif.timestep`	Float	Time Step
`type.anidif.nbiter`	Int	Nb Iterations
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.34: Parameters table for Smoothing.

Input Image Input image to smooth.

Output Image Output smoothed image.

Available RAM (Mb) Available memory for processing (in MB)

Smoothing Type Smoothing kernel to apply Available choices are:

- **Mean**
  - **Radius:** Mean radius (in pixels)
- **Gaussian**
  - **Radius:** Gaussian radius (in pixels)

- **Anisotropic Diffusion**
  - **Time Step:** Diffusion equation time step
  - **Nb Iterations:** Number of iterations

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Examples

Example 1 Image smoothing using a mean filter.To run this example in command-line, use the following:

```
otbcli_Smoothing -in Romania_Extract.tif -out smoothedImage_mean.png uchar -type mean
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the Smoothing application
Smoothing = otbApplication.Registry.CreateApplication("Smoothing")

# The following lines set all the application parameters:
Smoothing.SetParameterString("in", "Romania_Extract.tif")

Smoothing.SetParameterString("out", "smoothedImage_mean.png")
Smoothing.SetParameterOutputImagePixelType("out", 1)

Smoothing.SetParameterString("type","mean")

# The following line execute the application
Smoothing.ExecuteAndWriteOutput()
```

Example 2 Image smoothing using an anisotropic diffusion filter.To run this example in command-line, use the following:

```
otbcli_Smoothing -in Romania_Extract.tif -out smoothedImage_ani.png float -type anidif
     -type.anidif.timestep 0.1 -type.anidif.nbiter 5
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication
```

```
# The following line creates an instance of the Smoothing application
Smoothing = otbApplication.Registry.CreateApplication("Smoothing")

# The following lines set all the application parameters:
Smoothing.SetParameterString("in", "Romania_Extract.tif")

Smoothing.SetParameterString("out", "smoothedImage_ani.png")
Smoothing.SetParameterOutputImagePixelType("out", 6)

Smoothing.SetParameterString("type","anidif")

Smoothing.SetParameterFloat("type.anidif.timestep", 0.1)

Smoothing.SetParameterInt("type.anidif.nbiter", 5)

# The following line execute the application
Smoothing.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.6 Feature Extraction

5.6.1 Binary Morphological Operation

Performs morphological operations on an input image channel

Detailed description

This application performs binary morphological operations on a mono band image

Parameters

This section describes in details the parameters available for this application. Table 5.35, page 183 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `BinaryMorphologicalOperation`.

Parameter key	Parameter type	Parameter description
in	Input image	Input Image
out	Output image	Feature Output Image
channel	Int	Selected Channel
ram	Int	Available RAM (Mb)
structype	Choices	Structuring Element Type
structype ball	*Choice*	Ball
structype cross	*Choice*	Cross
structype.ball.xradius	Int	The Structuring Element X Radius
structype.ball.yradius	Int	The Structuring Element Y Radius
filter	Choices	Morphological Operation
filter dilate	*Choice*	Dilate
filter erode	*Choice*	Erode
filter opening	*Choice*	Opening
filter closing	*Choice*	Closing
filter.dilate.foreval	Float	Foreground Value
filter.dilate.backval	Float	Background Value
filter.erode.foreval	Float	Foreground Value
filter.erode.backval	Float	Background Value
filter.opening.foreval	Float	Foreground Value
filter.opening.backval	Float	Background Value
filter.closing.foreval	Float	Foreground Value
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.35: Parameters table for Binary Morphological Operation.

Input Image The input image to be filtered.

Feature Output Image Output image containing the filtered output image.

Selected Channel The selected channel index

Available RAM (Mb) Available memory for processing (in MB)

Structuring Element Type Choice of the structuring element type Available choices are:

- **Ball**

  - **The Structuring Element X Radius:** The Structuring Element X Radius
  - **The Structuring Element Y Radius:** The Structuring Element Y Radius
- **Cross**

Morphological Operation Choice of the morphological operation Available choices are:

- **Dilate**
  - **Foreground Value:** The Foreground Value
  - **Background Value:** The Background Value
- **Erode**
  - **Foreground Value:** The Foreground Value
  - **Background Value:** The Background Value
- **Opening**
  - **Foreground Value:** The Foreground Value
  - **Background Value:** The Background Value
- **Closing**
  - **Foreground Value:** The Foreground Value

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_BinaryMorphologicalOperation -in qb_RoadExtract.tif -out opened.tif -channel 1
     -structype.ball.xradius 5 -structype.ball.yradius 5 -filter erode
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the BinaryMorphologicalOperation application
```

```
BinaryMorphologicalOperation =
     otbApplication.Registry.CreateApplication("BinaryMorphologicalOperation")

# The following lines set all the application parameters:
BinaryMorphologicalOperation.SetParameterString("in", "qb_RoadExtract.tif")

BinaryMorphologicalOperation.SetParameterString("out", "opened.tif")

BinaryMorphologicalOperation.SetParameterInt("channel", 1)

BinaryMorphologicalOperation.SetParameterInt("structype.ball.xradius", 5)

BinaryMorphologicalOperation.SetParameterInt("structype.ball.yradius", 5)

BinaryMorphologicalOperation.SetParameterString("filter","erode")

# The following line execute the application
BinaryMorphologicalOperation.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- itkBinaryDilateImageFilter, itkBinaryErodeImageFilter, itkBinaryMorphologicalOpeningImageFilter and itkBinaryMorphologicalClosingImageFilter classes

5.6.2 Compute Polyline Feature From Image

This application compute for each studied polyline, contained in the input VectorData, the choosen descriptors.

Detailed description

The first step in the classifier fusion based validation is to compute, for each studied polyline, the choosen descriptors.

Parameters

This section describes in details the parameters available for this application. Table 5.36, page 186 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `ComputePolylineFeatureFromImage`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`vd`	Input vector data	Vector Data
`elev`	Group	Elevation management
`elev.dem`	Directory	DEM directory
`elev.geoid`	Input File name	Geoid File
`elev.default`	Float	Default elevation
`expr`	String	Feature expression
`field`	String	Feature name
`out`	Output vector data	Output Vector Data
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.36: Parameters table for Compute Polyline Feature From Image.

Input Image An image to compute the descriptors on.

Vector Data Vector data containing the polylines where the features will be computed.

Elevation management This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).
- **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles,

and no geoid file has been set. This is also used by some application as an average elevation value.

Feature expression The feature formula (b1 <0.3) where b1 is the standard name of input image first band

Feature name The field name corresponding to the feature codename (NONDVI, ROADSA...)

Output Vector Data The output vector data containing polylines with a new field

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_ComputePolylineFeatureFromImage -in NDVI.TIF -vd roads_ground_truth.shp -expr "(b1 >
    0.4)" -field NONDVI -out PolylineFeatureFromImage_LI_NONDVI_gt.shp
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the ComputePolylineFeatureFromImage application
ComputePolylineFeatureFromImage =
    otbApplication.Registry.CreateApplication("ComputePolylineFeatureFromImage")

# The following lines set all the application parameters:
ComputePolylineFeatureFromImage.SetParameterString("in", "NDVI.TIF")

ComputePolylineFeatureFromImage.SetParameterString("vd", "roads_ground_truth.shp")

ComputePolylineFeatureFromImage.SetParameterString("expr", "(b1 > 0.4)")

ComputePolylineFeatureFromImage.SetParameterString("field", "NONDVI")

ComputePolylineFeatureFromImage.SetParameterString("out",
    "PolylineFeatureFromImage_LI_NONDVI_gt.shp")

# The following line execute the application
ComputePolylineFeatureFromImage.ExecuteAndWriteOutput()
```

Limitations

Since it does not rely on streaming process, take care of the size of input image before launching application.

Authors

This application has been written by OTB-Team.

5.6.3 Fuzzy Model estimation

Estimate feature fuzzy model parameters using 2 vector data (ground truth samples and wrong samples).

Detailed description

Estimate feature fuzzy model parameters using 2 vector data (ground truth samples and wrong samples).

Parameters

This section describes in details the parameters available for this application. Table 5.37, page 189 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `DSFuzzyModelEstimation`.

Parameter key	Parameter type	Parameter description
`psin`	Input vector data	Input Positive Vector Data
`nsin`	Input vector data	Input Negative Vector Data
`belsup`	String list	Belief Support
`plasup`	String list	Plausibility Support
`cri`	String	Criterion
`wgt`	Float	Weighting
`initmod`	Input File name	initialization model
`desclist`	String list	Descriptor list
`maxnbit`	Int	Maximum number of iterations
`optobs`	Boolean	Optimizer Observer
`out`	Output File name	Output filename
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.37: Parameters table for Fuzzy Model estimation.

- **Input Positive Vector Data:** Ground truth vector data for positive samples
- **Input Negative Vector Data:** Ground truth vector data for negative samples
- **Belief Support:** Dempster Shafer study hypothesis to compute belief
- **Plausibility Support:** Dempster Shafer study hypothesis to compute plausibility
- **Criterion:** Dempster Shafer criterion (by default (belief+plausibility)/2)
- **Weighting:** Coefficient between 0 and 1 to promote undetection or false detections (default 0.5)
- **initialization model:** Initialization model (xml file) to be used. If the xml initialization model is set, the descriptor list is not used (specified using the option -desclist)
- **Descriptor list:** List of the descriptors to be used in the model (must be specified to perform an automatic initialization)
- **Maximum number of iterations:** Maximum number of optimizer iteration (default 200)
- **Optimizer Observer:** Activate the optimizer observer
- **Output filename:** Output model file name (xml file) contains the optimal model to perform informations fusion.
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_DSFuzzyModelEstimation -psin cdbTvComputePolylineFeatureFromImage_LI_NOBUIL_gt.shp
     -nsin cdbTvComputePolylineFeatureFromImage_LI_NOBUIL_wr.shp -belsup "ROADSA" -plasup
     "NONDVI" "ROADSA" "NOBUIL" -initmod Dempster-Shafer/DSFuzzyModel_Init.xml -maxnbit 4
     -optobs true -out DSFuzzyModelEstimation.xml
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the DSFuzzyModelEstimation application
DSFuzzyModelEstimation = otbApplication.Registry.CreateApplication("DSFuzzyModelEstimation")

# The following lines set all the application parameters:
DSFuzzyModelEstimation.SetParameterString("psin",
     "cdbTvComputePolylineFeatureFromImage_LI_NOBUIL_gt.shp")

DSFuzzyModelEstimation.SetParameterString("nsin",
     "cdbTvComputePolylineFeatureFromImage_LI_NOBUIL_wr.shp")

DSFuzzyModelEstimation.SetParameterStringList("belsup", ['"ROADSA"'])

DSFuzzyModelEstimation.SetParameterStringList("plasup", ['"NONDVI"', '"ROADSA"', '"NOBUIL"'])

DSFuzzyModelEstimation.SetParameterString("initmod", "Dempster-Shafer/DSFuzzyModel_Init.xml")

DSFuzzyModelEstimation.SetParameterInt("maxnbit", 4)

DSFuzzyModelEstimation.SetParameterString("optobs","1")

DSFuzzyModelEstimation.SetParameterString("out", "DSFuzzyModelEstimation.xml")

# The following line execute the application
DSFuzzyModelEstimation.ExecuteAndWriteOutput()
```

Limitations

None.

Authors

This application has been written by OTB-Team.

5.6.4 Edge Feature Extraction

Computes edge features on every pixel of the input image selected channel

Detailed description

This application computes edge features on a mono band image

Parameters

This section describes in details the parameters available for this application. Table 5.38, page 191 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `EdgeExtraction`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`channel`	Int	Selected Channel
`ram`	Int	Available RAM (Mb)
`filter`	Choices	Edge feature
`filter gradient`	*Choice*	Gradient
`filter sobel`	*Choice*	Sobel
`filter touzi`	*Choice*	Touzi
`filter.touzi.xradius`	Int	The X Radius
`filter.touzi.yradius`	Int	The Y Radius
`out`	Output image	Feature Output Image
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.38: Parameters table for Edge Feature Extraction.

Input Image The input image to compute the features on.

Selected Channel The selected channel index

Available RAM (Mb) Available memory for processing (in MB)

Edge feature Choice of edge feature Available choices are:

- **Gradient**
- **Sobel**
- **Touzi**
  - **The X Radius:**
  - **The Y Radius:**

Feature Output Image Output image containing the edge features.

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_EdgeExtraction -in qb_RoadExtract.tif -channel 1 -out Edges.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the EdgeExtraction application
EdgeExtraction = otbApplication.Registry.CreateApplication("EdgeExtraction")

# The following lines set all the application parameters:
EdgeExtraction.SetParameterString("in", "qb_RoadExtract.tif")

EdgeExtraction.SetParameterInt("channel", 1)

EdgeExtraction.SetParameterString("out", "Edges.tif")

# The following line execute the application
EdgeExtraction.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- otb class

5.6.5 Grayscale Morphological Operation

Performs morphological operations on a grayscale input image

Detailed description

This application performs grayscale morphological operations on a mono band image

Parameters

This section describes in details the parameters available for this application. Table 5.39, page 193 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `GrayScaleMorphologicalOperation`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`out`	Output image	Feature Output Image
`channel`	Int	Selected Channel
`ram`	Int	Available RAM (Mb)
`structype`	Choices	Structuring Element Type
`structype ball`	*Choice*	Ball
`structype cross`	*Choice*	Cross
`structype.ball.xradius`	Int	The Structuring Element X Radius
`structype.ball.yradius`	Int	The Structuring Element Y Radius
`filter`	Choices	Morphological Operation
`filter dilate`	*Choice*	Dilate
`filter erode`	*Choice*	Erode
`filter opening`	*Choice*	Opening
`filter closing`	*Choice*	Closing
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.39: Parameters table for Grayscale Morphological Operation.

Input Image The input image to be filtered.

Feature Output Image Output image containing the filtered output image.

Selected Channel The selected channel index

Available RAM (Mb) Available memory for processing (in MB)

Structuring Element Type Choice of the structuring element type Available choices are:

- **Ball**
  - **The Structuring Element X Radius:** The Structuring Element X Radius
  - **The Structuring Element Y Radius:** The Structuring Element Y Radius
- **Cross**

Morphological Operation Choice of the morphological operation Available choices are:

- **Dilate**
- **Erode**
- **Opening**
- **Closing**

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_GrayScaleMorphologicalOperation -in qb_RoadExtract.tif -out opened.tif -channel 1
     -structype.ball.xradius 5 -structype.ball.yradius 5 -filter erode
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the GrayScaleMorphologicalOperation application
GrayScaleMorphologicalOperation =
     otbApplication.Registry.CreateApplication("GrayScaleMorphologicalOperation")
```

```
# The following lines set all the application parameters:
GrayScaleMorphologicalOperation.SetParameterString("in", "qb_RoadExtract.tif")

GrayScaleMorphologicalOperation.SetParameterString("out", "opened.tif")

GrayScaleMorphologicalOperation.SetParameterInt("channel", 1)

GrayScaleMorphologicalOperation.SetParameterInt("structype.ball.xradius", 5)

GrayScaleMorphologicalOperation.SetParameterInt("structype.ball.yradius", 5)

GrayScaleMorphologicalOperation.SetParameterString("filter","erode")

# The following line execute the application
GrayScaleMorphologicalOperation.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- itkGrayscaleDilateImageFilter, itkGrayscaleErodeImageFilter, itkGrayscaleMorphologicalOpeningImageFilter and itkGrayscaleMorphologicalClosingImageFilter classes

5.6.6 Haralick Texture Extraction

Computes textures on every pixel of the input image selected channel

Detailed description

This application computes Haralick, advanced and higher order textures on a mono band image

Parameters

This section describes in details the parameters available for this application. Table 5.40, page 196 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `HaralickTextureExtraction`.

Parameter key	Parameter type	Parameter description
in	Input image	Input Image
channel	Int	Selected Channel
ram	Int	Available RAM (Mb)
parameters	Group	Texture feature parameters
parameters.xrad	Int	X Radius
parameters.yrad	Int	Y Radius
parameters.xoff	Int	X Offset
parameters.yoff	Int	Y Offset
parameters.min	Float	Image Minimum
parameters.max	Float	Image Maximum
parameters.nbbin	Int	Histogram number of bin
texture	Choices	Texture Set Selection
texture simple	*Choice*	Simple Haralick Texture Features
texture advanced	*Choice*	Advanced Texture Features
texture higher	*Choice*	Higher Order Texture Features
out	Output image	Output Image
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.40: Parameters table for Haralick Texture Extraction.

Input Image The input image to compute the features on.

Selected Channel The selected channel index

Available RAM (Mb) Available memory for processing (in MB)

Texture feature parameters This group of parameters allows to define texture parameters.

- **X Radius:** X Radius
- **Y Radius:** Y Radius
- **X Offset:** X Offset
- **Y Offset:** Y Offset
- **Image Minimum:** Image Minimum

- **Image Maximum:** Image Maximum
- **Histogram number of bin:** Histogram number of bin

Texture Set Selection Choice of The Texture Set Available choices are:

- **Simple Haralick Texture Features**: This group of parameters defines the 8 local Haralick texture feature output image. The image channels are: Energy, Entropy, Correlation, Inverse Difference Moment, Inertia, Cluster Shade, Cluster Prominence and Haralick Correlation
- **Advanced Texture Features**: This group of parameters defines the 9 advanced texture feature output image. The image channels are: Mean, Variance, Sum Average, Sum Variance, Sum Entropy, Difference of Entropies, Difference of Variances, IC1 and IC2
- **Higher Order Texture Features**: This group of parameters defines the 11 higher order texture feature output image. The image channels are: Short Run Emphasis, Long Run Emphasis, Grey-Level Nonuniformity, Run Length Nonuniformity, Run Percentage, Low Grey-Level Run Emphasis, High Grey-Level Run Emphasis, Short Run Low Grey-Level Emphasis, Short Run High Grey-Level Emphasis, Long Run Low Grey-Level Emphasis and Long Run High Grey-Level Emphasis

Output Image Output image containing the selected texture features.

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_HaralickTextureExtraction -in qb_RoadExtract.tif -channel 2 -parameters.xrad 3
     -parameters.yrad 3 -texture simple -out HaralickTextures.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the HaralickTextureExtraction application
HaralickTextureExtraction =
     otbApplication.Registry.CreateApplication("HaralickTextureExtraction")
```

```
# The following lines set all the application parameters:
HaralickTextureExtraction.SetParameterString("in", "qb_RoadExtract.tif")

HaralickTextureExtraction.SetParameterInt("channel", 2)

HaralickTextureExtraction.SetParameterInt("parameters.xrad", 3)

HaralickTextureExtraction.SetParameterInt("parameters.yrad", 3)

HaralickTextureExtraction.SetParameterString("texture","simple")

HaralickTextureExtraction.SetParameterString("out", "HaralickTextures.tif")

# The following line execute the application
HaralickTextureExtraction.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- otbScalarImageToTexturesFilter, otbScalarImageToAdvancedTexturesFilter and otbScalarImageToHigherOrderTexturesFilter classes

5.6.7 Homologous Points Extraction

Allows to compute homologous points between images using keypoints

Detailed description

This application allows to compute homologous points between images using keypoints. SIFT or SURF keypoints can be used and the band on which keypoints are computed can be set independantly for both images. The application offers two modes : the first is the full mode where keypoints are extracted from the full extent of both images (please note that in this mode large image file are not supported). The second mode, called geobins, allows to set-up spatial binning to get fewer points spread accross the entire image. In this mode, the corresponding spatial bin in the second image is estimated using geographical transform or sensor modelling, and is padded according to the user

defined precision. Last, in both modes the application can filter matches whose colocalisation in first image exceed this precision. The elevation parameters are to deal more precisely with sensor modelling in case of sensor geometry data. The outvector option allows to create a vector file with segments corresponding to the localisation error between the matches. It can be useful to assess the precision of a registration for instance. The vector file is always reprojected to EPSG:4326 to allow display in a GIS. This is done via reprojection or by applying the image sensor models.

Parameters

This section describes in details the parameters available for this application. Table 5.41, page 200 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `HomologousPointsExtraction`.

Parameter key	Parameter type	Parameter description
`in1`	Input image	Input Image 1
`band1`	Int	Input band 1
`in2`	Input image	Input Image 2
`band2`	Int	Input band 2
`algorithm`	Choices	Keypoints detection algorithm
`algorithm surf`	*Choice*	SURF algorithm
`algorithm sift`	*Choice*	SIFT algorithm
`threshold`	Float	Distance threshold for matching
`backmatching`	Boolean	Use back-matching to filter matches.
`mode`	Choices	Keypoints search mode
`mode full`	*Choice*	Extract and match all keypoints (no streaming)
`mode geobins`	*Choice*	Search keypoints in small spatial bins regularly spread accross first image
`mode.geobins.binsize`	Int	Size of bin
`mode.geobins.binsizey`	Int	Size of bin (y direction)
`mode.geobins.binstep`	Int	Steps between bins
`mode.geobins.binstepy`	Int	Steps between bins (y direction)
`mode.geobins.margin`	Int	Margin from image border to start/end bins (in pixels)
`precision`	Float	Estimated precision of the colocalisation function (in pixels).
`mfilter`	Boolean	Filter points according to geographical or sensor based colocalisation
`2wgs84`	Boolean	If enabled, points from second image will be exported in WGS84
`elev`	Group	Elevation management
`elev.dem`	Directory	DEM directory
`elev.geoid`	Input File name	Geoid File
`elev.default`	Float	Default elevation
.../...		

Parameter key	Parameter type	Parameter description
.../...		
out	Output File name	Output file with tie points
outvector	Output File name	Output vector file with tie points
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.41: Parameters table for Homologous Points Extraction.

Input Image 1 First input image

Input band 1 Index of the band from input image 1 to use for keypoints extraction

Input Image 2 Second input image

Input band 2 Index of the band from input image 1 to use for keypoints extraction

Keypoints detection algorithm Choice of the detection algorithm to use Available choices are:

- **SURF algorithm**
- **SIFT algorithm**

Distance threshold for matching The distance threshold for matching.

Use back-matching to filter matches. If set to true, matches should be consistent in both ways.

Keypoints search mode Available choices are:

- **Extract and match all keypoints (no streaming)**: Extract and match all keypoints, loading both images entirely into memory
- **Search keypoints in small spatial bins regularly spread accross first image**: This method allows to retrieve a set of tie points regulary spread accross image 1. Corresponding bins in image 2 are retrieved using sensor and geographical information if available. The first bin position takes into account the margin parameter. Bins are cropped to the largest image region shrinked by the margin parameter for both in1 and in2 images.

- **Size of bin:** Radius of the spatial bin in pixels
- **Size of bin (y direction):** Radius of the spatial bin in pixels (y direction). If not set, the mode.geobins.binsize value is used.
- **Steps between bins:** Steps between bins in pixels
- **Steps between bins (y direction):** Steps between bins in pixels (y direction). If not set, the mode.geobins.binstep value is used.
- **Margin from image border to start/end bins (in pixels):** Margin from image border to start/end bins (in pixels)

Estimated precision of the colocalisation function (in pixels). Estimated precision of the colocalisation function in pixels

Filter points according to geographical or sensor based colocalisation If enabled, this option allows to filter matches according to colocalisation from sensor or geographical information, using the given tolerancy expressed in pixels

If enabled, points from second image will be exported in WGS84

Elevation management This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).
- **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

Output file with tie points File containing the list of tie points

Output vector file with tie points File containing segments representing matches

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_HomologousPointsExtraction -in1 sensor_stereo_left.tif -in2 sensor_stereo_right.tif
     -mode full -out homologous.txt
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the HomologousPointsExtraction application
HomologousPointsExtraction =
     otbApplication.Registry.CreateApplication("HomologousPointsExtraction")

# The following lines set all the application parameters:
HomologousPointsExtraction.SetParameterString("in1", "sensor_stereo_left.tif")

HomologousPointsExtraction.SetParameterString("in2", "sensor_stereo_right.tif")

HomologousPointsExtraction.SetParameterString("mode","full")

HomologousPointsExtraction.SetParameterString("out", "homologous.txt")

# The following line execute the application
HomologousPointsExtraction.ExecuteAndWriteOutput()
```

Limitations

Full mode does not handle large images.

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- RefineSensorModel

5.6.8 Line segment detection

Detect line segments in raster

Detailed description

This application detects locally straight contours in a image. It is based on Burns, Hanson, and Riseman method and use an a contrario validation approach (Desolneux, Moisan, and Morel). The algorithm was published by Rafael Gromponevon Gioi, JÃ©rÃ©mie Jakubowicz, Jean-Michel Morel and Gregory Randall.
The given approach computes gradient and level lines of the image and detects aligned points in line support region. The application allows to export the detected lines in a vector data.

Parameters

This section describes in details the parameters available for this application. Table 5.42, page 203 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `LineSegmentDetection`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`out`	Output vector data	Output Detected lines
`elev`	Group	Elevation management
`elev.dem`	Directory	DEM directory
`elev.geoid`	Input File name	Geoid File
`elev.default`	Float	Default elevation
`norescale`	Boolean	No rescaling in [0, 255]
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.42: Parameters table for Line segment detection.

Input Image Input image on which lines will be detected.

Output Detected lines Output detected line segments (vector data).

Elevation management This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).
- **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

No rescaling in [0, 255] By default, the input image amplitude is rescaled between [0,255]. Turn on this parameter to skip rescaling

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_LineSegmentDetection -in QB_Suburb.png -out LineSegmentDetection.shp
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the LineSegmentDetection application
LineSegmentDetection = otbApplication.Registry.CreateApplication("LineSegmentDetection")

# The following lines set all the application parameters:
LineSegmentDetection.SetParameterString("in", "QB_Suburb.png")

LineSegmentDetection.SetParameterString("out", "LineSegmentDetection.shp")

# The following line execute the application
LineSegmentDetection.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- On Line demonstration of the LSD algorithm is available here: http://www.ipol.im/pub/algo/gjmr_line_segment_detector/

5.6.9 Local Statistic Extraction

Computes local statistical moments on every pixel in the selected channel of the input image

Detailed description

This application computes the 4 local statistical moments on every pixel in the selected channel of the input image, over a specified neighborhood. The output image is multi band with one statistical moment (feature) per band. Thus, the 4 output features are the Mean, the Variance, the Skewness and the Kurtosis. They are provided in this exact order in the output image.

Parameters

This section describes in details the parameters available for this application. Table 5.43, page 206 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `LocalStatisticExtraction`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`channel`	Int	Selected Channel
`ram`	Int	Available RAM (Mb)
`radius`	Int	Neighborhood radius
`out`	Output image	Feature Output Image
.../...		

Parameter key	Parameter type	Parameter description
.../...		
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.43: Parameters table for Local Statistic Extraction.

- **Input Image:** The input image to compute the features on.
- **Selected Channel:** The selected channel index
- **Available RAM (Mb):** Available memory for processing (in MB)
- **Neighborhood radius:** The computational window radius.
- **Feature Output Image:** Output image containing the local statistical moments.
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_LocalStatisticExtraction -in qb_RoadExtract.tif -channel 1 -radius 3 -out
     Statistics.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the LocalStatisticExtraction application
LocalStatisticExtraction =
     otbApplication.Registry.CreateApplication("LocalStatisticExtraction")

# The following lines set all the application parameters:
LocalStatisticExtraction.SetParameterString("in", "qb_RoadExtract.tif")

LocalStatisticExtraction.SetParameterInt("channel", 1)

LocalStatisticExtraction.SetParameterInt("radius", 3)

LocalStatisticExtraction.SetParameterString("out", "Statistics.tif")

# The following line execute the application
LocalStatisticExtraction.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- otbRadiometricMomentsImageFunction class

5.6.10 Multivariate alteration detector

Multivariate Alteration Detector

Detailed description

This application detects change between two given images.

Parameters

This section describes in details the parameters available for this application. Table 5.44, page 208 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `MultivariateAlterationDetector`.

Parameter key	Parameter type	Parameter description
`in1`	Input image	Input Image 1
`in2`	Input image	Input Image 2
`out`	Output image	Change Map
`ram`	Int	Available RAM (Mb)
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.44: Parameters table for Multivariate alteration detector.

- **Input Image 1:** Image which describe initial state of the scene.
- **Input Image 2:** Image which describe scene after perturbations.
- **Change Map:** Image of detected changes.
- **Available RAM (Mb):** Available memory for processing (in MB)
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_MultivariateAlterationDetector -in1 Spot5-Gloucester-before.tif -in2
     Spot5-Gloucester-after.tif -out detectedChangeImage.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the MultivariateAlterationDetector application
MultivariateAlterationDetector =
     otbApplication.Registry.CreateApplication("MultivariateAlterationDetector")

# The following lines set all the application parameters:
MultivariateAlterationDetector.SetParameterString("in1", "Spot5-Gloucester-before.tif")

MultivariateAlterationDetector.SetParameterString("in2", "Spot5-Gloucester-after.tif")

MultivariateAlterationDetector.SetParameterString("out", "detectedChangeImage.tif")

# The following line execute the application
MultivariateAlterationDetector.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- This filter implements the Multivariate Alteration Detector, based on the following work:
  A. A. Nielsen and K. Conradsen, Multivariate alteration detection (mad) in multispectral, bi-temporal image data: a new approach to change detection studies, Remote Sens. Environ., vol. 64, pp. 1-19, (1998)

  Multivariate Alteration Detector takes two images as inputs and produce a set of N change maps as a VectorImage (where N is the maximum of number of bands in first and second image) with the following properties:
  - Change maps are differences of a pair of linear combinations of bands from image 1 and bands from image 2 chosen to maximize the correlation.
  - Each change map is orthogonal to the others.

  This is a statistical method which can handle different modalities and even different bands and number of bands between images.

  If numbers of bands in image 1 and 2 are equal, then change maps are sorted by increasing correlation. If number of bands is different, the change maps are sorted by decreasing correlation.

  The GetV1() and GetV2() methods allow to retrieve the linear combinations used to generate the Mad change maps as a vnl_matrix of double, and the GetRho() method allows to retrieve the correlation associated to each Mad change maps as a vnl_vector.

  This filter has been implemented from the Matlab code kindly made available by the authors here:
  http://www2.imm.dtu.dk/ aa/software.html

  Both cases (same and different number of bands) have been validated by comparing the output image to the output produced by the Matlab code, and the reference images for testing have been generated from the Matlab code using Octave.

5.6.11 Radiometric Indices

Compute radiometric indices.

Detailed description

This application computes radiometric indices using the relevant channels of the input image. The output is a multi band image into which each channel is one of the selected indices.

Parameters

This section describes in details the parameters available for this application. Table 5.45, page 210 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `RadiometricIndices`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`out`	Output image	Output Image
`ram`	Int	Available RAM (Mb)
`channels`	Group	Channels selection
`channels.blue`	Int	Blue Channel
`channels.green`	Int	Green Channel
`channels.red`	Int	Red Channel
`channels.nir`	Int	NIR Channel
`channels.mir`	Int	Mir Channel
`list`	List	Available Radiometric Indices
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.45: Parameters table for Radiometric Indices.

Input Image Input image

Output Image Radiometric indices output image

Available RAM (Mb) Available memory for processing (in MB)

Channels selection Channels selection

- **Blue Channel:** Blue channel index

- **Green Channel:** Green channel index
- **Red Channel:** Red channel index
- **NIR Channel:** NIR channel index
- **Mir Channel:** Mir channel index

Available Radiometric Indices List of available radiometric indices with their relevant channels in brackets:
Vegetation:NDVI - Normalized difference vegetation index (Red, NIR)
Vegetation:TNDVI - Transformed normalized difference vegetation index (Red, NIR)
Vegetation:RVI - Ratio vegetation index (Red, NIR)
Vegetation:SAVI - Soil adjusted vegetation index (Red, NIR)
Vegetation:TSAVI - Transformed soil adjusted vegetation index (Red, NIR)
Vegetation:MSAVI - Modified soil adjusted vegetation index (Red, NIR)
Vegetation:MSAVI2 - Modified soil adjusted vegetation index 2 (Red, NIR)
Vegetation:GEMI - Global environment monitoring index (Red, NIR)
Vegetation:IPVI - Infrared percentage vegetation index (Red, NIR)

Water:NDWI - Normalized difference water index (Gao 1996) (NIR, MIR)
Water:NDWI2 - Normalized difference water index (Mc Feeters 1996) (Green, NIR)
Water:MNDWI - Modified normalized difference water index (Xu 2006) (Green, MIR)
Water:NDPI - Normalized difference pond index (Lacaux et al.) (MIR, Green)
Water:NDTI - Normalized difference turbidity index (Lacaux et al.) (Red, Green)

Soil:RI - Redness index (Red, Green)
Soil:CI - Color index (Red, Green)
Soil:BI - Brightness index (Red, Green)
Soil:BI2 - Brightness index 2 (NIR, Red, Green)

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_RadiometricIndices -in qb_RoadExtract.tif -list Vegetation:NDVI Vegetation:RVI
     Vegetation:IPVI -out RadiometricIndicesImage.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the RadiometricIndices application
RadiometricIndices = otbApplication.Registry.CreateApplication("RadiometricIndices")

# The following lines set all the application parameters:
RadiometricIndices.SetParameterString("in", "qb_RoadExtract.tif")

# The following line execute the application
RadiometricIndices.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- otbVegetationIndicesFunctor, otbWaterIndicesFunctor and otbSoilIndicesFunctor classes

5.6.12 SFS Texture Extraction

Computes Structural Feature Set textures on every pixel of the input image selected channel

Detailed description

This application computes SFS textures on a mono band image

Parameters

This section describes in details the parameters available for this application. Table 5.46, page 213 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `SFSTextureExtraction`.

Parameter key	Parameter type	Parameter description
in	Input image	Input Image
channel	Int	Selected Channel
ram	Int	Available RAM (Mb)
parameters	Group	Texture feature parameters
parameters.spethre	Float	Spectral Threshold
parameters.spathre	Int	Spatial Threshold
parameters.nbdir	Int	Number of Direction
parameters.alpha	Float	Alpha
parameters.maxcons	Int	Ratio Maximum Consideration Number
out	Output image	Feature Output Image
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.46: Parameters table for SFS Texture Extraction.

Input Image The input image to compute the features on.

Selected Channel The selected channel index

Available RAM (Mb) Available memory for processing (in MB)

Texture feature parameters This group of parameters allows to define SFS texture parameters. The available texture features are SFS'Length, SFS'Width, SFS'PSI, SFS'W-Mean, SFS'Ratio and SFS'SD. They are provided in this exact order in the output image.

- **Spectral Threshold:** Spectral Threshold
- **Spatial Threshold:** Spatial Threshold
- **Number of Direction:** Number of Direction
- **Alpha:** Alpha
- **Ratio Maximum Consideration Number:** Ratio Maximum Consideration Number

Feature Output Image Output image containing the SFS texture features.

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_SFSTextureExtraction -in qb_RoadExtract.tif -channel 1 -parameters.spethre 50.0
     -parameters.spathre 100 -out SFSTextures.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the SFSTextureExtraction application
SFSTextureExtraction = otbApplication.Registry.CreateApplication("SFSTextureExtraction")

# The following lines set all the application parameters:
SFSTextureExtraction.SetParameterString("in", "qb_RoadExtract.tif")

SFSTextureExtraction.SetParameterInt("channel", 1)

SFSTextureExtraction.SetParameterFloat("parameters.spethre", 50.0)

SFSTextureExtraction.SetParameterInt("parameters.spathre", 100)

SFSTextureExtraction.SetParameterString("out", "SFSTextures.tif")

# The following line execute the application
SFSTextureExtraction.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- otbSFSTexturesImageFilter class

5.6.13 Vector Data validation

Vector data validation based on the fusion of features using Dempster-Shafer evidence theory framework.

Detailed description

This application validates or unvalidate the studied samples using the Dempster-Shafer theory.

Parameters

This section describes in details the parameters available for this application. Table 5.47, page 215 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `VectorDataDSValidation`.

Parameter key	Parameter type	Parameter description
`in`	Input vector data	Input Vector Data
`descmod`	Input File name	Descriptors model filename
`belsup`	String list	Belief Support
`plasup`	String list	Plausibility Support
`cri`	String	Criterion
`thd`	Float	Criterion threshold
`out`	Output vector data	Output Vector Data
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.47: Parameters table for Vector Data validation.

- **Input Vector Data:** Input vector data to validate
- **Descriptors model filename:** Fuzzy descriptors model (xml file)
- **Belief Support:** Dempster Shafer study hypothesis to compute belief
- **Plausibility Support:** Dempster Shafer study hypothesis to compute plausibility
- **Criterion:** Dempster Shafer criterion (by default (belief+plausibility)/2)
- **Criterion threshold:** Criterion threshold (default 0.5)

- **Output Vector Data:** Output VectorData containing only the validated samples
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_VectorDataDSValidation -in cdbTvComputePolylineFeatureFromImage_LI_NOBUIL_gt.shp
     -belsup cdbTvComputePolylineFeatureFromImage_LI_NOBUIL_gt.shp -descmod DSFuzzyModel.xml
     -out VectorDataDSValidation.shp
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the VectorDataDSValidation application
VectorDataDSValidation = otbApplication.Registry.CreateApplication("VectorDataDSValidation")

# The following lines set all the application parameters:
VectorDataDSValidation.SetParameterString("in",
     "cdbTvComputePolylineFeatureFromImage_LI_NOBUIL_gt.shp")

VectorDataDSValidation.SetParameterStringList("belsup",
     ['cdbTvComputePolylineFeatureFromImage_LI_NOBUIL_gt.shp'])

VectorDataDSValidation.SetParameterString("descmod", "DSFuzzyModel.xml")

VectorDataDSValidation.SetParameterString("out", "VectorDataDSValidation.shp")

# The following line execute the application
VectorDataDSValidation.ExecuteAndWriteOutput()
```

Limitations

None.

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- http://en.wikipedia.org/wiki/Dempster-Shafer_theory

5.7 Stereo

5.7.1 Pixel-wise Block-Matching

Performs block-matching to estimate pixel-wise disparities between two images

Detailed description

This application allows to performs block-matching to estimate pixel-wise disparities between two images. The application allows to choose the block-matching method to use. It also allows to input masks (related to the left and right input image) of pixels for which the disparity should be investigated. Additionally, two criteria can be optionally used to disable disparity investigation for some pixel: a no-data value, and a threshold on the local variance. This allows to speed-up computation by avoiding to investigate disparities that will not be reliable anyway. For efficiency reasons, if the optimal metric values image is desired, it will be concatenated to the output image (which will then have three bands : horizontal disparity, vertical disparity and metric value). One can split these images afterward.

Parameters

This section describes in details the parameters available for this application. Table 5.48, page 218 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `BlockMatching`.

Parameter key	Parameter type	Parameter description
`io`	Group	Input and output data
`io.inleft`	Input image	Left input image
`io.inright`	Input image	Right input image
`io.out`	Output image	The output disparity map
`io.outmask`	Output image	The output mask corresponding to all criterions
`io.outmetric`	Boolean	Output optimal metric values as well
`mask`	Group	Image masking parameters
`mask.inleft`	Input image	Discard left pixels from mask image
`mask.inright`	Input image	Discard right pixels from mask image
`mask.nodata`	Float	Discard pixels with no-data value
`mask.variancet`	Float	Discard pixels with low local variance
`bm`	Group	Block matching parameters
.../...		

Parameter key	Parameter type	Parameter description
.../...		
bm.metric	Choices	Block-matching metric
bm.metric ssd	*Choice*	Sum of Squared Distances
bm.metric ncc	*Choice*	Normalized Cross-Correlation
bm.metric lp	*Choice*	Lp pseudo-norm
bm.metric.lp.p	Float	p value
bm.radius	Int	Radius of blocks
bm.minhd	Int	Minimum horizontal disparity
bm.maxhd	Int	Maximum horizontal disparity
bm.minvd	Int	Minimum vertical disparity
bm.maxvd	Int	Maximum vertical disparity
bm.subpixel	Choices	Sub-pixel interpolation
bm.subpixel none	*Choice*	None
bm.subpixel parabolic	*Choice*	Parabolic
bm.subpixel triangular	*Choice*	Triangular
bm.subpixel dichotomy	*Choice*	Dichotomy
bm.step	Int	Computation step
bm.startx	Int	X start index
bm.starty	Int	Y start index
bm.medianfilter	Group	Median filtering
bm.medianfilter.radius	Int	Radius
bm.medianfilter.incoherence	Float	Incoherence threshold
bm.initdisp	Choices	Initial disparities
bm.initdisp none	*Choice*	None
bm.initdisp uniform	*Choice*	Uniform initial disparity
bm.initdisp maps	*Choice*	Initial disparity maps
bm.initdisp.uniform.hdisp	Int	Horizontal initial disparity
bm.initdisp.uniform.vdisp	Int	Vertical initial disparity
bm.initdisp.uniform.hrad	Int	Horizontal exploration radius
bm.initdisp.uniform.vrad	Int	Vertical exploration radius
bm.initdisp.maps.hmap	Input image	Horizontal initial disparity map
bm.initdisp.maps.vmap	Input image	Vertical initial disparity map
bm.initdisp.maps.hrad	Int	Horizontal exploration radius
bm.initdisp.maps.vrad	Int	Vertical exploration radius
ram	Int	Available RAM (Mb)
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.48: Parameters table for Pixel-wise Block-Matching.

Input and output data This group of parameters allows to set the input and output images.

- **Left input image:** The left input image (reference)
- **Right input image:** The right input (secondary)
- **The output disparity map:** An image containing the estimated disparities as well as the metric values if the option is used
- **The output mask corresponding to all criterions:** A mask image corresponding to all citerions (see masking parameters). Only required if variance threshold or nodata criterions are set.
- **Output optimal metric values as well:** If used, the output image will have a second component with metric optimal values

Image masking parameters This group of parameters allows to determine the masking parameters to prevent disparities estimation for some pixels of the left image

- **Discard left pixels from mask image:** This parameter allows to provide a custom mask for the left image.Block matching will be only perform on pixels inside the mask.
- **Discard right pixels from mask image:** This parameter allows to provide a custom mask for the right image.Block matching will be perform only on pixels inside the mask.
- **Discard pixels with no-data value:** This parameter allows to discard pixels whose value is equal to the user-defined no-data value.
- **Discard pixels with low local variance:** This parameter allows to discard pixels whose local variance is too small (the size of the neighborhood is given by the radius parameter)

Block matching parameters This group of parameters allow to tune the block-matching behaviour

- **Block-matching metric:**

  Available choices are:

  - **Sum of Squared Distances**: Sum of squared distances between pixels value in the metric window
  - **Normalized Cross-Correlation**: Normalized Cross-Correlation between the left and right windows
  - **Lp pseudo-norm**: Lp pseudo-norm between the left and right windows
    - **p value:** Value of the p parameter in Lp pseudo-norm (must be positive)
- **Radius of blocks:** The radius (in pixels) of blocks in Block-Matching
- **Minimum horizontal disparity:** Minimum horizontal disparity to explore (can be negative)

- **Maximum horizontal disparity:** Maximum horizontal disparity to explore (can be negative)
- **Minimum vertical disparity:** Minimum vertical disparity to explore (can be negative)
- **Maximum vertical disparity:** Maximum vertical disparity to explore (can be negative)
- **Sub-pixel interpolation:** Estimate disparities with sub-pixel precision
  Available choices are:
  - **None**: No sub-pixel
  - **Parabolic**: Parabolic fit
  - **Triangular**: Triangular fit
  - **Dichotomy**: Dichotomic search
- **Computation step:** Location step between computed disparities
- **X start index:** X start index of the subsampled grid (wrt the input image grid)
- **Y start index:** Y start index of the subsampled grid (wrt the input image grid)
- **Median filtering:** Use a median filter to get a smooth disparity map
  - **Radius:** Radius for median filter
  - **Incoherence threshold:** Incoherence threshold between original and filtered disparity
- **Initial disparities:**
  Available choices are:
  - **None**: No initial disparity used
  - **Uniform initial disparity**: Use an uniform initial disparity estimate
    - **Horizontal initial disparity:** Value of the uniform horizontal disparity initial estimate (in pixels)
    - **Vertical initial disparity:** Value of the uniform vertical disparity initial estimate (in pixels)
    - **Horizontal exploration radius:** Horizontal exploration radius around the initial disparity estimate (in pixels)
    - **Vertical exploration radius:** Vertical exploration radius around the initial disparity estimate (in pixels)
  - **Initial disparity maps**: Use initial disparity maps
    - **Horizontal initial disparity map:** Map of the initial horizontal disparities
    - **Vertical initial disparity map:** Map of the initial vertical disparities
    - **Horizontal exploration radius:** Horizontal exploration radius around the initial disparity estimate (in pixels)
    - **Vertical exploration radius:** Vertical exploration radius around the initial disparity estimate (in pixels)

Available RAM (Mb) Available memory for processing (in MB)

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_BlockMatching -io.inleft StereoFixed.png -io.inright StereoMoving.png -bm.minhd -10
     -bm.maxhd 10 -mask.variancet 10 -io.out MyDisparity.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the BlockMatching application
BlockMatching = otbApplication.Registry.CreateApplication("BlockMatching")

# The following lines set all the application parameters:
BlockMatching.SetParameterString("io.inleft", "StereoFixed.png")

BlockMatching.SetParameterString("io.inright", "StereoMoving.png")

BlockMatching.SetParameterInt("bm.minhd", -10)

BlockMatching.SetParameterInt("bm.maxhd", 10)

BlockMatching.SetParameterFloat("mask.variancet", 10)

BlockMatching.SetParameterString("io.out", "MyDisparity.tif")

# The following line execute the application
BlockMatching.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- otbStereoRectificationGridGenerator

5.7.2 Disparity map to elevation map

Projects a disparity map into a regular elevation map

Detailed description

This application uses a disparity map computed from a stereo image pair to produce an elevation map on the ground area covered by the stereo pair. The needed inputs are : the disparity map, the stereo pair (in original geometry) and the epipolar deformation grids. These grids have to link the original geometry (stereo pair) and the epipolar geometry (disparity map).

Parameters

This section describes in details the parameters available for this application. Table 5.49, page 223 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `DisparityMapToElevationMap`.

Parameter key	Parameter type	Parameter description
`io`	Group	Input and output data
`io.in`	Input image	Input disparity map
`io.left`	Input image	Left sensor image
`io.right`	Input image	Right sensor image
`io.lgrid`	Input image	Left Grid
`io.rgrid`	Input image	Right Grid
`io.out`	Output image	Output elevation map
`io.mask`	Input image	Disparity mask
`step`	Float	DEM step
`hmin`	Float	Minimum elevation expected
`hmax`	Float	Maximum elevation expected
`elev`	Group	Elevation management
`elev.dem`	Directory	DEM directory
`elev.geoid`	Input File name	Geoid File
`elev.default`	Float	Default elevation
`ram`	Int	Available RAM (Mb)
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file
.../...		

Parameter key	Parameter type	Parameter description
.../...		

Figure 5.49: Parameters table for Disparity map to elevation map.

Input and output data This group of parameters allows to set the input and output images and grids.

- **Input disparity map:** The input disparity map (horizontal disparity in first band, vertical in second)
- **Left sensor image:** Left image in original (sensor) geometry
- **Right sensor image:** Right image in original (sensor) geometry
- **Left Grid:** Left epipolar grid (deformation grid between sensor et disparity spaces)
- **Right Grid:** Right epipolar grid (deformation grid between sensor et disparity spaces)
- **Output elevation map:** Output elevation map in ground projection
- **Disparity mask:** Masked disparity cells won't be projected

DEM step Spacing of the output elevation map (in meters)

Minimum elevation expected Minimum elevation expected (in meters)

Maximum elevation expected Maximum elevation expected (in meters)

Elevation management This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).

- **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

Available RAM (Mb) Available memory for processing (in MB)

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_DisparityMapToElevationMap -io.in disparity.tif -io.left sensor_left.tif -io.right
     sensor_right.tif -io.lgrid grid_epi_left.tif -io.rgrid grid_epi_right.tif -io.out dem.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the DisparityMapToElevationMap application
DisparityMapToElevationMap =
     otbApplication.Registry.CreateApplication("DisparityMapToElevationMap")

# The following lines set all the application parameters:
DisparityMapToElevationMap.SetParameterString("io.in", "disparity.tif")

DisparityMapToElevationMap.SetParameterString("io.left", "sensor_left.tif")

DisparityMapToElevationMap.SetParameterString("io.right", "sensor_right.tif")

DisparityMapToElevationMap.SetParameterString("io.lgrid", "grid_epi_left.tif")

DisparityMapToElevationMap.SetParameterString("io.rgrid", "grid_epi_right.tif")

DisparityMapToElevationMap.SetParameterString("io.out", "dem.tif")

# The following line execute the application
DisparityMapToElevationMap.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- otbStereoRectificationGridGenerator otbBlockMatching

5.7.3 Fine Registration

Estimate disparity map between two images.

Detailed description

Estimate disparity map between two images. Output image contain x offset, y offset and metric value.

Parameters

This section describes in details the parameters available for this application. Table 5.50, page 226 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `FineRegistration`.

Parameter key	Parameter type	Parameter description
`ref`	Input image	Reference Image
`sec`	Input image	Secondary Image
`out`	Output image	Output Image
`erx`	Int	Exploration Radius X
`ery`	Int	Exploration Radius Y
`mrx`	Int	Metric Radius X
`mry`	Int	Metric Radius Y
`w`	Input image	Image To Warp
`wo`	Output image	Output Warped Image
`cox`	Float	Coarse Offset X
`coy`	Float	Coarse Offset Y
`ssrx`	Float	Sub-Sampling Rate X
`ssry`	Float	Sub-Sampling Rate Y
`rgsx`	Float	Reference Gaussian Smoothing X
.../...		

Parameter key	Parameter type	Parameter description
.../...		
rgsy	Float	Reference Gaussian Smoothing Y
sgsx	Float	Secondary Gaussian Smoothing X
sgsy	Float	Secondary Gaussian Smoothing Y
m	String	Metric
spa	Float	SubPixelAccuracy
vmlt	Float	Validity Mask Lower Threshold
vmut	Float	Validity Mask Upper Threshold
ram	Int	Available RAM (Mb)
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.50: Parameters table for Fine Registration.

- **Reference Image:** The reference image.
- **Secondary Image:** The secondary image.
- **Output Image:** The output image.
- **Exploration Radius X:** The exploration radius along x (in pixels)
- **Exploration Radius Y:** The exploration radius along y (in pixels)
- **Metric Radius X:** Radius along x (in pixels) of the metric computation window
- **Metric Radius Y:** Radius along y (in pixels) of the metric computation window
- **Image To Warp:** The image to warp after disparity estimation is completed
- **Output Warped Image:** The output warped image
- **Coarse Offset X:** Coarse offset along x (in physical space) between the two images
- **Coarse Offset Y:** Coarse offset along y (in physical space) between the two images
- **Sub-Sampling Rate X:** Generates a result at a coarser resolution with a given sub-sampling rate along X
- **Sub-Sampling Rate Y:** Generates a result at a coarser resolution with a given sub-sampling rate along Y
- **Reference Gaussian Smoothing X:** Performs a gaussian smoothing of the reference image. Parameter is gaussian sigma (in pixels) in X direction.

- **Reference Gaussian Smoothing Y:** Performs a gaussian smoothing of the reference image. Parameter is gaussian sigma (in pixels) in Y direction.
- **Secondary Gaussian Smoothing X:** Performs a gaussian smoothing of the secondary image. Parameter is gaussian sigma (in pixels) in X direction.
- **Secondary Gaussian Smoothing Y:** Performs a gaussian smoothing of the secondary image. Parameter is gaussian sigma (in pixels) in Y direction.
- **Metric:** Choose the metric used for block matching. Available metrics are cross-correlation (CC), cross-correlation with subtracted mean (CCSM), mean-square difference (MSD), mean reciprocal square difference (MRSD) and mutual information (MI). Default is cross-correlation
- **SubPixelAccuracy:** Metric extrema location will be refined up to the given accuracy. Default is 0.01
- **Validity Mask Lower Threshold:** Lower threshold to obtain a validity mask.
- **Validity Mask Upper Threshold:** Upper threshold to obtain a validity mask.
- **Available RAM (Mb):** Available memory for processing (in MB)
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_FineRegistration -ref StereoFixed.png -sec StereoMoving.png -out FineRegistration.tif
     -erx 2 -ery 2 -mrx 3 -mry 3
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the FineRegistration application
FineRegistration = otbApplication.Registry.CreateApplication("FineRegistration")

# The following lines set all the application parameters:
FineRegistration.SetParameterString("ref", "StereoFixed.png")

FineRegistration.SetParameterString("sec", "StereoMoving.png")

FineRegistration.SetParameterString("out", "FineRegistration.tif")

FineRegistration.SetParameterInt("erx", 2)
```

```
FineRegistration.SetParameterInt("ery", 2)

FineRegistration.SetParameterInt("mrx", 3)

FineRegistration.SetParameterInt("mry", 3)

# The following line execute the application
FineRegistration.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.7.4 Stereo Framework

Compute the ground elevation based on one or multiple stereo pair(s)

Detailed description

Compute the ground elevation with a stereo block matching algorithm between one or mulitple stereo pair in sensor geometry. The output is projected in desired geographic or cartographic map projection (UTM by default). The pipeline is made of the following steps:
for each sensor pair :
- compute the epipolar displacement grids from the stereo pair (direct and inverse)
- resample the stereo pair into epipolar geometry using BCO interpolation
- create masks for each epipolar image : remove black borders and resample input masks
- compute horizontal disparities with a block matching algorithm
- refine disparities to sub-pixel precision with a dichotomy algorithm
- apply an optional median filter
- filter disparities based on the correlation score and exploration bounds
- translate disparities in sensor geometry
convert disparity to 3D Map.
Then fuse all 3D maps to produce DSM.

Parameters

This section describes in details the parameters available for this application. Table 5.51, page 230 presents a summary of these parameters and the parameters keys to be used in command-line and

programming languages. Application key is `StereoFramework`.

Parameter key	Parameter type	Parameter description
input	Group	Input parameters
input.il	Input image list	Input images list
input.co	String	Couples list
input.channel	Int	Image channel used for the block matching
elev	Group	Elevation management
elev.dem	Directory	DEM directory
elev.geoid	Input File name	Geoid File
elev.default	Float	Default elevation
output	Group	Output parameters
output.res	Float	Output resolution
output.nodata	Float	NoData value
output.fusionmethod	Choices	Method to fuse measures in each DSM cell
output.fusionmethod max	*Choice*	The cell is filled with the maximum measured elevation values
output.fusionmethod min	*Choice*	The cell is filled with the minimum measured elevation values
output.fusionmethod mean	*Choice*	The cell is filled with the mean of measured elevation values
output.fusionmethod acc	*Choice*	accumulator mode. The cell is filled with the the number of values (for debugging purposes).
output.out	Output image	Output DSM
output.mode	Choices	Parameters estimation modes
output.mode fit	*Choice*	Fit to sensor image
output.mode user	*Choice*	User Defined
output.mode.user.ulx	Float	Upper Left X
output.mode.user.uly	Float	Upper Left Y
output.mode.user.sizex	Int	Size X
output.mode.user.sizey	Int	Size Y
output.mode.user.spacingx	Float	Pixel Size X
output.mode.user.spacingy	Float	Pixel Size Y
map	Choices	Output Cartographic Map Projection
map utm	*Choice*	Universal Trans-Mercator (UTM)
map lambert2	*Choice*	Lambert II Etendu
map lambert93	*Choice*	Lambert93
map wgs	*Choice*	WGS 84
map epsg	*Choice*	EPSG Code
map.utm.zone	Int	Zone number
map.utm.northhem	Boolean	Northern Hemisphere
map.epsg.code	Int	EPSG Code
stereorect	Group	Stereorectification Grid parameters
stereorect.fwdgridstep	Int	Step of the displacement grid (in pixels)

.../...

Parameter key	Parameter type	Parameter description
.../...		
stereorect.invgridssrate	Int	Sub-sampling rate for epipolar grid inversion
bm	Group	Block matching parameters
bm.metric	Choices	Block-matching metric
bm.metric ssdmean	*Choice*	Sum of Squared Distances divided by mean of block
bm.metric ssd	*Choice*	Sum of Squared Distances
bm.metric ncc	*Choice*	Normalized Cross-Correlation
bm.metric lp	*Choice*	Lp pseudo-norm
bm.metric.lp.p	Float	p value
bm.radius	Int	Radius of blocks for matching filter (in pixels)
bm.minhoffset	Float	Minimum altitude offset (in meters)
bm.maxhoffset	Float	Maximum altitude offset (in meters)
postproc	Group	Postprocessing parameters
postproc.bij	Boolean	Use bijection consistency in block matching strategy
postproc.med	Boolean	Use median disparities filtering
postproc.metrict	Float	Correlation metric threshold
mask	Group	Masks
mask.left	Input image	Input left mask
mask.right	Input image	Input right mask
mask.variancet	Float	Discard pixels with low local variance
ram	Int	Available RAM (Mb)
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.51: Parameters table for Stereo Framework.

Input parameters This group of parameters allows to parametrize input data.

- **Input images list:** The list of images.
- **Couples list:** List of index of couples im image list. Couples must be separated by a comma. (index start at 0). for example : 0 1,1 2 will process a first couple composed of the first and the second image in image list, then the first and the third image
  . note that images are handled by pairs. if left empty couples are created from input index i.e. a first couple will be composed of the first and second image, a second couple with third and fourth image etc. (in this case image list must be even).

- **Image channel used for the block matching:** Used channel for block matching (used for all images)

Elevation management This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).
- **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

Output parameters This group of parameters allows to choose the DSM resolution, nodata value, and projection parameters.

- **Output resolution:** Spatial sampling distance of the output elevation : the cell size (in m)
- **NoData value:** DSM empty cells are filled with this value (optional -32768 by default)
- **Method to fuse measures in each DSM cell:** This parameter allows to choose the method used to fuse elevation measurements in each output DSM cell

  Available choices are:

  - **The cell is filled with the maximum measured elevation values**
  - **The cell is filled with the minimum measured elevation values**
  - **The cell is filled with the mean of measured elevation values**
  - **accumulator mode. The cell is filled with the the number of values (for debugging purposes).**
- **Output DSM:** Output elevation image
- **Parameters estimation modes:**

  Available choices are:

  - **Fit to sensor image**: Fit the size, origin and spacing to an existing ortho image (uses the value of outputs.ortho)

  - **User Defined**: This mode allows you to fully modify default values.
    * **Upper Left X :** Cartographic X coordinate of upper-left corner (meters for cartographic projections, degrees for geographic ones)
    * **Upper Left Y :** Cartographic Y coordinate of the upper-left corner (meters for cartographic projections, degrees for geographic ones)
    * **Size X :** Size of projected image along X (in pixels)
    * **Size Y :** Size of projected image along Y (in pixels)
    * **Pixel Size X :** Size of each pixel along X axis (meters for cartographic projections, degrees for geographic ones)
    * **Pixel Size Y :** Size of each pixel along Y axis (meters for cartographic projections, degrees for geographic ones)

Output Cartographic Map Projection Parameters of the output map projection to be used. Available choices are:

- **Universal Trans-Mercator (UTM)**: A system of transverse mercator projections dividing the surface of Earth between 80S and 84N latitude.
  - **Zone number:** The zone number ranges from 1 to 60 and allows to define the transverse mercator projection (along with the hemisphere)
  - **Northern Hemisphere:** The transverse mercator projections are defined by their zone number as well as the hemisphere. Activate this parameter if your image is in the northern hemisphere.
- **Lambert II Etendu**: This is a Lambert Conformal Conic projection mainly used in France.
- **Lambert93**: This is a Lambert 93 projection mainly used in France.
- **WGS 84**: This is a Geographical projection
- **EPSG Code**: This code is a generic way of identifying map projections, and allows to specify a large amount of them. See www.spatialreference.org to find which EPSG code is associated to your projection;
  - **EPSG Code:** See www.spatialreference.org to find which EPSG code is associated to your projection

Stereorectification Grid parameters This group of parameters allows to choose direct and inverse grid subsampling. These parameters are very useful to tune time and memory consumption.

- **Step of the displacement grid (in pixels):** Stereo-rectification displacement grid only varies slowly. Therefore, it is recommended to use a coarser grid (higher step value) in case of large images

- **Sub-sampling rate for epipolar grid inversion:** Grid inversion is an heavy process that implies spline regression on control points. To avoid eating to much memory, this parameter allows to first sub-sample the field to invert.

Block matching parameters This group of parameters allow to tune the block-matching behavior

- **Block-matching metric:**

  Available choices are:

  - **Sum of Squared Distances divided by mean of block**: derived version of Sum of Squared Distances between pixels value in the metric window (SSD divided by mean over window)
  - **Sum of Squared Distances**: Sum of squared distances between pixels value in the metric window
  - **Normalized Cross-Correlation**: Normalized Cross-Correlation between the left and right windows
  - **Lp pseudo-norm**: Lp pseudo-norm between the left and right windows
    - **p value:** Value of the p parameter in Lp pseudo-norm (must be positive)

- **Radius of blocks for matching filter (in pixels):** The radius of blocks in Block-Matching (in pixels)
- **Minimum altitude offset (in meters):** Minimum altitude below the selected elevation source (in meters)
- **Maximum altitude offset (in meters):** Maximum altitude above the selected elevation source (in meters)

Postprocessing parameters This group of parameters allow use optional filters.

- **Use bijection consistency in block matching strategy:** use bijection consistency. Right to Left correlation is computed to validate Left to Right disparities. If bijection is not found pixel is rejected.
- **Use median disparities filtering:** disparities output can be filtered using median post filtering (disabled by default).
- **Correlation metric threshold:** Use block matching metric output to discard pixels with low correlation value (disabled by default, float value)

Masks

- **Input left mask:** Mask for left input image
- **Input right mask:** Mask for right input image
- **Discard pixels with low local variance:** This parameter allows to discard pixels whose local variance is too small (the size of the neighborhood is given by the radius parameter)

Available RAM (Mb)

Available memory for processing (in MB)

Load otb application from xml file

Load otb application from xml file

Save otb application to xml file

Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_StereoFramework -input.il sensor_stereo_left.tif sensor_stereo_right.tif -elev.default
     200 -stereorect.fwdgridstep 8 -stereorect.invgridssrate 4 -postproc.med 1 -output.res
     2.5 -output.out dem.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the StereoFramework application
StereoFramework = otbApplication.Registry.CreateApplication("StereoFramework")

# The following lines set all the application parameters:
StereoFramework.SetParameterStringList("input.il", ['sensor_stereo_left.tif',
     'sensor_stereo_right.tif'])

StereoFramework.SetParameterFloat("elev.default", 200)

StereoFramework.SetParameterInt("stereorect.fwdgridstep", 8)

StereoFramework.SetParameterInt("stereorect.invgridssrate", 4)

StereoFramework.SetParameterString("postproc.med","1")

StereoFramework.SetParameterFloat("output.res", 2.5)

StereoFramework.SetParameterString("output.out", "dem.tif")

# The following line execute the application
StereoFramework.ExecuteAndWriteOutput()
```

Authors

This application has been written by OTB-Team.

5.7.5 Stereo-rectification deformation grid generator

Generates two deformation fields to stereo-rectify (i.e. resample in epipolar geometry) a pair of stereo images up to the sensor model precision

Detailed description

This application generates a pair of deformation grid to stereo-rectify a pair of stereo images according to sensor modelling and a mean elevation hypothesis. The deformation grids can be passed to the StereoRectificationGridGenerator application for actual resampling in epipolar geometry.

Parameters

This section describes in details the parameters available for this application. Table 5.52, page 236 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `StereoRectificationGridGenerator`.

Parameter key	Parameter type	Parameter description
`io`	Group	Input and output data
`io.inleft`	Input image	Left input image
`io.inright`	Input image	Right input image
`io.outleft`	Output image	Left output deformation grid
`io.outright`	Output image	Right output deformation grid
`epi`	Group	Epipolar geometry and grid parameters
`epi.elevation`	Group	Elevation management
`epi.elevation.dem`	Directory	DEM directory
`epi.elevation.geoid`	Input File name	Geoid File
`epi.elevation.default`	Float	Default elevation
`epi.elevation.avgdem`	Group	Average elevation computed from DEM
`epi.elevation.avgdem.step`	Int	Sub-sampling step
`epi.elevation.avgdem.value`	Float	Average elevation value
`epi.elevation.avgdem.mindisp`	Float	Minimum disparity from DEM
`epi.elevation.avgdem.maxdisp`	Float	Maximum disparity from DEM
`epi.scale`	Float	Scale of epipolar images
`epi.step`	Int	Step of the deformation grid (in nb. of pixels)
`epi.rectsizex`	Int	Rectified image size X
.../...		

Parameter key	Parameter type	Parameter description
.../...		
epi.rectsizey	Int	Rectified image size Y
epi.baseline	Float	Mean baseline ratio
inverse	Group	Write inverse fields
inverse.outleft	Output image	Left inverse deformation grid
inverse.outright	Output image	Right inverse deformation grid
inverse.ssrate	Int	Sub-sampling rate for inversion
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.52: Parameters table for Stereo-rectification deformation grid generator.

Input and output data This group of parameters allows to set the input and output images.

- **Left input image:** The left input image to resample
- **Right input image:** The right input image to resample
- **Left output deformation grid:** The output deformation grid to be used to resample the left input image
- **Right output deformation grid:** The output deformation grid to be used to resample the right input image

Epipolar geometry and grid parameters Parameters of the epipolar geometry and output grids

- **Elevation management:** This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.
  - **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles
  - **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).
  - **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

  - **Average elevation computed from DEM:** Average elevation computed from the provided DEM
    * **Sub-sampling step:** Step of sub-sampling for average elevation estimation
    * **Average elevation value:** Average elevation value estimated from DEM
    * **Minimum disparity from DEM:** Disparity corresponding to estimated minimum elevation over the left image
    * **Maximum disparity from DEM:** Disparity corresponding to estimated maximum elevation over the left image

- **Scale of epipolar images:** The scale parameter allows to generated zoomed-in (scale <1) or zoomed-out (scale >1) epipolar images.

- **Step of the deformation grid (in nb. of pixels):** Stereo-rectification deformation grid only varies slowly. Therefore, it is recommanded to use a coarser grid (higher step value) in case of large images

- **Rectified image size X:** The application computes the optimal rectified image size so that the whole left input image fits into the rectified area. However, due to the scale and step parameter, this size may not match the size of the deformation field output. In this case, one can use these output values.

- **Rectified image size Y:** The application computes the optimal rectified image size so that the whole left input image fits into the rectified area. However, due to the scale and step parameter, this size may not match the size of the deformation field output. In this case, one can use these output values.

- **Mean baseline ratio:** This parameter is the mean value, in pixels.meters^{-}1, of the baseline to sensor altitude ratio. It can be used to convert disparities to physical elevation, since a disparity of one pixel will correspond to an elevation offset of the invert of this value with respect to the mean elevation.

Write inverse fields This group of parameter allows to generate the inverse fields as well

- **Left inverse deformation grid:** The output deformation grid to be used to resample the epipolar left image

- **Right inverse deformation grid:** The output deformation grid to be used to resample the epipolar right image

- **Sub-sampling rate for inversion:** Grid inversion is an heavy process that implies spline regression on control points. To avoid eating to much memory, this parameter allows to first sub-sample the field to invert.

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_StereoRectificationGridGenerator -io.inleft wv2_xs_left.tif -io.inright
     wv2_xs_left.tif -io.outleft wv2_xs_left_epi_field.tif -io.outright
     wv2_xs_right_epi_field.tif -epi.elevation.default 400
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the StereoRectificationGridGenerator application
StereoRectificationGridGenerator =
     otbApplication.Registry.CreateApplication("StereoRectificationGridGenerator")

# The following lines set all the application parameters:
StereoRectificationGridGenerator.SetParameterString("io.inleft", "wv2_xs_left.tif")

StereoRectificationGridGenerator.SetParameterString("io.inright", "wv2_xs_left.tif")

StereoRectificationGridGenerator.SetParameterString("io.outleft", "wv2_xs_left_epi_field.tif")

StereoRectificationGridGenerator.SetParameterString("io.outright",
     "wv2_xs_right_epi_field.tif")

StereoRectificationGridGenerator.SetParameterFloat("epi.elevation.default", 400)

# The following line execute the application
StereoRectificationGridGenerator.ExecuteAndWriteOutput()
```

Limitations

Generation of the deformation grid is not streamable, pay attention to this fact when setting the grid step.

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- otbGridBasedImageResampling

5.8 Learning

5.8.1 Classification Map Regularization

Filters the input labeled image using Majority Voting in a ball shaped neighbordhood.

Detailed description

This application filters the input labeled image (with a maximal class label = 65535) using Majority Voting in a ball shaped neighbordhood. Majority Voting takes the more representative value of all the pixels identified by the ball shaped structuring element and then sets the center pixel to this majority label value.
-NoData is the label of the NOT classified pixels in the input image. These input pixels keep their NoData label in the output image.
-Pixels with more than 1 majority class are marked as Undecided if the parameter 'ip.suvbool == true', or keep their Original labels otherwise.

Parameters

This section describes in details the parameters available for this application. Table 5.53, page 239 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `ClassificationMapRegularization`.

Parameter key	Parameter type	Parameter description
`io`	Group	Input and output images
`io.in`	Input image	Input classification image
`io.out`	Output image	Output regularized image
`ip`	Group	Regularization parameters
`ip.radius`	Int	Structuring element radius (in pixels)
`ip.suvbool`	Boolean	Multiple majority: Undecided(X)/Original
`ip.nodatalabel`	Int	Label for the NoData class
`ip.undecidedlabel`	Int	Label for the Undecided class
`ram`	Int	Available RAM (Mb)
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.53: Parameters table for Classification Map Regularization.

Input and output images This group of parameters allows to set input and output images for classification map regularization by Majority Voting.

- **Input classification image:** The input labeled image to regularize.
- **Output regularized image:** The output regularized labeled image.

Regularization parameters This group allows to set parameters for classification map regularization by Majority Voting.

- **Structuring element radius (in pixels):** The radius of the ball shaped structuring element (expressed in pixels). By default, 'ip.radius = 1 pixel'.
- **Multiple majority: Undecided(X)/Original:** Pixels with more than 1 majority class are marked as Undecided if this parameter is checked (true), or keep their Original labels otherwise (false). Please note that the Undecided value must be different from existing labels in the input labeled image. By default, 'ip.suvbool = false'.
- **Label for the NoData class:** Label for the NoData class. Such input pixels keep their NoData label in the output image. By default, 'ip.nodatalabel = 0'.
- **Label for the Undecided class:** Label for the Undecided class. By default, 'ip.undecidedlabel = 0'.

Available RAM (Mb) Available memory for processing (in MB)

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_ClassificationMapRegularization -io.in clLabeledImageQB123_1.tif -io.out
     clLabeledImageQB123_1_CMR_r2_nodl_10_undl_7.tif -ip.radius 2 -ip.suvbool true
     -ip.nodatalabel 10 -ip.undecidedlabel 7
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication
```

```
# The following line creates an instance of the ClassificationMapRegularization application
ClassificationMapRegularization =
     otbApplication.Registry.CreateApplication("ClassificationMapRegularization")

# The following lines set all the application parameters:
ClassificationMapRegularization.SetParameterString("io.in", "clLabeledImageQB123_1.tif")

ClassificationMapRegularization.SetParameterString("io.out",
     "clLabeledImageQB123_1_CMR_r2_nodl_10_undl_7.tif")

ClassificationMapRegularization.SetParameterInt("ip.radius", 2)

ClassificationMapRegularization.SetParameterString("ip.suvbool","1")

ClassificationMapRegularization.SetParameterInt("ip.nodatalabel", 10)

ClassificationMapRegularization.SetParameterInt("ip.undecidedlabel", 7)

# The following line execute the application
ClassificationMapRegularization.ExecuteAndWriteOutput()
```

Limitations

The input image must be a single band labeled image (with a maximal class label = 65535). The structuring element radius must have a minimum value equal to 1 pixel. Please note that the Undecided value must be different from existing labels in the input labeled image.

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- Documentation of the ClassificationMapRegularization application.

5.8.2 Confusion matrix Computation

Computes the confusion matrix of a classification

Detailed description

This application computes the confusion matrix of a classification map relatively to a ground truth. This ground truth can be given as a raster or a vector data. Only reference and produced pixels with

values different from NoData are handled in the calculation of the confusion matrix. The confusion matrix is organized the following way: rows = reference labels, columns = produced labels. In the header of the output file, the reference and produced class labels are ordered according to the rows/columns of the confusion matrix.

Parameters

This section describes in details the parameters available for this application. Table 5.54, page 242 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `ComputeConfusionMatrix`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`out`	Output File name	Matrix output
`ref`	Choices	Ground truth
`ref raster`	*Choice*	Ground truth as a raster image
`ref vector`	*Choice*	Ground truth as a vector data file
`ref.raster.in`	Input image	Input reference image
`ref.vector.in`	Input File name	Input reference vector data
`ref.vector.field`	String	Field name
`nodatalabel`	Int	Value for nodata pixels
`ram`	Int	Available RAM (Mb)
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.54: Parameters table for Confusion matrix Computation.

Input Image The input classification image.

Matrix output Filename to store the output matrix (csv format)

Ground truth Choice of ground truth format Available choices are:

- **Ground truth as a raster image**
  - **Input reference image:** Input image containing the ground truth labels
- **Ground truth as a vector data file**

- **Input reference vector data:** Input vector data of the ground truth
- **Field name:** Field name containing the label values

Value for nodata pixels Label for the NoData class. Such input pixels will be discarded from the ground truth and from the input classification map. By default, 'nodatalabel = 0'.

Available RAM (Mb) Available memory for processing (in MB)

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_ComputeConfusionMatrix -in clLabeledImageQB1.tif -out ConfusionMatrix.csv -ref vector
     -ref.vector.in VectorData_QB1_bis.shp -ref.vector.field Class -nodatalabel 255
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the ComputeConfusionMatrix application
ComputeConfusionMatrix = otbApplication.Registry.CreateApplication("ComputeConfusionMatrix")

# The following lines set all the application parameters:
ComputeConfusionMatrix.SetParameterString("in", "clLabeledImageQB1.tif")

ComputeConfusionMatrix.SetParameterString("out", "ConfusionMatrix.csv")

ComputeConfusionMatrix.SetParameterString("ref","vector")

ComputeConfusionMatrix.SetParameterString("ref.vector.in", "VectorData_QB1_bis.shp")

ComputeConfusionMatrix.SetParameterString("ref.vector.field", "Class")

ComputeConfusionMatrix.SetParameterInt("nodatalabel", 255)

# The following line execute the application
ComputeConfusionMatrix.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.8.3 Compute Images second order statistics

Computes global mean and standard deviation for each band from a set of images and optionally saves the results in an XML file.

Detailed description

This application computes a global mean and standard deviation for each band of a set of images and optionally saves the results in an XML file. The output XML is intended to be used an input for the TrainImagesClassifier application to normalize samples before learning.

Parameters

This section describes in details the parameters available for this application. Table 5.55, page 244 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `ComputeImagesStatistics`.

Parameter key	Parameter type	Parameter description
`il`	Input image list	Input images
`bv`	Float	Background Value
`out`	Output File name	Output XML file
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.55: Parameters table for Compute Images second order statistics.

- **Input images:** List of input images filenames.
- **Background Value:** Background value to ignore in statistics computation.
- **Output XML file:** XML filename where the statistics are saved for future reuse.
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_ComputeImagesStatistics -il QB_1_ortho.tif -out EstimateImageStatisticsQB1.xml
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the ComputeImagesStatistics application
ComputeImagesStatistics = otbApplication.Registry.CreateApplication("ComputeImagesStatistics")

# The following lines set all the application parameters:
ComputeImagesStatistics.SetParameterStringList("il", ['QB_1_ortho.tif'])

ComputeImagesStatistics.SetParameterString("out", "EstimateImageStatisticsQB1.xml")

# The following line execute the application
ComputeImagesStatistics.ExecuteAndWriteOutput()
```

Limitations

Each image of the set must contain the same bands as the others (i.e. same types, in the same order).

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- Documentation of the TrainImagesClassifier application.

5.8.4 Fusion of Classifications

Fuses several classifications maps of the same image on the basis of class labels.

Detailed description

This application allows to fuse several classification maps and produces a single more robust classification map. Fusion is done either by mean of Majority Voting, or with the Dempster Shafer

combination method on class labels.
-MAJORITY VOTING: for each pixel, the class with the highest number of votes is selected.
-DEMPSTER SHAFER: for each pixel, the class label for which the Belief Function is maximal is selected. This Belief Function is calculated by mean of the Dempster Shafer combination of Masses of Belief, and indicates the belief that each input classification map presents for each label value. Moreover, the Masses of Belief are based on the input confusion matrices of each classification map, either by using the PRECISION or RECALL rates, or the OVERALL ACCURACY, or the KAPPA coefficient. Thus, each input classification map needs to be associated with its corresponding input confusion matrix file for the Dempster Shafer fusion.
-Input pixels with the NODATA label are not handled in the fusion of classification maps. Moreover, pixels for which all the input classifiers are set to NODATA keep this value in the output fused image.
-In case of number of votes equality, the UNDECIDED label is attributed to the pixel.

Parameters

This section describes in details the parameters available for this application. Table 5.56, page 246 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `FusionOfClassifications`.

Parameter key	Parameter type	Parameter description
`il`	Input image list	Input classifications
`method`	Choices	Fusion method
`method majorityvoting`	*Choice*	Majority Voting
`method dempstershafer`	*Choice*	Dempster Shafer combination
`method.dempstershafer.cmfl`	Input File name list	Confusion Matrices
`method.dempstershafer.mob`	Choices	Mass of belief measurement
`method.dempstershafer.mob precision`	*Choice*	Precision
`method.dempstershafer.mob recall`	*Choice*	Recall
`method.dempstershafer.mob accuracy`	*Choice*	Overall Accuracy
`method.dempstershafer.mob kappa`	*Choice*	Kappa
`nodatalabel`	Int	Label for the NoData class
`undecidedlabel`	Int	Label for the Undecided class
`out`	Output image	The output classification image
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.56: Parameters table for Fusion of Classifications.

Input classifications List of input classification maps to fuse. Labels in each classification image must represent the same class.

Fusion method Selection of the fusion method and its parameters. Available choices are:

- **Majority Voting**: Fusion of classification maps by majority voting for each output pixel.
- **Dempster Shafer combination**: Fusion of classification maps by the Dempster Shafer combination method for each output pixel.
  - **Confusion Matrices:** A list of confusion matrix files (*.CSV format) to define the masses of belief and the class labels. Each file should be formatted the following way: the first line, beginning with a '#' symbol, should be a list of the class labels present in the corresponding input classification image, organized in the same order as the confusion matrix rows/columns.
  - **Mass of belief measurement:** Type of confusion matrix measurement used to compute the masses of belief of each classifier.

Label for the NoData class Label for the NoData class. Such input pixels keep their NoData label in the output image and are not handled in the fusion process. By default, 'nodatalabel = 0'.

Label for the Undecided class Label for the Undecided class. Pixels with more than 1 fused class are marked as Undecided. Please note that the Undecided value must be different from existing labels in the input classifications. By default, 'undecidedlabel = 0'.

The output classification image The output classification image resulting from the fusion of the input classification images.

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_FusionOfClassifications -il classification1.tif classification2.tif
     classification3.tif -method dempstershafer -method.dempstershafer.cmfl
     classification1.csv classification2.csv classification3.csv -method.dempstershafer.mob
     precision -nodatalabel 0 -undecidedlabel 10 -out classification_fused.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the FusionOfClassifications application
FusionOfClassifications = otbApplication.Registry.CreateApplication("FusionOfClassifications")

# The following lines set all the application parameters:
FusionOfClassifications.SetParameterStringList("il", ['classification1.tif',
     'classification2.tif', 'classification3.tif'])

FusionOfClassifications.SetParameterString("method","dempstershafer")


FusionOfClassifications.SetParameterString("method.dempstershafer.mob","precision")

FusionOfClassifications.SetParameterInt("nodatalabel", 0)

FusionOfClassifications.SetParameterInt("undecidedlabel", 10)

FusionOfClassifications.SetParameterString("out", "classification_fused.tif")

# The following line execute the application
FusionOfClassifications.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- ImageClassifier application

5.8.5 Image Classification

Performs a classification of the input image according to a model file.

Detailed description

This application performs an image classification based on a model file produced by the TrainImagesClassifier application. Pixels of the output image will contain the class labels decided by the classifier (maximal class label = 65535). The input pixels can be optionally centered and reduced according to the statistics file produced by the ComputeImagesStatistics application. An optional input mask can be provided, in which case only input image pixels whose corresponding mask value is greater than 0 will be classified. The remaining of pixels will be given the label 0 in the output image.

Parameters

This section describes in details the parameters available for this application. Table 5.57, page 249 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `ImageClassifier`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`mask`	Input image	Input Mask
`model`	Input File name	Model file
`imstat`	Input File name	Statistics file
`out`	Output image	Output Image
`ram`	Int	Available RAM (Mb)
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.57: Parameters table for Image Classification.

- **Input Image:** The input image to classify.
- **Input Mask:** The mask allows to restrict classification of the input image to the area where mask pixel values are greater than 0.
- **Model file:** A model file (produced by TrainImagesClassifier application, maximal class label = 65535).
- **Statistics file:** A XML file containing mean and standard deviation to center and reduce samples before classification (produced by ComputeImagesStatistics application).
- **Output Image:** Output image containing class labels

- **Available RAM (Mb):** Available memory for processing (in MB)
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_ImageClassifier -in QB_1_ortho.tif -imstat EstimateImageStatisticsQB1.xml -model
     clsvmModelQB1.svm -out clLabeledImageQB1.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the ImageClassifier application
ImageClassifier = otbApplication.Registry.CreateApplication("ImageClassifier")

# The following lines set all the application parameters:
ImageClassifier.SetParameterString("in", "QB_1_ortho.tif")

ImageClassifier.SetParameterString("imstat", "EstimateImageStatisticsQB1.xml")

ImageClassifier.SetParameterString("model", "clsvmModelQB1.svm")

ImageClassifier.SetParameterString("out", "clLabeledImageQB1.tif")

# The following line execute the application
ImageClassifier.ExecuteAndWriteOutput()
```

Limitations

The input image must have the same type, order and number of bands than the images used to produce the statistics file and the SVM model file. If a statistics file was used during training by the TrainImagesClassifier, it is mandatory to use the same statistics file for classification. If an input mask is used, its size must match the input image size.

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- TrainImagesClassifier, ValidateImagesClassifier, ComputeImagesStatistics

5.8.6 Unsupervised KMeans image classification

Unsupervised KMeans image classification

Detailed description

Performs unsupervised KMeans image classification.

Parameters

This section describes in details the parameters available for this application. Table 5.58, page 251 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `KMeansClassification`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`out`	Output image	Output Image
`ram`	Int	Available RAM (Mb)
`vm`	Input image	Validity Mask
`ts`	Int	Training set size
`nc`	Int	Number of classes
`maxit`	Int	Maximum number of iterations
`ct`	Float	Convergence threshold
`outmeans`	Output File name	Centroid filename
`rand`	Int	set user defined seed
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.58: Parameters table for Unsupervised KMeans image classification.

- **Input Image:** Input image to classify.
- **Output Image:** Output image containing the class indexes.
- **Available RAM (Mb):** Available memory for processing (in MB)
- **Validity Mask:** Validity mask. Only non-zero pixels will be used to estimate KMeans modes.

- **Training set size:** Size of the training set (in pixels).
- **Number of classes:** Number of modes, which will be used to generate class membership.
- **Maximum number of iterations:** Maximum number of iterations for the learning step.
- **Convergence threshold:** Convergence threshold for class centroid (L2 distance, by default 0.0001).
- **Centroid filename:** Output text file containing centroid positions
- **set user defined seed:** Set specific seed. with integer value.
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_KMeansClassification -in QB_1_ortho.tif -ts 1000 -nc 5 -maxit 1000 -ct 0.0001 -out
     ClassificationFilterOutput.tif
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the KMeansClassification application
KMeansClassification = otbApplication.Registry.CreateApplication("KMeansClassification")

# The following lines set all the application parameters:
KMeansClassification.SetParameterString("in", "QB_1_ortho.tif")

KMeansClassification.SetParameterInt("ts", 1000)

KMeansClassification.SetParameterInt("nc", 5)

KMeansClassification.SetParameterInt("maxit", 1000)

KMeansClassification.SetParameterFloat("ct", 0.0001)

KMeansClassification.SetParameterString("out", "ClassificationFilterOutput.tif")

# The following line execute the application
KMeansClassification.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.8.7 SOM Classification

SOM image classification.

Detailed description

Unsupervised Self Organizing Map image classification.

Parameters

This section describes in details the parameters available for this application. Table 5.59, page 254 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `SOMClassification`.

Parameter key	Parameter type	Parameter description
`in`	Input image	InputImage
`out`	Output image	OutputImage
`vm`	Input image	ValidityMask
`tp`	Float	TrainingProbability
`ts`	Int	TrainingSetSize
`sl`	Int	StreamingLines
`som`	Output image	SOM Map
`sx`	Int	SizeX
`sy`	Int	SizeY
`nx`	Int	NeighborhoodX
`ny`	Int	NeighborhoodY
`ni`	Int	NumberIteration
`bi`	Float	BetaInit
`bf`	Float	BetaFinal
`iv`	Float	InitialValue
`ram`	Int	Available RAM (Mb)
`rand`	Int	set user defined seed
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.59: Parameters table for SOM Classification.

- **InputImage:** Input image to classify.
- **OutputImage:** Output classified image (each pixel contains the index of its corresponding vector in the SOM).
- **ValidityMask:** Validity mask (only pixels corresponding to a mask value greater than 0 will be used for learning)
- **TrainingProbability:** Probability for a sample to be selected in the training set
- **TrainingSetSize:** Maximum training set size (in pixels)
- **StreamingLines:** Number of lines in each streaming block (used during data sampling)
- **SOM Map:** Output image containing the Self-Organizing Map
- **SizeX:** X size of the SOM map
- **SizeY:** Y size of the SOM map
- **NeighborhoodX:** X size of the initial neighborhood in the SOM map
- **NeighborhoodY:** Y size of the initial neighborhood in the SOM map
- **NumberIteration:** Number of iterations for SOM learning
- **BetaInit:** Initial learning coefficient
- **BetaFinal:** Final learning coefficient
- **InitialValue:** Maximum initial neuron weight
- **Available RAM (Mb):** Available memory for processing (in MB)
- **set user defined seed:** Set specific seed. with integer value.
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_SOMClassification -in QB_1_ortho.tif -out SOMClassification.tif -tp 1.0 -ts 16384 -sl
     32 -sx 32 -sy 32 -nx 10 -ny 10 -ni 5 -bi 1.0 -bf 0.1 -iv 0
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the SOMClassification application
SOMClassification = otbApplication.Registry.CreateApplication("SOMClassification")

# The following lines set all the application parameters:
SOMClassification.SetParameterString("in", "QB_1_ortho.tif")

SOMClassification.SetParameterString("out", "SOMClassification.tif")

SOMClassification.SetParameterFloat("tp", 1.0)

SOMClassification.SetParameterInt("ts", 16384)

SOMClassification.SetParameterInt("sl", 32)

SOMClassification.SetParameterInt("sx", 32)

SOMClassification.SetParameterInt("sy", 32)

SOMClassification.SetParameterInt("nx", 10)

SOMClassification.SetParameterInt("ny", 10)

SOMClassification.SetParameterInt("ni", 5)

SOMClassification.SetParameterFloat("bi", 1.0)

SOMClassification.SetParameterFloat("bf", 0.1)

SOMClassification.SetParameterFloat("iv", 0)

# The following line execute the application
SOMClassification.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.8.8 Train a classifier from multiple images

Train a classifier from multiple pairs of images and training vector data.

Detailed description

This application performs a classifier training from multiple pairs of input images and training vector data. Samples are composed of pixel values in each band optionally centered and reduced using an XML statistics file produced by the ComputeImagesStatistics application.
The training vector data must contain polygons with a positive integer field representing the class label. The name of this field can be set using the "Class label field" parameter. Training and validation sample lists are built such that each class is equally represented in both lists. One parameter allows to control the ratio between the number of samples in training and validation sets. Two parameters allow to manage the size of the training and validation sets per class and per image.
Several classifier parameters can be set depending on the chosen classifier. In the validation process, the confusion matrix is organized the following way: rows = reference labels, columns = produced labels. In the header of the optional confusion matrix output file, the validation (reference) and predicted (produced) class labels are ordered according to the rows/columns of the confusion matrix.
This application is based on LibSVM and on OpenCV Machine Learning classifiers, and is compatible with OpenCV 2.3.1 and later.

Parameters

This section describes in details the parameters available for this application. Table 5.60, page 259 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `TrainImagesClassifier`.

Parameter key	Parameter type	Parameter description
`io`	Group	Input and output data
`io.il`	Input image list	Input Image List
`io.vd`	Input vector data list	Input Vector Data List
`io.imstat`	Input File name	Input XML image statistics file
`io.confmatout`	Output File name	Output confusion matrix
`io.out`	Output File name	Output model
`elev`	Group	Elevation management
`elev.dem`	Directory	DEM directory
`elev.geoid`	Input File name	Geoid File
`elev.default`	Float	Default elevation
`sample`	Group	Training and validation samples parameters
`sample.mt`	Int	Maximum training sample size per class
`sample.mv`	Int	Maximum validation sample size per class
`sample.edg`	Boolean	On edge pixel inclusion
.../...		

Parameter key	Parameter type	Parameter description
.../...		
sample.vtr	Float	Training and validation sample ratio
sample.vfn	String	Name of the discrimination field
classifier	Choices	Classifier to use for the training
classifier libsvm	*Choice*	LibSVM classifier
classifier svm	*Choice*	SVM classifier (OpenCV)
classifier boost	*Choice*	Boost classifier
classifier dt	*Choice*	Decision Tree classifier
classifier gbt	*Choice*	Gradient Boosted Tree classifier
classifier ann	*Choice*	Artificial Neural Network classifier
classifier bayes	*Choice*	Normal Bayes classifier
classifier rf	*Choice*	Random forests classifier
classifier knn	*Choice*	KNN classifier
classifier.libsvm.k	Choices	SVM Kernel Type
classifier.libsvm.k linear	*Choice*	Linear
classifier.libsvm.k rbf	*Choice*	Gaussian radial basis function
classifier.libsvm.k poly	*Choice*	Polynomial
classifier.libsvm.k sigmoid	*Choice*	Sigmoid
classifier.libsvm.c	Float	Cost parameter C
classifier.libsvm.opt	Boolean	Parameters optimization
classifier.svm.m	Choices	SVM Model Type
classifier.svm.m csvc	*Choice*	C support vector classification
classifier.svm.m nusvc	*Choice*	Nu support vector classification
classifier.svm.m oneclass	*Choice*	Distribution estimation (One Class SVM)
classifier.svm.k	Choices	SVM Kernel Type
classifier.svm.k linear	*Choice*	Linear
classifier.svm.k rbf	*Choice*	Gaussian radial basis function
classifier.svm.k poly	*Choice*	Polynomial
classifier.svm.k sigmoid	*Choice*	Sigmoid
classifier.svm.c	Float	Cost parameter C
classifier.svm.nu	Float	Parameter nu of a SVM optimization problem (NU_SVC / ONE_CLASS)
classifier.svm.coef0	Float	Parameter coef0 of a kernel function (POLY / SIGMOID)
classifier.svm.gamma	Float	Parameter gamma of a kernel function (POLY / RBF / SIGMOID)
classifier.svm.degree	Float	Parameter degree of a kernel function (POLY)
classifier.svm.opt	Boolean	Parameters optimization
classifier.boost.t	Choices	Boost Type
classifier.boost.t discrete	*Choice*	Discrete AdaBoost
classifier.boost.t real	*Choice*	Real AdaBoost (technique using confidence-rated predictions and working well with categorical data)
.../...		

Parameter key	Parameter type	Parameter description
.../...		
classifier.boost.t logit	*Choice*	LogitBoost (technique producing good regression fits)
classifier.boost.t gentle	*Choice*	Gentle AdaBoost (technique setting less weight on outlier data points and, for that reason, being often good with regression data)
classifier.boost.w	Int	Weak count
classifier.boost.r	Float	Weight Trim Rate
classifier.boost.m	Int	Maximum depth of the tree
classifier.dt.max	Int	Maximum depth of the tree
classifier.dt.min	Int	Minimum number of samples in each node
classifier.dt.ra	Float	Termination criteria for regression tree
classifier.dt.cat	Int	Cluster possible values of a categorical variable into K <= cat clusters to find a suboptimal split
classifier.dt.f	Int	K-fold cross-validations
classifier.dt.r	Boolean	Set Use1seRule flag to false
classifier.dt.t	Boolean	Set TruncatePrunedTree flag to false
classifier.gbt.w	Int	Number of boosting algorithm iterations
classifier.gbt.s	Float	Regularization parameter
classifier.gbt.p	Float	Portion of the whole training set used for each algorithm iteration
classifier.gbt.max	Int	Maximum depth of the tree
classifier.ann.t	Choices	Train Method Type
classifier.ann.t reg	*Choice*	RPROP algorithm
classifier.ann.t back	*Choice*	Back-propagation algorithm
classifier.ann.sizes	String list	Number of neurons in each intermediate layer
classifier.ann.f	Choices	Neuron activation function type
classifier.ann.f ident	*Choice*	Identity function
classifier.ann.f sig	*Choice*	Symmetrical Sigmoid function
classifier.ann.f gau	*Choice*	Gaussian function (Not completely supported)
classifier.ann.a	Float	Alpha parameter of the activation function
classifier.ann.b	Float	Beta parameter of the activation function
classifier.ann.bpdw	Float	Strength of the weight gradient term in the BACKPROP method
classifier.ann.bpms	Float	Strength of the momentum term (the difference between weights on the 2 previous iterations)
classifier.ann.rdw	Float	Initial value Delta_0 of update-values Delta_ij in RPROP method
classifier.ann.rdwm	Float	Update-values lower limit Delta_min in RPROP method
.../...		

Parameter key	Parameter type	Parameter description
.../...		
classifier.ann.term	Choices	Termination criteria
classifier.ann.term iter	*Choice*	Maximum number of iterations
classifier.ann.term eps	*Choice*	Epsilon
classifier.ann.term all	*Choice*	Max. iterations + Epsilon
classifier.ann.eps	Float	Epsilon value used in the Termination criteria
classifier.ann.iter	Int	Maximum number of iterations used in the Termination criteria
classifier.rf.max	Int	Maximum depth of the tree
classifier.rf.min	Int	Minimum number of samples in each node
classifier.rf.ra	Float	Termination Criteria for regression tree
classifier.rf.cat	Int	Cluster possible values of a categorical variable into K <= cat clusters to find a suboptimal split
classifier.rf.var	Int	Size of the randomly selected subset of features at each tree node
classifier.rf.nbtrees	Int	Maximum number of trees in the forest
classifier.rf.acc	Float	Sufficient accuracy (OOB error)
classifier.knn.k	Int	Number of Neighbors
rand	Int	set user defined seed
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.60: Parameters table for Train a classifier from multiple images.

Input and output data This group of parameters allows to set input and output data.

- **Input Image List:** A list of input images.
- **Input Vector Data List:** A list of vector data to select the training samples.
- **Input XML image statistics file:** Input XML file containing the mean and the standard deviation of the input images.
- **Output confusion matrix:** Output file containing the confusion matrix (.csv format).
- **Output model:** Output file containing the model estimated (.txt format).

Elevation management This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).
- **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

Training and validation samples parameters This group of parameters allows to set training and validation sample lists parameters.

- **Maximum training sample size per class:** Maximum size per class (in pixels) of the training sample list (default = 1000) (no limit = -1). If equal to -1, then the maximal size of the available training sample list per class will be equal to the surface area of the smallest class multiplied by the training sample ratio.
- **Maximum validation sample size per class:** Maximum size per class (in pixels) of the validation sample list (default = 1000) (no limit = -1). If equal to -1, then the maximal size of the available validation sample list per class will be equal to the surface area of the smallest class multiplied by the validation sample ratio.
- **On edge pixel inclusion:** Takes pixels on polygon edge into consideration when building training and validation samples.
- **Training and validation sample ratio:** Ratio between training and validation samples (0.0 = all training, 1.0 = all validation) (default = 0.5).
- **Name of the discrimination field:** Name of the field used to discriminate class labels in the input vector data files.

Classifier to use for the training Choice of the classifier to use for the training. Available choices are:

- **LibSVM classifier**: This group of parameters allows to set SVM classifier parameters.
  - **SVM Kernel Type:** SVM Kernel Type.

  - **Cost parameter C:** SVM models have a cost parameter C (1 by default) to control the trade-off between training errors and forcing rigid margins.
  - **Parameters optimization:** SVM parameters optimization flag.

- **SVM classifier (OpenCV)**: This group of parameters allows to set SVM classifier parameters. See complete documentation here http://docs.opencv.org/modules/ml/doc/support_vector_machines.html.

  - **SVM Model Type:** Type of SVM formulation.
  - **SVM Kernel Type:** SVM Kernel Type.
  - **Cost parameter C:** SVM models have a cost parameter C (1 by default) to control the trade-off between training errors and forcing rigid margins.
  - **Parameter nu of a SVM optimization problem (NU_SVC / ONE_CLASS):** Parameter nu of a SVM optimization problem.
  - **Parameter coef0 of a kernel function (POLY / SIGMOID):** Parameter coef0 of a kernel function (POLY / SIGMOID).
  - **Parameter gamma of a kernel function (POLY / RBF / SIGMOID):** Parameter gamma of a kernel function (POLY / RBF / SIGMOID).
  - **Parameter degree of a kernel function (POLY):** Parameter degree of a kernel function (POLY).
  - **Parameters optimization:** SVM parameters optimization flag.
    -If set to True, then the optimal SVM parameters will be estimated. Parameters are considered optimal by OpenCV when the cross-validation estimate of the test set error is minimal. Finally, the SVM training process is computed 10 times with these optimal parameters over subsets corresponding to 1/10th of the training samples using the k-fold cross-validation (with k = 10).
    -If set to False, the SVM classification process will be computed once with the currently set input SVM parameters over the training samples.
    -Thus, even with identical input SVM parameters and a similar random seed, the output SVM models will be different according to the method used (optimized or not) because the samples are not identically processed within OpenCV.

- **Boost classifier**: This group of parameters allows to set Boost classifier parameters. See complete documentation here http://docs.opencv.org/modules/ml/doc/boosting.html.

  - **Boost Type:** Type of Boosting algorithm.
  - **Weak count:** The number of weak classifiers.
  - **Weight Trim Rate:** A threshold between 0 and 1 used to save computational time. Samples with summary weight <= (1 - weight_trim_rate) do not participate in the next iteration of training. Set this parameter to 0 to turn off this functionality.
  - **Maximum depth of the tree:** Maximum depth of the tree.

- **Decision Tree classifier**: This group of parameters allows to set Decision Tree classifier parameters. See complete documentation here http://docs.opencv.org/modules/ml/doc/decision_trees.html.
  - **Maximum depth of the tree:** The training algorithm attempts to split each node while its depth is smaller than the maximum possible depth of the tree. The actual depth may be smaller if the other termination criteria are met, and/or if the tree is pruned.
  - **Minimum number of samples in each node:** If all absolute differences between an estimated value in a node and the values of the train samples in this node are smaller than this regression accuracy parameter, then the node will not be split.
  - **Termination criteria for regression tree:**
  - **Cluster possible values of a categorical variable into K <= cat clusters to find a suboptimal split:** Cluster possible values of a categorical variable into K <= cat clusters to find a suboptimal split.
  - **K-fold cross-validations:** If cv_folds >1, then it prunes a tree with K-fold cross-validation where K is equal to cv_folds.
  - **Set Use1seRule flag to false:** If true, then a pruning will be harsher. This will make a tree more compact and more resistant to the training data noise but a bit less accurate.
  - **Set TruncatePrunedTree flag to false:** If true, then pruned branches are physically removed from the tree.
- **Gradient Boosted Tree classifier**: This group of parameters allows to set Gradient Boosted Tree classifier parameters. See complete documentation here http://docs.opencv.org/modules/ml/doc/gradient_boosted_trees.html.
  - **Number of boosting algorithm iterations:** Number "w" of boosting algorithm iterations, with w*K being the total number of trees in the GBT model, where K is the output number of classes.
  - **Regularization parameter:** Regularization parameter.
  - **Portion of the whole training set used for each algorithm iteration:** Portion of the whole training set used for each algorithm iteration. The subset is generated randomly.
  - **Maximum depth of the tree:** The training algorithm attempts to split each node while its depth is smaller than the maximum possible depth of the tree. The actual depth may be smaller if the other termination criteria are met, and/or if the tree is pruned.
- **Artificial Neural Network classifier**: This group of parameters allows to set Artificial Neural Network classifier parameters. See complete documentation here http://docs.opencv.org/modules/ml/doc/neural_networks.html.
  - **Train Method Type:** Type of training method for the multilayer perceptron (MLP) neural network.
  - **Number of neurons in each intermediate layer:** The number of neurons in each intermediate layer (excluding input and output layers).

  - **Neuron activation function type:** Neuron activation function.
  - **Alpha parameter of the activation function:** Alpha parameter of the activation function (used only with sigmoid and gaussian functions).
  - **Beta parameter of the activation function:** Beta parameter of the activation function (used only with sigmoid and gaussian functions).
  - **Strength of the weight gradient term in the BACKPROP method:** Strength of the weight gradient term in the BACKPROP method. The recommended value is about 0.1.
  - **Strength of the momentum term (the difference between weights on the 2 previous iterations):** Strength of the momentum term (the difference between weights on the 2 previous iterations). This parameter provides some inertia to smooth the random fluctuations of the weights. It can vary from 0 (the feature is disabled) to 1 and beyond. The value 0.1 or so is good enough.
  - **Initial value Delta_0 of update-values Delta_ij in RPROP method:** Initial value Delta_0 of update-values Delta_ij in RPROP method (default = 0.1).
  - **Update-values lower limit Delta_min in RPROP method:** Update-values lower limit Delta_min in RPROP method. It must be positive (default = 1e-7).
  - **Termination criteria:** Termination criteria.
  - **Epsilon value used in the Termination criteria:** Epsilon value used in the Termination criteria.
  - **Maximum number of iterations used in the Termination criteria:** Maximum number of iterations used in the Termination criteria.

- **Normal Bayes classifier**: Use a Normal Bayes Classifier. See complete documentation here http://docs.opencv.org/modules/ml/doc/normal_bayes_classifier.html.

- **Random forests classifier**: This group of parameters allows to set Random Forests classifier parameters. See complete documentation here http://docs.opencv.org/modules/ml/doc/random_trees.html.

  - **Maximum depth of the tree:** The depth of the tree. A low value will likely underfit and conversely a high value will likely overfit. The optimal value can be obtained using cross validation or other suitable methods.
  - **Minimum number of samples in each node:** If the number of samples in a node is smaller than this parameter, then the node will not be split. A reasonable value is a small percentage of the total data e.g. 1 percent.
  - **Termination Criteria for regression tree:** If all absolute differences between an estimated value in a node and the values of the train samples in this node are smaller than this regression accuracy parameter, then the node will not be split.
  - **Cluster possible values of a categorical variable into K <= cat clusters to find a suboptimal split:** Cluster possible values of a categorical variable into K <= cat clusters to find a suboptimal split.

  - **Size of the randomly selected subset of features at each tree node:** The size of the subset of features, randomly selected at each tree node, that are used to find the best split(s). If you set it to 0, then the size will be set to the square root of the total number of features.
  - **Maximum number of trees in the forest:** The maximum number of trees in the forest. Typically, the more trees you have, the better the accuracy. However, the improvement in accuracy generally diminishes and reaches an asymptote for a certain number of trees. Also to keep in mind, increasing the number of trees increases the prediction time linearly.
  - **Sufficient accuracy (OOB error):** Sufficient accuracy (OOB error).

- **KNN classifier**: This group of parameters allows to set KNN classifier parameters. See complete documentation here http://docs.opencv.org/modules/ml/doc/k_nearest_neighbors.html.
  - **Number of Neighbors:** The number of neighbors to use.

set user defined seed Set specific seed. with integer value.

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_TrainImagesClassifier -io.il QB_1_ortho.tif -io.vd VectorData_QB1.shp -io.imstat
     EstimateImageStatisticsQB1.xml -sample.mv 100 -sample.mt 100 -sample.vtr 0.5 -sample.edg
     false -sample.vfn Class -classifier libsvm -classifier.libsvm.k linear
     -classifier.libsvm.c 1 -classifier.libsvm.opt false -io.out svmModelQB1.txt
     -io.confmatout svmConfusionMatrixQB1.csv
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the TrainImagesClassifier application
TrainImagesClassifier = otbApplication.Registry.CreateApplication("TrainImagesClassifier")

# The following lines set all the application parameters:
TrainImagesClassifier.SetParameterStringList("io.il", ['QB_1_ortho.tif'])

TrainImagesClassifier.SetParameterStringList("io.vd", ['VectorData_QB1.shp'])
```

```
TrainImagesClassifier.SetParameterString("io.imstat", "EstimateImageStatisticsQB1.xml")

TrainImagesClassifier.SetParameterInt("sample.mv", 100)

TrainImagesClassifier.SetParameterInt("sample.mt", 100)

TrainImagesClassifier.SetParameterFloat("sample.vtr", 0.5)

TrainImagesClassifier.SetParameterString("sample.edg","1")

TrainImagesClassifier.SetParameterString("sample.vfn", "Class")

TrainImagesClassifier.SetParameterString("classifier","libsvm")

TrainImagesClassifier.SetParameterString("classifier.libsvm.k","linear")

TrainImagesClassifier.SetParameterFloat("classifier.libsvm.c", 1)

TrainImagesClassifier.SetParameterString("classifier.libsvm.opt","1")

TrainImagesClassifier.SetParameterString("io.out", "svmModelQB1.txt")

TrainImagesClassifier.SetParameterString("io.confmatout", "svmConfusionMatrixQB1.csv")

# The following line execute the application
TrainImagesClassifier.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- OpenCV documentation for machine learning http://docs.opencv.org/modules/ml/doc/ml.html

5.9 Segmentation

5.9.1 Connected Component Segmentation

Connected component segmentation and object based image filtering of the input image according to user-defined criterions.

Detailed description

This application allows to perform a masking, connected components segmentation and object based image filtering. First and optionally, a mask can be built based on user-defined criterions to select pixels of the image which will be segmented. Then a connected component segmentation is performed with a user defined criterion to decide whether two neighbouring pixels belong to the same segment or not. After this segmentation step, an object based image filtering is applied using another user-defined criterion reasoning on segment properties, like shape or radiometric attributes. Criterions are mathematical expressions analysed by the MuParser library (http://muparser.sourceforge.net/). For instance, expression "((b1>80) and intensity>95)" will merge two neighbouring pixel in a single segment if their intensity is more than 95 and their value in the first image band is more than 80. See parameters documentation for a list of available attributes. The output of the object based image filtering is vectorized and can be written in shapefile or KML format. If the input image is in raw geometry, resulting polygons will be transformed to WGS84 using sensor modelling before writing, to ensure consistency with GIS softwares. For this purpose, a Digital Elevation Model can be provided to the application. The whole processing is done on a per-tile basis for large images, so this application can handle images of arbitrary size.

Parameters

This section describes in details the parameters available for this application. Table 5.61, page 266 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `ConnectedComponentSegmentation`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`out`	Output vector data	Output Shape
`mask`	String	Mask expression
`expr`	String	Connected Component Expression
`minsize`	Int	Minimum Object Size
`obia`	String	OBIA Expression
`elev`	Group	Elevation management
`elev.dem`	Directory	DEM directory
`elev.geoid`	Input File name	Geoid File
`elev.default`	Float	Default elevation
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.61: Parameters table for Connected Component Segmentation.

Input Image The image to segment.

Output Shape The segmentation shape.

Mask expression Mask mathematical expression (only if support image is given)

Connected Component Expression Formula used for connected component segmentation

Minimum Object Size Min object size (area in pixel)

OBIA Expression OBIA mathematical expression

Elevation management This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).
- **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_ConnectedComponentSegmentation -in ROI_QB_MUL_4.tif -mask "((b1>80)*intensity>95)"
     -expr "distance<10" -minsize 15 -obia "SHAPE_Elongation>8" -out
     ConnectedComponentSegmentation.shp
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the ConnectedComponentSegmentation application
ConnectedComponentSegmentation =
     otbApplication.Registry.CreateApplication("ConnectedComponentSegmentation")

# The following lines set all the application parameters:
ConnectedComponentSegmentation.SetParameterString("in", "ROI_QB_MUL_4.tif")

ConnectedComponentSegmentation.SetParameterString("mask", "((b1>80)*intensity>95)")

ConnectedComponentSegmentation.SetParameterString("expr", "distance<10")

ConnectedComponentSegmentation.SetParameterInt("minsize", 15)

ConnectedComponentSegmentation.SetParameterString("obia", "SHAPE_Elongation>8")

ConnectedComponentSegmentation.SetParameterString("out", "ConnectedComponentSegmentation.shp")

# The following line execute the application
ConnectedComponentSegmentation.ExecuteAndWriteOutput()
```

Limitations

Due to the tiling scheme in case of large images, some segments can be arbitrarily split across multiple tiles.

Authors

This application has been written by OTB-Team.

5.9.2 Hoover compare segmentation

Compare two segmentations with Hoover metrics

Detailed description

This application compares a machine segmentation (MS) with a partial ground truth segmentation (GT). The Hoover metrics are used to estimate scores for correct detection, over-segmentation,

under-segmentation and missed detection.
The application can output the overall Hoover scores along with coloredimages of the MS and GT segmentation showing the state of each region (correct detection, over-segmentation, under-segmentation, missed)
The Hoover metrics are described in : Hoover et al., "An experimental comparison of range image segmentation algorithms", IEEE PAMI vol. 18, no. 7, July 1996.

Parameters

This section describes in details the parameters available for this application. Table 5.62, page 269 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `HooverCompareSegmentation`.

Parameter key	Parameter type	Parameter description
`ingt`	Input image	Input ground truth
`inms`	Input image	Input machine segmentation
`bg`	Int	Background label
`th`	Float	Overlapping threshold
`outgt`	Output image	Colored ground truth output
`outms`	Output image	Colored machine segmentation output
`rc`	Float	Correct detection score
`rf`	Float	Over-segmentation score
`ra`	Float	Under-segmentation score
`rm`	Float	Missed detection score
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.62: Parameters table for Hoover compare segmentation.

- **Input ground truth:** A partial ground truth segmentation image.
- **Input machine segmentation:** A machine segmentation image.
- **Background label:** Label value of the background in the input segmentations
- **Overlapping threshold:** Overlapping threshold used to find Hoover instances.
- **Colored ground truth output:** The colored ground truth output image.
- **Colored machine segmentation output:** The colored machine segmentation output image.

- **Correct detection score:** Overall score for correct detection (RC)
- **Over-segmentation score:** Overall score for over segmentation (RF)
- **Under-segmentation score:** Overall score for under segmentation (RA)
- **Missed detection score:** Overall score for missed detection (RM)
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_HooverCompareSegmentation -ingt maur_GT.tif -inms maur_labelled.tif -outgt
     maur_colored_GT.tif uint8
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the HooverCompareSegmentation application
HooverCompareSegmentation =
     otbApplication.Registry.CreateApplication("HooverCompareSegmentation")

# The following lines set all the application parameters:
HooverCompareSegmentation.SetParameterString("ingt", "maur_GT.tif")

HooverCompareSegmentation.SetParameterString("inms", "maur_labelled.tif")

HooverCompareSegmentation.SetParameterString("outgt", "maur_colored_GT.tif")
HooverCompareSegmentation.SetParameterOutputImagePixelType("outgt", 1)

# The following line execute the application
HooverCompareSegmentation.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- otbHooverMatrixFilter, otbHooverInstanceFilter, otbLabelMapToAttributeImageFilter

5.9.3 Exact Large-Scale Mean-Shift segmentation, step 2

Second step of the exact Large-Scale Mean-Shift segmentation workflow.

Detailed description

This application performs the second step of the exact Large-Scale Mean-Shift segmentation workflow (LSMS). Filtered range image and spatial image should be created with the MeanShiftSmoothing application, with modesearch parameter disabled. If spatial image is not set, the application will only process the range image and spatial radius parameter will not be taken into account. This application will produce a labeled image where neighbor pixels whose range distance is below range radius (and optionally spatial distance below spatial radius) will be grouped together into the same cluster. For large images one can use the nbtilesx and nbtilesy parameters for tile-wise processing, with the guarantees of identical results. Please note that this application will generate a lot of temporary files (as many as the number of tiles), and will therefore require twice the size of the final result in term of disk space. The cleanup option (activated by default) allows to remove all temporary file as soon as they are not needed anymore (if cleanup is activated, tmpdir set and tmpdir does not exists before running the application, it will be removed as well during cleanup). The tmpdir option allows to define a directory where to write the temporary files. Please also note that the output image type should be set to uint32 to ensure that there are enough labels available.

Parameters

This section describes in details the parameters available for this application. Table 5.63, page 272 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `LSMSSegmentation`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Filtered image
`inpos`	Input image	Spatial image
`out`	Output image	Output Image
`ranger`	Float	Range radius
`spatialr`	Float	Spatial radius
`minsize`	Int	Minimum Region Size
.../...		

Parameter key	Parameter type	Parameter description
.../...		
tilesizex	Int	Size of tiles in pixel (X-axis)
tilesizey	Int	Size of tiles in pixel (Y-axis)
tmpdir	Directory	Directory where to write temporary files
cleanup	Boolean	Temporary files cleaning
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.63: Parameters table for Exact Large-Scale Mean-Shift segmentation, step 2.

- **Filtered image:** The filtered image (cf. Adaptive MeanShift Smoothing application).
- **Spatial image:** The spatial image. Spatial input is the displacement map (output of the Adaptive MeanShift Smoothing application).
- **Output Image:** The output image. The output image is the segmentation of the filtered image. It is recommended to set the pixel type to uint32.
- **Range radius:** Range radius defining the radius (expressed in radiometry unit) in the multi-spectral space.
- **Spatial radius:** Spatial radius of the neighborhood.
- **Minimum Region Size:** Minimum Region Size. If, after the segmentation, a region is of size lower than this criterion, the region is deleted.
- **Size of tiles in pixel (X-axis):** Size of tiles along the X-axis.
- **Size of tiles in pixel (Y-axis):** Size of tiles along the Y-axis.
- **Directory where to write temporary files:** This applications need to write temporary files for each tile. This parameter allows choosing the path where to write those files. If disabled, the current path will be used.
- **Temporary files cleaning:** If activated, the application will try to clean all temporary files it created
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_LSMSSegmentation -in smooth.tif -inpos position.tif -out segmentation.tif -ranger 15
     -spatialr 5 -minsize 0 -tilesizex 256 -tilesizey 256
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the LSMSSegmentation application
LSMSSegmentation = otbApplication.Registry.CreateApplication("LSMSSegmentation")

# The following lines set all the application parameters:
LSMSSegmentation.SetParameterString("in", "smooth.tif")

LSMSSegmentation.SetParameterString("inpos", "position.tif")

LSMSSegmentation.SetParameterString("out", "segmentation.tif")

LSMSSegmentation.SetParameterFloat("ranger", 15)

LSMSSegmentation.SetParameterFloat("spatialr", 5)

LSMSSegmentation.SetParameterInt("minsize", 0)

LSMSSegmentation.SetParameterInt("tilesizex", 256)

LSMSSegmentation.SetParameterInt("tilesizey", 256)

# The following line execute the application
LSMSSegmentation.ExecuteAndWriteOutput()
```

Limitations

This application is part of the Large-Scale Mean-Shift segmentation workflow (LSMS) and may not be suited for any other purpose.

Authors

This application has been written by David Youssefi.

See also

These additional ressources can be useful for further information:

- MeanShiftSmoothing, LSMSSmallRegionsMerging, LSMSVectorization

5.9.4 Exact Large-Scale Mean-Shift segmentation, step 3 (optional)

Third (optional) step of the exact Large-Scale Mean-Shift segmentation workflow.

Detailed description

This application performs the third step of the exact Large-Scale Mean-Shift segmentation workflow (LSMS). Given a segmentation result (label image) and the original image, it will merge regions whose size in pixels is lower than minsize parameter with the adjacent regions with the adjacent region with closest radiometry and acceptable size. Small regions will be processed by size: first all regions of area, which is equal to 1 pixel will be merged with adjacent region, then all regions of area equal to 2 pixels, until regions of area minsize. For large images one can use the nbtilesx and nbtilesy parameters for tile-wise processing, with the guarantees of identical results.

Parameters

This section describes in details the parameters available for this application. Table 5.64, page 274 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `LSMSSmallRegionsMerging`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input image
`inseg`	Input image	Segmented image
`out`	Output image	Output Image
`minsize`	Int	Minimum Region Size
`tilesizex`	Int	Size of tiles in pixel (X-axis)
`tilesizey`	Int	Size of tiles in pixel (Y-axis)
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.64: Parameters table for Exact Large-Scale Mean-Shift segmentation, step 3 (optional).

- **Input image:** The input image.
- **Segmented image:** The segmented image input. Segmented image input is the segmentation of the input image.
- **Output Image:** The output image. The output image is the input image where the minimal regions have been merged.

- **Minimum Region Size:** Minimum Region Size. If, after the segmentation, a region is of size lower than this criterion, the region is merged with the ”nearest” region (radiometrically).
- **Size of tiles in pixel (X-axis):** Size of tiles along the X-axis.
- **Size of tiles in pixel (Y-axis):** Size of tiles along the Y-axis.
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_LSMSSmallRegionsMerging -in smooth.tif -inseg segmentation.tif -out merged.tif
     -minsize 20 -tilesizex 256 -tilesizey 256
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the LSMSSmallRegionsMerging application
LSMSSmallRegionsMerging = otbApplication.Registry.CreateApplication("LSMSSmallRegionsMerging")

# The following lines set all the application parameters:
LSMSSmallRegionsMerging.SetParameterString("in", "smooth.tif")

LSMSSmallRegionsMerging.SetParameterString("inseg", "segmentation.tif")

LSMSSmallRegionsMerging.SetParameterString("out", "merged.tif")

LSMSSmallRegionsMerging.SetParameterInt("minsize", 20)

LSMSSmallRegionsMerging.SetParameterInt("tilesizex", 256)

LSMSSmallRegionsMerging.SetParameterInt("tilesizey", 256)

# The following line execute the application
LSMSSmallRegionsMerging.ExecuteAndWriteOutput()
```

Limitations

This application is part of the Large-Scale Mean-Shift segmentation workflow (LSMS) and may not be suited for any other purpose.

Authors

This application has been written by David Youssefi.

See also

These additional ressources can be useful for further information:

- LSMSSegmentation, LSMSVectorization, MeanShiftSmoothing

5.9.5 Exact Large-Scale Mean-Shift segmentation, step 4

Fourth step of the exact Large-Scale Mean-Shift segmentation workflow.

Detailed description

This application performs the fourth step of the exact Large-Scale Mean-Shift segmentation workflow (LSMS). Given a segmentation result (label image), that may have been processed for small regions merging or not, it will convert it to a GIS vector file containing one polygon per segment. Each polygon contains additional fields: mean and variance of each channels from input image (in parameter), segmentation image label, number of pixels in the polygon. For large images one can use the nbtilesx and nbtilesy parameters for tile-wise processing, with the guarantees of identical results.

Parameters

This section describes in details the parameters available for this application. Table 5.65, page 276 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `LSMSVectorization`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`inseg`	Input image	Segmented image
`out`	Output File name	Output GIS vector file
`tilesizex`	Int	Size of tiles in pixel (X-axis)
`tilesizey`	Int	Size of tiles in pixel (Y-axis)
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.65: Parameters table for Exact Large-Scale Mean-Shift segmentation, step 4.

- **Input Image:** The input image.
- **Segmented image:** The segmented image input. Segmented image input is the segmentation of the input image.
- **Output GIS vector file:** The output GIS vector file, representing the vectorized version of the segmented image where the features of the polygons are the radiometric means and variances.
- **Size of tiles in pixel (X-axis):** Size of tiles along the X-axis.
- **Size of tiles in pixel (Y-axis):** Size of tiles along the Y-axis.
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_LSMSVectorization -in avions.tif -inseg merged.tif -out vector.shp -tilesizex 256
     -tilesizey 256
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the LSMSVectorization application
LSMSVectorization = otbApplication.Registry.CreateApplication("LSMSVectorization")

# The following lines set all the application parameters:
LSMSVectorization.SetParameterString("in", "avions.tif")

LSMSVectorization.SetParameterString("inseg", "merged.tif")

LSMSVectorization.SetParameterString("out", "vector.shp")

LSMSVectorization.SetParameterInt("tilesizex", 256)

LSMSVectorization.SetParameterInt("tilesizey", 256)

# The following line execute the application
LSMSVectorization.ExecuteAndWriteOutput()
```

Limitations

This application is part of the Large-Scale Mean-Shift segmentation workflow (LSMS) and may not be suited for any other purpose.

Authors

This application has been written by David Youssefi.

See also

These additional ressources can be useful for further information:

- MeanShiftSmoothing, LSMSSegmentation, LSMSSmallRegionsMerging

5.9.6 Segmentation

Performs segmentation of an image, and output either a raster or a vector file. In vector mode, large input datasets are supported.

Detailed description

This application allows to perform various segmentation algorithms on a multispectral image.Available segmentation algorithms are two different versions of Mean-Shift segmentation algorithm (one being multi-threaded), simple pixel based connected components according to a user-defined criterion, and watershed from the gradient of the intensity (norm of spectral bands vector). The application has two different modes that affects the nature of its output.

In raster mode, the output of the application is a classical image of unique labels identifying the segmented regions. The labeled output can be passed to the ColorMapping application to render regions with contrasted colours. Please note that this mode loads the whole input image into memory, and as such can not handle large images.

To segment large data, one can use the vector mode. In this case, the output of the application is a vector file or database. The input image is split into tiles (whose size can be set using the tilesize parameter), and each tile is loaded, segmented with the chosen algorithm, vectorized, and written into the output file or database. This piece-wise behavior ensure that memory will never get overloaded, and that images of any size can be processed. There are few more options in the vector mode. The simplify option allows to simplify the geometry (i.e. remove nodes in polygons) according to a user-defined tolerance. The stitch option allows to application to try to stitch together polygons corresponding to segmented region that may have been split by the tiling scheme.

Parameters

This section describes in details the parameters available for this application. Table 5.66, page 280 presents a summary of these parameters and the parameters keys to be used in command-line and

programming languages. Application key is `Segmentation`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`filter`	Choices	Segmentation algorithm
`filter meanshift`	*Choice*	Mean-Shift
`filter edison`	*Choice*	Edison mean-shift
`filter cc`	*Choice*	Connected components
`filter watershed`	*Choice*	Watershed
`filter mprofiles`	*Choice*	Morphological profiles based segmentation
`filter.meanshift.spatialr`	Int	Spatial radius
`filter.meanshift.ranger`	Float	Range radius
`filter.meanshift.thres`	Float	Mode convergence threshold
`filter.meanshift.maxiter`	Int	Maximum number of iterations
`filter.meanshift.minsize`	Int	Minimum region size
`filter.edison.spatialr`	Int	Spatial radius
`filter.edison.ranger`	Float	Range radius
`filter.edison.minsize`	Int	Minimum region size
`filter.edison.scale`	Float	Scale factor
`filter.cc.expr`	String	Condition
`filter.watershed.threshold`	Float	Depth Threshold
`filter.watershed.level`	Float	Flood Level
`filter.mprofiles.size`	Int	Profile Size
`filter.mprofiles.start`	Int	Initial radius
`filter.mprofiles.step`	Int	Radius step.
`filter.mprofiles.sigma`	Float	Threshold of the final decision rule
`mode`	Choices	Processing mode
`mode vector`	*Choice*	Tile-based large-scale segmentation with vector output
`mode raster`	*Choice*	Standard segmentation with labeled raster output
`mode.vector.out`	Output File name	Output vector file
`mode.vector.outmode`	Choices	Writing mode for the output vector file
`mode.vector.outmode ulco`	*Choice*	Update output vector file, only allow to create new layers
`mode.vector.outmode ovw`	*Choice*	Overwrite output vector file if existing.
`mode.vector.outmode ulovw`	*Choice*	Update output vector file, overwrite existing layer
`mode.vector.outmode ulu`	*Choice*	Update output vector file, update existing layer
`mode.vector.inmask`	Input image	Mask Image
`mode.vector.neighbor`	Boolean	8-neighbor connectivity
`mode.vector.stitch`	Boolean	Stitch polygons
`mode.vector.minsize`	Int	Minimum object size
`mode.vector.simplify`	Float	Simplify polygons
`mode.vector.layername`	String	Layer name

.../...

Parameter key	Parameter type	Parameter description
.../...		
mode.vector.fieldname	String	Geometry index field name
mode.vector.tilesize	Int	Tiles size
mode.vector.startlabel	Int	Starting geometry index
mode.vector.ogroptions	String list	OGR options for layer creation
mode.raster.out	Output image	Output labeled image
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.66: Parameters table for Segmentation.

Input Image The input image to segment

Segmentation algorithm Choice of segmentation algorithm (mean-shift by default) Available choices are:

- **Mean-Shift**: OTB implementation of the Mean-Shift algorithm (multi-threaded).
  - **Spatial radius:** Spatial radius of the neighborhood.
  - **Range radius:** Range radius defining the radius (expressed in radiometry unit) in the multispectral space.
  - **Mode convergence threshold:** Algorithm iterative scheme will stop if mean-shift vector is below this threshold or if iteration number reached maximum number of iterations.
  - **Maximum number of iterations:** Algorithm iterative scheme will stop if convergence hasn't been reached after the maximum number of iterations.
  - **Minimum region size:** Minimum size of a region (in pixel unit) in segmentation. Smaller clusters will be merged to the neighboring cluster with the closest radiometry. If set to 0 no pruning is done.
- **Edison mean-shift**: Edison implementation of mean-shift algorithm, by its authors.
  - **Spatial radius:** Spatial radius defining neighborhood.
  - **Range radius:** Range radius defining the radius (expressed in radiometry unit) in the multi-spectral space.
  - **Minimum region size:** Minimum size of a region in segmentation. Smaller clusters will be merged to the neighboring cluster with the closest radiometry.

  - **Scale factor:** Scaling of the image before processing. This is useful for images with narrow decimal ranges (like [0,1] for instance).

- **Connected components**: Simple pixel-based connected-components algorithm with a user-defined connection condition.
  - **Condition:** User defined connection condition, written as a mathematical expression. Available variables are p(i)b(i), intensity_p(i) and distance (example of expression : distance <10)

- **Watershed**: The traditional watershed algorithm. The height function is the gradient magnitude of the amplitude (square root of the sum of squared bands).
  - **Depth Threshold:** Depth threshold Units in percentage of the maximum depth in the image.
  - **Flood Level:** flood level for generating the merge tree from the initial segmentation (between 0 and 1)

- **Morphological profiles based segmentation**: Segmentation based on morphological profiles, as described in Martino Pesaresi and Jon Alti Benediktsson, Member, IEEE: A new approach for the morphological segmentation of high resolution satellite imagery. IEEE Transactions on geoscience and remote sensing, vol. 39, NO. 2, February 2001, p. 309-320.
  - **Profile Size:** Size of the profiles
  - **Initial radius:** Initial radius of the structuring element (in pixels)
  - **Radius step.:** Radius step along the profile (in pixels)
  - **Threshold of the final decision rule:** Profiles values under the threshold will be ignored.

Processing mode Choice of processing mode, either raster or large-scale. Available choices are:

- **Tile-based large-scale segmentation with vector output**: In this mode, the application will output a vector file or database, and process the input image piecewise. This allows to perform segmentation of very large images.
  - **Output vector file:** The output vector file or database (name can be anything understood by OGR)
  - **Writing mode for the output vector file:** This allows to set the writing behaviour for the output vector file. Please note that the actual behaviour depends on the file format.
  - **Mask Image:** Only pixels whose mask value is strictly positive will be segmented.
  - **8-neighbor connectivity:** Activate 8-Neighborhood connectivity (default is 4).
  - **Stitch polygons:** Scan polygons on each side of tiles and stitch polygons which connect by more than one pixel.

  - **Minimum object size:** Objects whose size is below the minimum object size (area in pixels) will be ignored during vectorization.
  - **Simplify polygons:** Simplify polygons according to a given tolerance (in pixel). This option allows to reduce the size of the output file or database.
  - **Layer name:** Name of the layer in the vector file or database (default is Layer).
  - **Geometry index field name:** Name of the field holding the geometry index in the output vector file or database.
  - **Tiles size:** User defined tiles size for tile-based segmentation. Optimal tile size is selected according to available RAM if null.
  - **Starting geometry index:** Starting value of the geometry index field
  - **OGR options for layer creation:** A list of layer creation options in the form KEY=VALUE that will be passed directly to OGR without any validity checking. Options may depend on the file format, and can be found in OGR documentation.

- **Standard segmentation with labeled raster output**: In this mode, the application will output a standard labeled raster. This mode can not handle large data.
  - **Output labeled image:** The output labeled image.

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Examples

Example 1 Example of use with vector mode and watershed segmentationTo run this example in command-line, use the following:

```
otbcli_Segmentation -in QB_Toulouse_Ortho_PAN.tif -mode vector -mode.vector.out
     SegmentationVector.sqlite -filter watershed
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the Segmentation application
Segmentation = otbApplication.Registry.CreateApplication("Segmentation")

# The following lines set all the application parameters:
Segmentation.SetParameterString("in", "QB_Toulouse_Ortho_PAN.tif")

Segmentation.SetParameterString("mode","vector")
```

```
Segmentation.SetParameterString("mode.vector.out", "SegmentationVector.sqlite")

Segmentation.SetParameterString("filter","watershed")

# The following line execute the application
Segmentation.ExecuteAndWriteOutput()
```

Example 2 Example of use with raster mode and mean-shift segmentationTo run this example in command-line, use the following:

```
otbcli_Segmentation -in QB_Toulouse_Ortho_PAN.tif -mode raster -mode.raster.out
     SegmentationRaster.tif uint16 -filter meanshift
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the Segmentation application
Segmentation = otbApplication.Registry.CreateApplication("Segmentation")

# The following lines set all the application parameters:
Segmentation.SetParameterString("in", "QB_Toulouse_Ortho_PAN.tif")

Segmentation.SetParameterString("mode","raster")

Segmentation.SetParameterString("mode.raster.out", "SegmentationRaster.tif")
Segmentation.SetParameterOutputImagePixelType("mode.raster.out", 3)

Segmentation.SetParameterString("filter","meanshift")

# The following line execute the application
Segmentation.ExecuteAndWriteOutput()
```

Limitations

In raster mode, the application can not handle large input images. Stitching step of vector mode might become slow with very large input images.
MeanShift filter results depends on the number of threads used.
Watershed and multiscale geodesic morphology segmentation will be performed on the amplitude of the input image.

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- MeanShiftSegmentation

5.10 Miscellanous

5.10.1 Band Math

Perform a mathematical operation on monoband images

Detailed description

This application performs a mathematical operation on monoband images. Mathematical formula interpretation is done via MuParser libraries http://muparser.sourceforge.net/

Parameters

This section describes in details the parameters available for this application. Table 5.67, page 284 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `BandMath`.

Parameter key	Parameter type	Parameter description
`il`	Input image list	Input image list
`out`	Output image	Output Image
`ram`	Int	Available RAM (Mb)
`exp`	String	Expression
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.67: Parameters table for Band Math.

- **Input image list:** Image list to perform computation on.
- **Output Image:** Output image.

- **Available RAM (Mb):** Available memory for processing (in MB)
- **Expression:** The mathematical expression to apply.
  Use im1b1 for the first band, im1b2 for the second one...
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_BandMath -il verySmallFSATSW_r.tif verySmallFSATSW_nir.tif verySmallFSATSW.tif -out
    apTvUtBandMathOutput.tif -exp "cos(im1b1)+im2b1*im3b1-im3b2+ndvi(im3b3, im3b4)"
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the BandMath application
BandMath = otbApplication.Registry.CreateApplication("BandMath")

# The following lines set all the application parameters:
BandMath.SetParameterStringList("il", ['verySmallFSATSW_r.tif', 'verySmallFSATSW_nir.tif',
    'verySmallFSATSW.tif'])

BandMath.SetParameterString("out", "apTvUtBandMathOutput.tif")

BandMath.SetParameterString("exp", "cos(im1b1)+im2b1*im3b1-im3b2+ndvi(im3b3, im3b4)")

# The following line execute the application
BandMath.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.10.2 Images comparaison

Estimator between 2 images.

Detailed description

This application computes MSE (Mean Squared Error), MAE (Mean Absolute Error) and PSNR (Peak Signal to Noise Ratio) between the channel of two images (reference and measurement). The user has to set the used channel and can specify a ROI.

Parameters

This section describes in details the parameters available for this application. Table 5.68, page 286 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `CompareImages`.

Parameter key	Parameter type	Parameter description
`ref`	Group	Reference image properties
`ref.in`	Input image	Reference image
`ref.channel`	Int	Reference image channel
`meas`	Group	Measured image properties
`meas.in`	Input image	Measured image
`meas.channel`	Int	Measured image channel
`roi`	Group	Region Of Interest (relative to reference image)
`roi.startx`	Int	Start X
`roi.starty`	Int	Start Y
`roi.sizex`	Int	Size X
`roi.sizey`	Int	Size Y
`mse`	Float	MSE
`mae`	Float	MAE
`psnr`	Float	PSNR
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.68: Parameters table for Images comparaison.

Reference image properties

- **Reference image:** Image used as reference in the comparison
- **Reference image channel:** Used channel for the reference image

Measured image properties

- **Measured image:** Image used as measured in the comparison
- **Measured image channel:** Used channel for the measured image

Region Of Interest (relative to reference image)

- **Start X:** ROI start x position.
- **Start Y:** ROI start y position.
- **Size X:** Size along x in pixels.
- **Size Y:** Size along y in pixels.

MSE Mean Squared Error value

MAE Mean Absolute Error value

PSNR Peak Signal to Noise Ratio value

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_CompareImages -ref.in GomaApres.png -ref.channel 1 -meas.in GomaAvant.png
     -meas.channel 2 -roi.startx 20 -roi.starty 30 -roi.sizex 150 -roi.sizey 200
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the CompareImages application
CompareImages = otbApplication.Registry.CreateApplication("CompareImages")

# The following lines set all the application parameters:
```

```
CompareImages.SetParameterString("ref.in", "GomaApres.png")

CompareImages.SetParameterInt("ref.channel", 1)

CompareImages.SetParameterString("meas.in", "GomaAvant.png")

CompareImages.SetParameterInt("meas.channel", 2)

CompareImages.SetParameterInt("roi.startx", 20)

CompareImages.SetParameterInt("roi.starty", 30)

CompareImages.SetParameterInt("roi.sizex", 150)

CompareImages.SetParameterInt("roi.sizey", 200)

# The following line execute the application
CompareImages.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- BandMath application, ImageStatistics

5.10.3 Hyperspectral data unmixing

Estimate abundance maps from an hyperspectral image and a set of endmembers.

Detailed description

The application applies a linear unmixing algorithm to an hyperspectral data cube. This method supposes that the mixture between materials in the scene is macroscopic and simulates a linear mixing model of spectra.
The Linear Mixing Model (LMM) acknowledges that reflectance spectrum associated with each pixel is a linear combination of pure materials in the recovery area, commonly known as endmembers. Endmembers can be estimated using the VertexComponentAnalysis application.

The application allows to estimate the abundance maps with several algorithms : Unconstrained Least Square (ucls), Fully Constrained Least Square (fcls), Image Space Reconstruction Algorithm (isra) and Non-negative constrained Least Square (ncls) and Minimum Dispertion Constrained Non Negative Matrix Factorization (MDMDNMF).

Parameters

This section describes in details the parameters available for this application. Table 5.69, page 289 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `HyperspectralUnmixing`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image Filename
`out`	Output image	Output Image
`ie`	Input image	Input endmembers
`ua`	Choices	Unmixing algorithm
`ua ucls`	*Choice*	UCLS
`ua ncls`	*Choice*	NCLS
`ua isra`	*Choice*	ISRA
`ua mdmdnmf`	*Choice*	MDMDNMF
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.69: Parameters table for Hyperspectral data unmixing.

- **Input Image Filename:** The hyperspectral data cube to unmix
- **Output Image:** The output abundance map
- **Input endmembers:** The endmembers (estimated pure pixels) to use for unmixing. Must be stored as a multispectral image, where each pixel is interpreted as an endmember
- **Unmixing algorithm:** The algorithm to use for unmixing Available choices are:
  - **UCLS**: Unconstrained Least Square
  - **NCLS**: Non-negative constrained Least Square
  - **ISRA**: Image Space Reconstruction Algorithm
  - **MDMDNMF**: Minimum Dispertion Constrained Non Negative Matrix Factorization

- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_HyperspectralUnmixing -in cupriteSubHsi.tif -ie cupriteEndmembers.tif -out
     HyperspectralUnmixing.tif double -ua ucls
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the HyperspectralUnmixing application
HyperspectralUnmixing = otbApplication.Registry.CreateApplication("HyperspectralUnmixing")

# The following lines set all the application parameters:
HyperspectralUnmixing.SetParameterString("in", "cupriteSubHsi.tif")

HyperspectralUnmixing.SetParameterString("ie", "cupriteEndmembers.tif")

HyperspectralUnmixing.SetParameterString("out", "HyperspectralUnmixing.tif")
HyperspectralUnmixing.SetParameterOutputImagePixelType("out", 7)

HyperspectralUnmixing.SetParameterString("ua","ucls")

# The following line execute the application
HyperspectralUnmixing.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- VertexComponentAnalysis

5.10.4 Image to KMZ Export

Export the input image in a KMZ product.

Detailed description

This application exports the input image in a kmz product that can be display in the Google Earth software. The user can set the size of the product size, a logo and a legend to the product. Furthemore, to obtain a product that fits the relief, a DEM can be used.

Parameters

This section describes in details the parameters available for this application. Table 5.70, page 291 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `KmzExport`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input image
`out`	Output File name	Output .kmz product
`tilesize`	Int	Tile Size
`logo`	Input image	Image logo
`legend`	Input image	Image legend
`elev`	Group	Elevation management
`elev.dem`	Directory	DEM directory
`elev.geoid`	Input File name	Geoid File
`elev.default`	Float	Default elevation
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.70: Parameters table for Image to KMZ Export.

Input image Input image

Output .kmz product Output Kmz product directory (with .kmz extension)

Tile Size Size of the tiles in the kmz product, in number of pixels (default = 512).

Image logo Path to the image logo to add to the KMZ product.

Image legend Path to the image legend to add to the KMZ product.

Elevation management This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).
- **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles, and no geoid file has been set. This is also used by some application as an average elevation value.

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_KmzExport -in qb_RoadExtract2.tif -out otbKmzExport.kmz -logo otb_big.png
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the KmzExport application
KmzExport = otbApplication.Registry.CreateApplication("KmzExport")

# The following lines set all the application parameters:
KmzExport.SetParameterString("in", "qb_RoadExtract2.tif")

KmzExport.SetParameterString("out", "otbKmzExport.kmz")
```

```
KmzExport.SetParameterString("logo", "otb_big.png")

# The following line execute the application
KmzExport.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- Conversion

5.10.5 Open Street Map layers importations applications

Generate a vector data from OSM on the input image extend

Detailed description

Generate a vector data from Open Street Map data. A DEM could be use. By default, the entire layer is downloaded, an image can be use as support for the OSM data. The application can provide also available classes in layers . This application required an Internet access. Informations about the OSM project : http://www.openstreetmap.fr/

Parameters

This section describes in details the parameters available for this application. Table 5.71, page 294 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `OSMDownloader`.

Parameter key	Parameter type	Parameter description
.../...		

Parameter key	Parameter type	Parameter description
.../...		
out	Output vector data	Output vector data
support	Input image	Support image
key	String	OSM tag key
value	String	OSM tag value
elev	Group	Elevation management
elev.dem	Directory	DEM directory
elev.geoid	Input File name	Geoid File
elev.default	Float	Default elevation
printclasses	Boolean	option to display available key/value classes
inxml	XML input parameters file	Load otb application from xml file
outxml	XML output parameters file	Save otb application to xml file

Figure 5.71: Parameters table for Open Street Map layers importations applications.

Output vector data Generated output vector data path

Support image Image used as support to estimate the models

OSM tag key OSM tag key to extract (highway, building...)

OSM tag value OSM tag value to extract (motorway, footway...)

Elevation management This group of parameters allows to manage elevation values. Supported formats are SRTM, DTED or any geotiff. DownloadSRTMTiles application could be a useful tool to list/download tiles related to a product.

- **DEM directory:** This parameter allows to select a directory containing Digital Elevation Model tiles
- **Geoid File:** Use a geoid grid to get the height above the ellipsoid in case there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles. A version of the geoid can be found on the OTB website (http://hg.orfeo-toolbox.org/OTB-Data/raw-file/404aa6e4b3e0/Input/DEM/egm96.grd).
- **Default elevation:** This parameter allows to set the default height above ellipsoid when there is no DEM available, no coverage for some points or pixels with no_data in the DEM tiles,

and no geoid file has been set. This is also used by some application as an average elevation value.

option to display available key/value classes Print the key/value classes available for the bounding box of the input image
** If not used : Note that the options OSMKey (-key) and Output (-out) become mandatory

Load otb application from xml file Load otb application from xml file

Save otb application to xml file Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_OSMDownloader -support qb_RoadExtract.tif -key highway -out apTvUtOSMDownloader.shp
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the OSMDownloader application
OSMDownloader = otbApplication.Registry.CreateApplication("OSMDownloader")

# The following lines set all the application parameters:
OSMDownloader.SetParameterString("support", "qb_RoadExtract.tif")

OSMDownloader.SetParameterString("key", "highway")

OSMDownloader.SetParameterString("out", "apTvUtOSMDownloader.shp")

# The following line execute the application
OSMDownloader.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

See also

These additional ressources can be useful for further information:

- Convertion

5.10.6 Obtain UTM Zone From Geo Point

UTM zone determination from a geographic point.

Detailed description

This application returns the UTM zone of an input geographic point.

Parameters

This section describes in details the parameters available for this application. Table 5.72, page 296 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `ObtainUTMZoneFromGeoPoint`.

Parameter key	Parameter type	Parameter description
`lat`	Float	Latitude
`lon`	Float	Longitude
`utm`	Int	UTMZone
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.72: Parameters table for Obtain UTM Zone From Geo Point.

- **Latitude:** Latitude value of desired point.
- **Longitude:** Longitude value of desired point.
- **UTMZone:** UTM Zone
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

Obtain a UTM Zone To run this example in command-line, use the following:

```
otbcli_ObtainUTMZoneFromGeoPoint -lat 10.0 -lon 124.0
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the ObtainUTMZoneFromGeoPoint application
ObtainUTMZoneFromGeoPoint =
     otbApplication.Registry.CreateApplication("ObtainUTMZoneFromGeoPoint")

# The following lines set all the application parameters:
ObtainUTMZoneFromGeoPoint.SetParameterFloat("lat", 10.0)

ObtainUTMZoneFromGeoPoint.SetParameterFloat("lon", 124.0)

# The following line execute the application
ObtainUTMZoneFromGeoPoint.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.10.7 Pixel Value

Get the value of a pixel.

Detailed description

Get the value of a pixel.
Pay attention, index starts at 0.

Parameters

This section describes in details the parameters available for this application. Table 5.73, page 298 presents a summary of these parameters and the parameters keys to be used in command-line and

programming languages. Application key is `PixelValue`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`coordx`	Int	Col index
`coordy`	Int	Line index
`cl`	List	Channels
`value`	String	Pixel Value
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.73: Parameters table for Pixel Value.

- **Input Image:** Input image
- **Col index:** Column index of the wanted pixel (starts at 0).
- **Line index:** Line index of the wanted pixel (starts at 0).
- **Channels:** Displayed channels
- **Pixel Value:** Pixel radiometric value
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_PixelValue -in QB_Toulouse_Ortho_XS.tif -coordx 50 -coordy 100 -cl Channel1
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the PixelValue application
PixelValue = otbApplication.Registry.CreateApplication("PixelValue")

# The following lines set all the application parameters:
PixelValue.SetParameterString("in", "QB_Toulouse_Ortho_XS.tif")
```

```
PixelValue.SetParameterInt("coordx", 50)

PixelValue.SetParameterInt("coordy", 100)

# The following line execute the application
PixelValue.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.

5.10.8 Vertex Component Analysis

Find endmembers in hyperspectral images with Vertex Component Analysis

Detailed description

Applies the Vertex Component Analysis to an hyperspectral image to extract endmembers

Parameters

This section describes in details the parameters available for this application. Table 5.74, page 300 presents a summary of these parameters and the parameters keys to be used in command-line and programming languages. Application key is `VertexComponentAnalysis`.

Parameter key	Parameter type	Parameter description
`in`	Input image	Input Image
`ne`	Int	Number of endmembers
`outendm`	Output image	Output Endmembers
`rand`	Int	set user defined seed
`inxml`	XML input parameters file	Load otb application from xml file
`outxml`	XML output parameters file	Save otb application to xml file

Figure 5.74: Parameters table for Vertex Component Analysis.

- **Input Image:** Input hyperspectral data cube
- **Number of endmembers:** The number of endmembers to extract from the data cube
- **Output Endmembers:** The endmebers, stored in a one-line multi-spectral image, each pixel representing an endmember
- **set user defined seed:** Set specific seed. with integer value.
- **Load otb application from xml file:** Load otb application from xml file
- **Save otb application to xml file:** Save otb application to xml file

Example

To run this example in command-line, use the following:

```
otbcli_VertexComponentAnalysis -in cupriteSubHsi.tif -ne 5 -outendm
     VertexComponentAnalysis.tif double
```

To run this example from Python, use the following code snippet:

```
#!/usr/bin/python

# Import the otb applications package
import otbApplication

# The following line creates an instance of the VertexComponentAnalysis application
VertexComponentAnalysis = otbApplication.Registry.CreateApplication("VertexComponentAnalysis")

# The following lines set all the application parameters:
VertexComponentAnalysis.SetParameterString("in", "cupriteSubHsi.tif")

VertexComponentAnalysis.SetParameterInt("ne", 5)

VertexComponentAnalysis.SetParameterString("outendm", "VertexComponentAnalysis.tif")
VertexComponentAnalysis.SetParameterOutputImagePixelType("outendm", 7)

# The following line execute the application
VertexComponentAnalysis.ExecuteAndWriteOutput()
```

Limitations

None

Authors

This application has been written by OTB-Team.