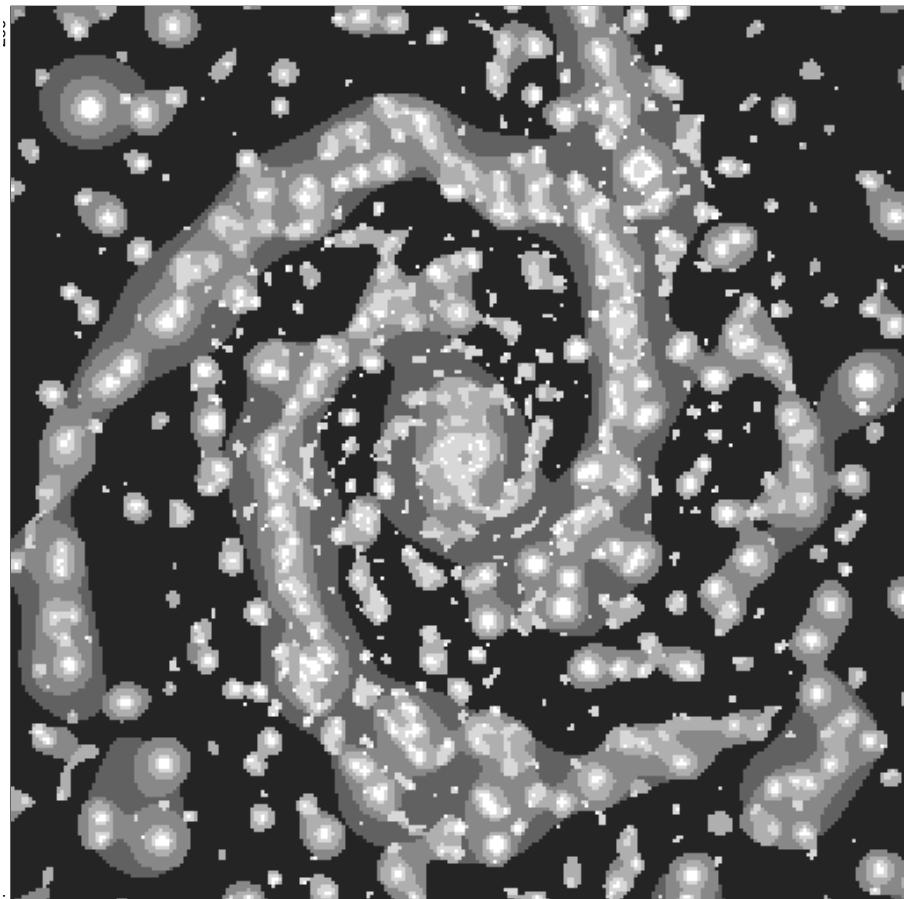


The Multiresolution Analysis Software

- **MR/1 V4.0:** Multiresolution Environment
- **MR/2 V1.2:** Multiscale Entropy
- **MR/3 V2.0:** 3D and Multichannel Data
- **MR/4 V1.0:** Ridgelet and Curvelet

CEA-SACLAY/SEDI-SAP



Contents

Contents	1
I MR/1 : The Multiresolution Environment	11
1 Introduction	13
2 MR/1 Image Processing Tools	17
2.1 Introduction	17
2.1.1 License manager: cea_lm	17
2.1.2 JAVA interface	17
2.1.3 IDL routines	18
2.2 Image Format	19
2.3 Large Image Manipulation	19
2.4 Image Manipulation Tools	21
2.4.1 Image conversion: im_convert	21
2.4.2 Image information: im_info	22
2.4.3 Image information: im_snr	23
2.4.4 Extract a subimage: im_get	24
2.4.5 Insert a subimage: im_put	24
2.4.6 Operation on two images: im_op	25
2.4.7 Create a noise map: im_random	25
2.4.8 Image simulation: im_simu	26
2.4.9 Image segmentation: im_segment	27
2.4.10 Image mirror extension: im_mirror	27
2.4.11 Image thresholding: im_threshold	27
2.4.12 Image rotation: im_rot	28
2.5 Fourier analysis	30
2.5.1 Introduction	30
2.5.2 Fourier transform (float): fft_f_2d	31
2.5.3 Fourier transform (complex): fft_cf_2d	31
2.5.4 Cosine transform: im_dct	32
2.5.5 Sinus transform: im_dst	32
2.6 Mathematical morphology	33
2.6.1 Introduction	33
2.6.2 Erosion: im_erode	34

2.6.3	Dilation: im_dilate	34
2.6.4	Opening: im_opening	35
2.6.5	Closing: im_closing	35
2.6.6	Morphological skeleton: im_thin	35
2.7	Principal component analysis	36
2.7.1	Introduction	36
2.7.2	Transform: im_pca	36
2.7.3	Transform: im_pca_rec	37
3	MR/1 Multiresolution	39
3.1	Multiscale Transform	39
3.1.1	Introduction	39
3.1.2	Methods	40
3.2	Multiresolution object	42
3.2.1	Multiresolution transform of image: mr_transform	42
3.2.2	How to select a filter bank?	47
3.2.3	Extraction of a scale: mr_extract	50
3.2.4	Insertion of an image: mr_insert	50
3.2.5	Reconstruction: mr_recons	51
3.3	Multiresolution tools	51
3.3.1	Visualization: mr_visu	51
3.3.2	Multiresolution support creation: mr_support	52
3.3.3	Statistical information: mr_info	54
3.3.4	Multiresolution segmentation: mr_segment	55
3.3.5	Noise analysis: mr_sigma	56
3.3.6	Background subtraction: mr_background	57
3.3.7	Image comparison: mr_compare	57
4	MR/1 Image Restoration	61
4.1	Introduction	61
4.1.1	Statistical significance test	61
4.1.2	Noise modeling	62
4.1.3	Filtering Methods	63
4.2	Filtering: mr_filter	67
4.3	Image with Speckle Noise	72
4.3.1	Speckle noise statistics	72
4.3.2	Speckle filtering: mr_rfilter	73
4.4	Sparse Images: à trous Algorithm	78
4.4.1	Initialization: mr_abaque	78
4.4.2	Support creation: mr_psupport	79
4.4.3	Filtering: mr_pfilter	80
4.5	Images with Poisson Noise: Haar Transform	82
4.5.1	Introduction	82
4.5.2	Poisson noise and Haar wavelet coefficients	82
4.5.3	Filtering: mr_hfilter	84
4.6	Image deconvolution	87
4.6.1	Introduction	87

4.6.2	Standard methods: im_deconv	87
4.6.3	Multiresolution methods: mr_deconv	89
4.6.4	Wavelet CLEAN: mr_mrc	91
5	MR/1 Image Compression	93
5.1	Introduction	93
5.2	Compression Methods	95
5.3	Large Image Visualization Environment: LIVE	96
5.4	Compression/Decompression Programs	99
5.4.1	Image Compression: wcomp	99
5.4.2	Noisy image Compression: mr_comp	100
5.4.3	Compression: mr_lcomp	103
5.4.4	Compression strategies	104
5.4.5	Decompression: mr_decomp	105
5.4.6	Decompression: mr_upresol	106
6	MR/1 Object Detection	109
6.1	Multiscale Vision Model	109
6.1.1	Introduction	109
6.1.2	Definition	109
6.1.3	Reconstruction	110
6.2	Detection and Deconvolution	111
6.2.1	Object reconstruction using the PSF	111
6.2.2	The algorithm	111
6.3	Object Detection: mr_detect	113
6.4	Examples and Strategies	117
6.4.1	Choice of multiscale transform	117
7	MR/1 Geometric Registration	121
7.1	Introduction	121
7.2	Deformation model	122
7.3	Image registration: mr_fusion	122
8	MR/1 Edge Detection	129
8.1	Introduction	129
8.2	First Order Derivative Edge Detection	129
8.2.1	Gradient	129
8.2.2	Gradient mask operators	130
8.2.3	Compass operators	130
8.2.4	Derivative of Gaussian	133
8.2.5	Thinning the contour	133
8.3	Second Order Derivative Edge Detection	133
8.4	Edge Detection Program: im_edge	134
8.5	Wavelets and Edge Detection	136
8.5.1	Multiscale first derivative	136
8.5.2	Image reconstruction from its multiscale edges	136
8.6	Multiscale Edge Detection Program	137

8.6.1	First derivative: mr_edge	137
8.6.2	Second derivative: mr_at_edge	138
8.6.3	Image reconstruction: mr_rec_edge	138
8.7	Contrast Enhancement	139
8.7.1	Introduction	139
8.7.2	Multiscale Edge Enhancement	139
8.7.3	The LOG-Wavelet Representation of Gray Images	140
8.7.4	Contrast Enhancement Program: mr_contrast	140
9	MR/1 1D Data	143
9.1	1D Tools	143
9.1.1	Conversion: im1d_convert	143
9.1.2	Statistical information: im1d_info	143
9.1.3	Tendency estimation: im1d_tend	145
9.2	Wavelet Transform	146
9.2.1	Multiresolution transform: mr1d_trans	146
9.2.2	Reconstruction: mr1d_recons	149
9.3	Filtering: mr1d_filter	149
9.4	Band Detection: mr1d_detect	151
9.5	Modulus Maxima Representation	153
9.5.1	Modulus maxima detection: mr1d_max	153
9.5.2	Reconstruction: mr1d_maxrecons	153
9.6	Wavelet Analysis of Time Series	155
9.6.1	Introduction	155
9.6.2	The redundant Haar wavelet transform	155
9.6.3	Autoregressive Multiscale Prediction	156
9.6.4	Wavelets and autocorrelation function: mr1d_acor	158
9.6.5	Transition detection: mr1d_nowcast	158
9.6.6	Prediction: mr1d_fcast	159
9.7	Time-Frequencies Analysysis	161
9.7.1	The Short Term Fourier Transform: im1d_stf	161
9.7.2	Time-Frequency Distribution: im1d_tfreq	163
10	MR/1 Wavelet and Multifractal Analysis	165
10.1	Fractal	165
10.1.1	Introduction	165
10.1.2	The Hausdorff and Minkowski measure	165
10.1.3	The Hausdorff and Minkowski dimension	166
10.2	Multifractality	167
10.2.1	Definition	167
10.2.2	Generalized fractal dimension	168
10.3	Wavelets and Multifractality	168
10.3.1	Singularity analysis	168
10.3.2	Wavelet transform of multifractal signals	169
10.3.3	Numerical applications of WWTM method	170
10.4	Program	173
10.4.1	Devil's Staircase: mf1d_create_ds	173

10.4.2	Chain maxima wavelet coefficients: mf1d_chain	173
10.4.3	Partition function: mf1d_repart	174
10.4.4	Multifractal analysis: mf_analyse	174
10.4.5	Examples	175
11	IDL Routines	177
11.1	Introduction	177
11.2	IDL Tools	177
11.2.1	del_pattern	177
11.2.2	block_trans	177
11.2.3	delete	178
11.2.4	im_smooth	178
11.2.5	Tikhonov	178
11.3	Noise Related Routines	179
11.3.1	poisson_image	179
11.3.2	poisson_to_gauss	179
11.3.3	sigma_clip	180
11.3.4	get_noise	180
11.4	Display	180
11.4.1	tvlut	180
11.4.2	xdisp	181
11.4.3	x3d	181
11.4.4	xdump	182
11.5	Multiresolution Routines (1D)	182
11.5.1	mr1d_atrou	182
11.5.2	mr1d_pavemed	182
11.5.3	mr1d_minmax	183
11.5.4	mr1d_pyrmmed	183
11.5.5	mr1d_paverec	183
11.5.6	mr1d_pyrrec	183
11.5.7	mr1d_pyrinterp	184
11.5.8	mr1d_tabcoef	184
11.5.9	mr1d_trans	184
11.5.10	mr1d_recons	185
11.5.11	mr1d_filter	186
11.6	Spectral Analysis	186
11.6.1	mr1d_continuum	186
11.6.2	mr1d_optical_depth	186
11.6.3	mr1d_detect	187
11.7	Multiresolution Routines (2D)	187
11.7.1	mr_transform	187
11.7.2	xmr_transform	188
11.7.3	mr_info	188
11.7.4	mr_extract	188
11.7.5	mr_read	188
11.7.6	mr_compare	189
11.7.7	mr_background	190

11.7.8 mr_filter	190
11.7.9 im_deconv	190
11.7.10 mr_deconv	190
11.7.11 mr_detect	191
11.7.12 xlive	192
Appendix A: The Lifting Scheme	193
Appendix B: Wavelet Packets	197
Appendix C: Pyramidal Wavelet Transform (PMWT)	199
Appendix D: Half Pyramidal Wavelet Transform	201
II MR/2 : Multiscale Entropy and Applications	203
12 Multiscale Entropy Theory	205
12.1 Entropy and Image Restoration	205
12.1.1 Introduction	205
12.1.2 The concept of entropy	208
12.1.3 Conclusion	211
12.2 Entropy from Noise Modeling	213
12.2.1 Information and wavelet coefficient	213
12.2.2 Signal information and noise information	214
12.2.3 Conclusion	217
13 Multiscale Entropy Applied to Filtering	219
13.1 Introduction	219
13.2 Multiresolution and Filtering	219
13.2.1 The choice of the multiresolution transform	219
13.2.2 Filtering in wavelet space	220
13.3 Multiscale Entropy Filtering	223
13.3.1 Filtering	223
13.3.2 The regularization parameter	223
13.3.3 The use of a model	224
13.3.4 The multiscale entropy filtering algorithm	226
13.3.5 Derivative calculation of h_s and h_n for N2-MSE	227
13.3.6 General Case	230
13.3.7 Optimization	230
13.4 Examples	231
13.4.1 1D data filtering	231
13.4.2 Image filtering	231
13.5 Comparison with Other Methods from Simulations	237
13.5.1 Simulation descriptions	237
13.6 Simulation Analysis	238
13.7 Conclusion: Toward Combined Filtering	239

14 Multiscale Entropy Applied to Deconvolution	245
14.1 Introduction to Deconvolution	245
14.2 Non-Regularized Deconvolution Methods	245
14.3 Tikhonov Regularization	247
14.4 The CLEAN Approach	248
14.4.1 The CLEAN algorithm	248
14.4.2 Multiresolution CLEAN	249
14.4.3 CLEAN and wavelets	250
14.5 Regularization from the Multiresolution Support	253
14.5.1 Noise suppression based on the wavelet transform	253
14.5.2 Noise suppression based on the multiresolution support	253
14.6 Regularization using a Markov Model	255
14.6.1 Introduction	255
14.6.2 Application	258
14.7 Multiscale Entropy Deconvolution	261
14.7.1 The principle	261
14.7.2 The parameters	262
15 Multiscale Entropy applied to Background Fluctuation Analysis	265
16 MR/2 Programs	269
16.1 Probability in wavelet space: mw_proba	269
16.2 Entropy of an image: mw_entrop	269
16.3 Filtering	270
16.3.1 1D filtering: mw1d_filter	270
16.3.2 2D filtering: mw_filter	271
16.3.3 2D combined filtering: mw_comb_filter	272
16.4 Deconvolution: mw_deconv	273
16.5 IDL Routines	276
16.5.1 Introduction	276
16.5.2 mw1d_filter	276
16.5.3 mw1d_predict	276
16.5.4 mw_filter	276
16.5.5 mw_deconv	277
Appendix A: The “À Trou” Wavelet Transform Algorithm	279
Appendix B: Noise Modeling the Wavelet Space	281
Appendix C: Derivative Needed for the Minimization	285
Appendix D: Generalization of Derivative Needed for Minimization	289
III MR/3 : Multichannel Data	291
17 MR/3 Data Processing Tools	293
17.1 Introduction	293

17.2	3D-Image Manipulation Tools	293
17.2.1	Image conversion: im3d_convert	293
17.2.2	Image information: im3d_info	294
17.2.3	Extract a subcube: im3d_get	294
17.2.4	Insert a subcube: im3d_put	295
17.2.5	Operation on two cubes: im3d_op	295
17.2.6	Image simulation: im3d_simu	295
17.3	Multi-Temporal Images	296
17.3.1	Introduction	296
17.3.2	Image Coaddition: im3d_coadd	296
17.3.3	Deconvolution: im3d_deconv	298
17.4	Color Images	300
17.4.1	Introduction	300
17.4.2	Color Image Filtering: col_filter	301
17.4.3	Color Image Compression: col_comp	301
17.4.4	Color Image Decompression: col_decomp	302
17.4.5	Color Image Enhancement: col_contrast	304
18	MR/3 Three Dimensional Data Set	309
18.1	MR/3 Multiresolution	309
18.1.1	Introduction	309
18.1.2	Multiresolution transform of cube: mr3d_trans	310
18.1.3	Extraction of a scale: mr3d_extract	311
18.1.4	Insertion of an image: mr3d_insert	311
18.1.5	Reconstruction: mr3d_recons	312
18.2	MR/3 Denoising: mr3d_filter	312
18.2.1	Gaussian noise filtering: mr3d_filter	312
18.3	3D Point Clouds Analysis	313
18.3.1	ASCII Catalog to 3D Cube in FITS Format: mr3d_cat2fits	313
18.3.2	Wavelet Histogram Autoconvolution: mr3d_phisto	314
18.3.3	Multiresolution Support Calculation: mr3d_psupport	315
18.3.4	Data Cube Filtering: mr3d_pfilter	316
19	MR/3 Multi-Channel Data Set	319
19.1	Introduction	319
19.2	The Wavelet-Karhunen-Loëve transform	319
19.2.1	Definition	319
19.2.2	Correlation matrix and noise modeling	321
19.2.3	Scale and Karhunen-Loëve transform	321
19.2.4	The WT-KLT transform	322
19.2.5	The WT-KLT reconstruction algorithms	322
19.2.6	WT-KLT Transform of 1D Multichannel Data: wk1d_trans	322
19.2.7	WT-KLT Reconstruction of 1D Multichannel Data: wk1d_trec	323
19.2.8	WT-KLT Transform of 2D Multichannel Data: wk_trans	324
19.2.9	WT-KLT Reconstruction of 2D Multichannel Data: wk_trec	326
19.3	Noise Modeling in the WT-KLT Space	327
19.3.1	Non-Gaussian noise	327

19.3.2 Noise level on WT-KLT coefficients	327
19.4 Multichannel Data Filtering	327
19.4.1 Introduction	327
19.4.2 Reconstruction from a subset of eigenvectors	328
19.4.3 WT-KLT Coefficient Thresholding	328
19.4.4 Multiscale Entropy	328
19.4.5 WT-KLT Filtering of 1D Multichannel Data: wk1d_filter	329
19.4.6 WT-KLT Filtering of 2D Multichannel Data: wk_filter	330
19.4.7 WT-KLT Filtering of 2D Multichannel Data by the Multiscale Entropy Method: wk_memfilter	331
19.5 Filtering using the Haar-Multichannel Transform	333
19.5.1 Definition	333
19.5.2 WT-KLT Filtering of 2D Multichannel Data by the Multiscale Haar Transform: ww_filter	333
19.6 Independent Component Analysis	335
19.6.1 JADE-ICA IDL Programs	335
IV MR/4 : Ridgelets and Curvelets	341
20 The Curvelet Transform	343
20.1 Introduction	343
20.2 Continuous Ridgelet Transform	345
20.2.1 The Radon Transform	346
20.2.2 Ridgelet Pyramids	346
20.3 Digital Ridgelet Transform	347
20.3.1 The RectoPolar Ridgelet transform	347
20.3.2 The Orthonormal Finite Ridgelet Transform	351
20.3.3 The Slant Stack Ridgelet Transform	353
20.4 Local Ridgelet Transforms	353
20.5 Digital Curvelet Transform	355
20.5.1 Discrete Curvelet Transform of Continuum Functions	355
20.5.2 Digital Realization	356
20.5.3 Algorithm	356
21 Filtering	359
21.1 Curvelet Coefficient Thresholding	359
21.2 Filtering Experiments	360
22 The Combined Filtering Method	367
22.1 Introduction	367
22.2 The Combined Filtering Principle	368
22.3 The Minimization Method	368
22.4 Experiments	369
22.5 Discussion	370

23 Contrast Enhancement	373
23.1 Introduction	373
23.2 Contrast Enhancement by the Curvelet Transform	374
23.3 Examples	376
23.4 Discussion	377
24 MR/4 Programs	381
24.1 Ridgelet Transform	381
24.1.1 im_radon	381
24.1.2 rid_trans	382
24.1.3 rid_stat	383
24.1.4 rid_filter	384
24.2 Curvelet Transform	385
24.2.1 cur_trans	385
24.2.2 cur_stat	386
24.2.3 cur_filter	386
24.2.4 cur_colfilter	387
24.2.5 cur_contrast	388
24.2.6 cur_colcontrast	389
24.3 Combined Filtering	390
24.3.1 cb_filter	390
Bibliography	392
Index	405

Part I

MR/1 : The Multiresolution Environment

Chapter 1

Introduction

MR/1 is a set of software components developed by CEA (Saclay, France) and Nice Observatory. This project originated in astronomy, and involved the development of a range of innovative methods built around multiscale analysis. Multiresolution techniques have been developed in recent years, and furnish a powerful and insightful representation of the data. By means of multiresolution or multiscale analysis, an image can be decomposed into a set of images (or scales), each scale containing only structures of a given size. This data representation, associated with noise modeling, has been applied to very different applications such as data filtering, deconvolution, compression, object detection, and so on. Results are enhanced in all such processing because the multiresolution approach allows a better understanding of how the data values are distributed in an image, and how the signal can be separated from the noise.

The MR/1 software components include almost all applications presented in the book *Image and Data Analysis: the Multiscale Approach* [191]. The goal of MR/1 is not to replace existing image processing packages, but to complement them, offering the user a complete set of multiresolution tools. These tools are executable programs, which work on a wide range of platforms, independently of current image processing systems. They allow the user to perform various tasks using multiresolution, such as wavelet transforms, filtering, deconvolution, and so on.

The programs, written in C++, are built on three classes: the “image” class, the “multiresolution” class, and the “noise-modeling class”. Fig. 1.1 illustrates this architecture. A multiresolution transform is applied to the input data, and noise modeling is performed. Hence the multiple scales can be derived, and the programs can use this in order to know at which scales, and at which positions, significant signal has been detected. A wide range of multiresolution transforms are available (see Fig. 1.2), allowing significant flexibility. Fig. 1.3 summarizes how the multiresolution support data structure is derived from the data and the noise-modeling.

A set of IDL¹ (Interactive Data Language) and PV~Wave² routines are included in the package which interface the executables to these image processing packages.

MR/1 is an important package, introducing front-line methods to scientists in the physical, space and medical domains among other fields; to engineers in such disciplines as geology and electrical engineering; and to financial engineers and those in other fields requiring control and analysis of large quantities of noisy data.

¹Research Systems Inc., 2995 Wilderness Place, Boulder, Colorado 80301.

²Visual Numerics Inc., 6230 Lookout Road, Boulder, Colorado 80301, USA.



Figure 1.1: MR/1 diagram.

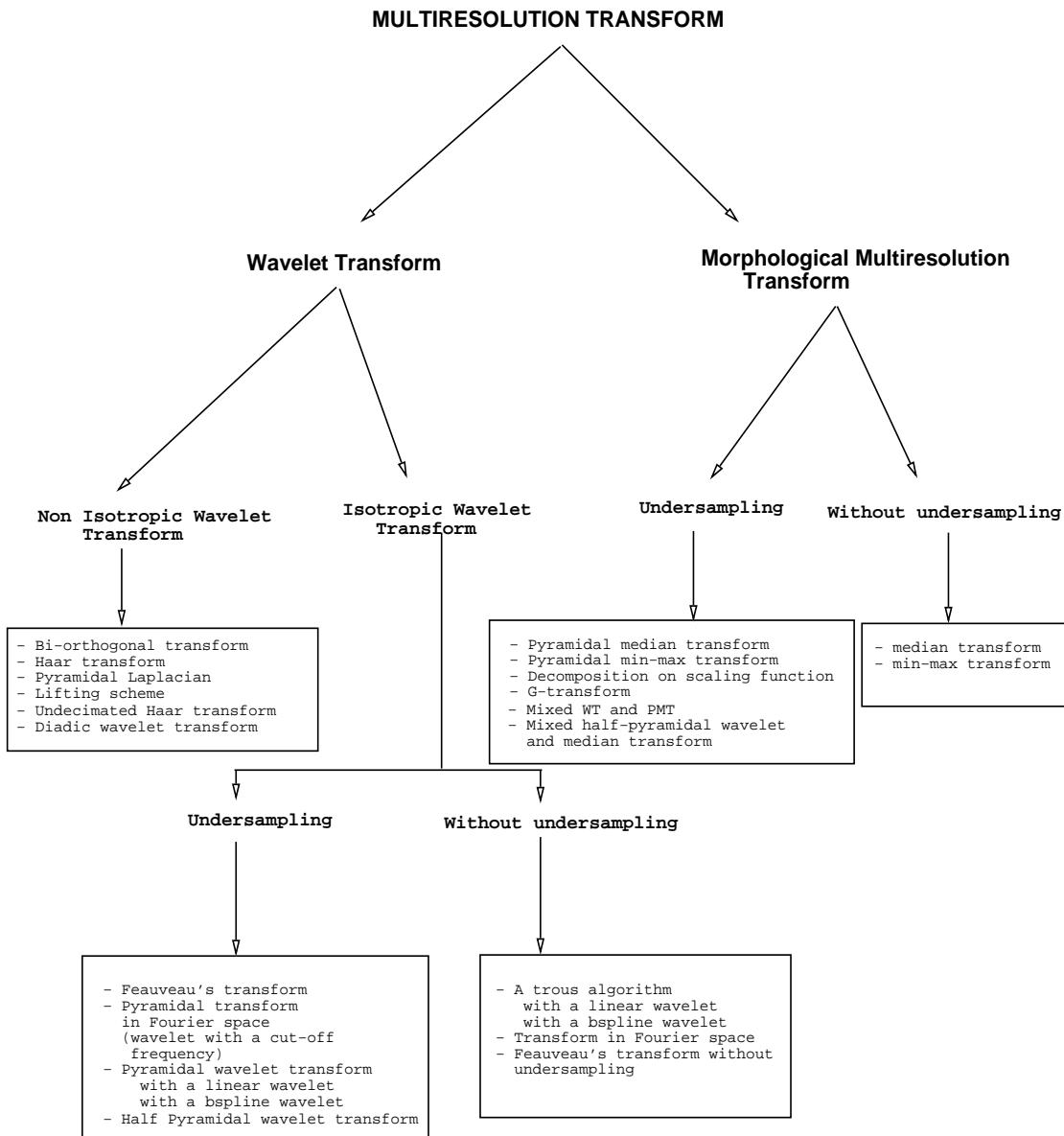


Figure 1.2: Multiresolution transforms available in MR/1 (a selection).

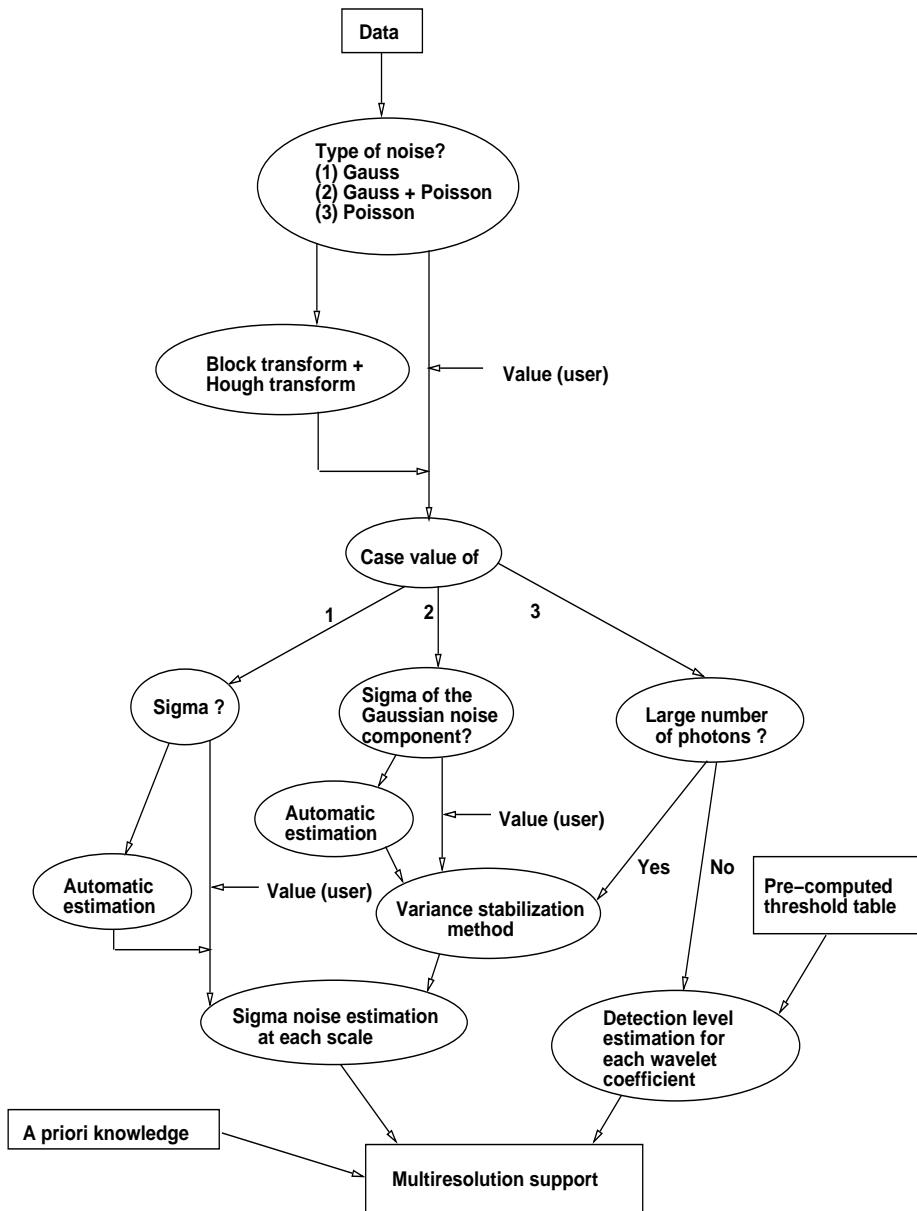


Figure 1.3: Determination of multiresolution support from noise modeling.

Chapter 2

MR/1 Image Processing Tools

2.1 Introduction

2.1.1 License manager: cea_lm

The directory with executables must be in the user path. The license manager can be run (if not already done so by your system manager) by typing

```
cea_lm -file fileName
```

or directly **cea_lm** if the default file name, **cea_lic**, is used. The license manager can be stopped before by typing

```
cea_lm -kill
```

For non-floating licenses, the programs can only be executed on the computer where the license manager is installed.

For floating licenses, the programs can be run from any computer (with the correct type of platform) on the network if the environment variable **CEA_LICENSE_HOST** is set to the correct address (for example ariane.saclay.cea.fr), or if the license file is in the directory **/usr/local/cea** (and **C:\Program Files\MR** for Windows 95/98/NT/ME, ...).

When the license manager is running, all executables described in this document can be called.

2.1.2 JAVA interface

Required packages

- JDK (<http://java.sun.com/products/index.html>), at least version 1.2.
- Swing (<http://java.sun.com/products/index.html>), at least version 1.1. With recent JDK versions, Swing is included.
- MR/1 interface Java class (mr.class,)

Running the MR Java application

On Unix computers, the **CLASSPATH** variable must contain the Java class directory and the Swing class directory.

Run the program by typing:

```
> java mr
```

On Windows, run the program by typing:

```
> jmr1
```

if the default installation directory has been chosen. Otherwise you will have to edit the jmr1.bat file, and modify the path variables according to your installation.

Exit the program by selecting “Quit...” in the “System” menu.

```
"System">"Quit..."
```

Description

The MR/1 graphical user interface allows the user to execute all the MR/1 programs. The application window contains a menu bar with 5 tear-off menus:

- System: quit, and license manipulation.
- MR/1 IMAGE: image manipulation programs (multiresolution not used).
- MR/1 Multiresol: manipulation of multiresolution objects.
- MR/1 Application: filtering, deconvolution, compression, detection, etc.
- MR/1 1D: 1D programs.
- MR/2: MR/2 programs.
- MR/3: MR/3 programs.
- MR/4: MR/4 programs.

To execute an MR program select the associated item (program name):

“MR/1 Image” > “Manipulation tools” > “im_info”, for the im_info program.

The application displays a popup window. All the options of the program can be selected. If a popup window is validated (by pressing the OK button) with an obligatory option not initialized, a warning dialog box prevents the user continuing.

When the dialog box is validated the program is executed. Information on the execution is displayed in the current window:

```
Program is running ...
... information relative to the program execution ...
Program is finished!
```

The execution is now terminated.

The help button of the dialog box displays a window with the help of the program.

2.1.3 IDL routines

To use IDL routines, the **CEA_MR_DIR** variable must be initialized to the directory where the MR/1-IDL software has been installed. Then the command

```
idl $CEA_MR_DIR/idl/pro/mre
```

runs the IDL session with the multiresolution environment. An IDL help facility on the available routines is obtained by typing **mrh** in the IDL session.

2.2 Image Format

All programs read/write to raw format (.d), FITS format (.fits), GIF format (.gif), JPEG format (.jpg) and PGM format (.pgm). A raw image file name must have the following syntax:
FileName_x_Nl_Nc.d

where x defines the data type:

```
x = b (byte)
      s (short int)
      i (int)
      f (float)
      d (double)
Nl = number of lines
Nc = number of columns
```

Example: “ngc21_f_256_256.d” is a floating image (256×256).

A recommendation to use FITS format, wherever possible, follows. Noise modeling is closely associated with high precision measurement. Therefore the image’s dynamic range is very important. FITS supports 32 bit floating values. We recommend that FITS be used for all processing in order to benefit from the use of float values in MR/1 . Images can be converted to and from FITS before and after use of MR/1 routines. FITS is a very widely-used image format in astronomy, and information on this standard is to be found at various places on the Internet (for example, see <http://heasarc.gsfc.nasa.gov/docs/software/fitsio>). Public domain image viewers are available such as xv, ds9, or SAOimage, available from the Smithsonian Astrophysical Observatory. Other public domain viewers, available on the Web, include FITSview from NRAO, and fv from GSFC.

FITS files may contain several extensions, and each extension may contain an image. MR/1 programs read only the first extension. However, the program *im_convert* allows the user to extract an image in a given extension, and to save it into a simple fits file, or any other format.

When the character “-” is used in place of an image file name, the standard input or output is used.

2.3 Large Image Manipulation

All computations are carried out by default in memory. If the computer has not enough memory to run the program, a virtual memory mode is available allowing the use of disk instead of memory. A temporary file is created each time a large buffer needs to be allocated. This obviously increases the computation time, but allows large image manipulation (assuming enough disk space is available!). Virtual Memory (VM) can be activated by two options: “-z” and “-Z”. All programs working with images accept these two options.

When a program is run with the “-z” option, the virtual memory is activated using the default parameters, which are the the minimum size buffer for activating the VM, and the directory where the temporary files are to be created. The minimum size is read from the environment variable **CEA_VM_SIZE**, and the default directory is read from the environment variable **CEA_VM_DIR**. If **CEA_VM_SIZE** and **CEA_VM_DIR** are not defined, the default minimum size is 4 MB, and the default directory is the current one, i.e. “.” (where the program is executed).

When a program is run with the “-Z” option, virtual memory is activated and the user has to specify the minimum size and the directory for the temporary files.

Examples:

- `im_info image.fits`
gives information about an image without using the VM. The image is loaded in memory.
- `im_info -z image.fits`
same, but using the VM and its default parameters.
CEA_VM_SIZE and **CEA_VM_DIR** are read, and if the image size is larger than the value read from **CEA_VM_SIZE** (or 4 MB if the environment variable is not defined), a temporary file is created.
- `im_info -Z 1:/tmp image.fits`
Same, but the VM is activated using a minimum size of only 1 MB, and the directory for storing the temporary file is “/tmp”.

2.4 Image Manipulation Tools

A number of programs have been developed for basic image manipulation.

2.4.1 Image conversion: im_convert

The program *im_convert* converts an image from one data format to another one. Currently supported image formats are the raw format, the FITS format, GIF, JPEG and PGM. In addition the 3D FITS files used by the curvelet and ridgelet transforms, with extension “.rid” and “.cur”, are also supported.

USAGE: im_convert options file_name_in file_name_out

where options are:

- **[-b]**
Normalize the data between 0 and 255.
- **[-x]**
Flip x-axis.
- **[-y]**
Flip y-axis.
- **[-r]**
90 degrees rotation.
- **[-h FitsHduNbr]**
FITS HDU number. FITS files may contain several extensions (or HDU), and each extension may contain an image. This option allows the user to select the HDU number.

If the “-b” option is set, data are scaled in order to have a minimum value equal to 0, and a maximum value equal to 255.

Examples:

- `im_convert image_f_256_256.d image.fits`
Converts an image from raw format to FITS format.
- `im_convert -b image_f_256_256.d image.gif`
Converts an image from raw format to GIF format with normalization of scale.
- `im_convert -h 2 image.fits hdu2.fits`
Extracts an image from a fits file, and save it in another fits file.
- `im_convert -r -x image.fits ima.fits`
Flips first x-axis, and then rotates the image by 90 degrees, and saves the result in the same format.

2.4.2 Image information: im_info

im_info gives information about an image:

- the number of rows and columns,
- the minimum, the maximum,
- the arithmetic mean: $\bar{x} = \frac{1}{N} \sum_k x_k$
- the standard deviation: $\sigma = \sqrt{\frac{1}{N} \sum_k (x_k - \bar{x})^2} = \sqrt{\bar{x}^2 - \bar{x}^2}$
- the flux: $\sum_k I_k$
- the energy: $\sum_k I_k^2$

When the “-e” or “-E” option is set, then the Shannon Entropy, the Frieden Entropy and the Fisher information are also calculated:

- Shannon Entropy: The Shannon Entropy [166] is defined as

$$H_s(I) = - \sum_{k=1}^{N_b} p_k \log p_k \quad (2.1)$$

where p_k values are derived from the histogram of I ($p_k = \#[I_j = k]$).

- Frieden Entropy: Frieden [76] entropy function is similar to Shannon’s one, but the p_k values are replaced by the normalized pixel values of the image

$$H_f(I) = - \sum_{k=1}^N \psi_k \ln(\psi_k) \quad (2.2)$$

where $\psi_k = \frac{I_k}{\sum_k I_k}$.

- Fisher information: The Fisher information can be expressed by [78, 86]:

$$F(I) = 4 \sum_k \nabla \left[\sqrt{\psi_k} \right]^* \nabla \left[\sqrt{\psi_k} \right] \quad (2.3)$$

In practice, we approximate the gradient operator, ∇ , with second-order finite difference equations for each cardinal direction [86].

Fisher information, and Shannon Entropy increase when the data become more noisy or more complex, while Frieden entropy decreases.

When the “-M” option is set, then the skewness and the kurtosis are calculated by:

$$\begin{aligned} S &= \frac{1}{N\sigma^3} \sum_k (x_k - \bar{x})^3 \\ &= \frac{1}{\sigma^3} (\bar{x}^3 - 3\bar{x}\bar{x}^2 + 2\bar{x}^3) \end{aligned} \quad (2.4)$$

$$\begin{aligned} K &= \frac{1}{N\sigma^4} \sum_k (x_k - \bar{x})^4 - 3 \\ &= \frac{1}{\sigma^4} (\bar{x}^4 - 4\bar{x}\bar{x}^3 + 6\bar{x}^2\bar{x}^2 - 3\bar{x}^4) - 3 \end{aligned} \quad (2.5)$$

The skewness S is zero if the data are symmetrically distributed around the mean. If a tail extends to the right (resp. left), S is positive (resp. negative). Positive K implies a higher peak and larger wings than the Gaussian distribution with the same mean and variance. Negative K means a wider peak and shorter wings.

The command line is:

USAGE: im_info file_name.in

where options are:

- **[-e]**

Calculates the Shannon Entropy, the Frieden Entropy and the Fisher information. The histogram bin size value, used by for the Shannon entropy calculation, is 1.

- **[-E HistoBinSize]**

Calculates the Shannon Entropy, the Frieden Entropy and the Fisher information. The histogram bin size value, used for the Shannon entropy calculation, is $HistoBinSize$.

- **[-M]**

Skewness and Kurtosis calculation.

Examples:

- `im_info image_f_256_256.d`

Gives information about the image.

- `im_info -e image_f_256_256.d`

Ditto, but calculates also the Shannon Entropy, the Frieden Entropy and the Fisher information.

2.4.3 Image information: im_snr

im_snr gives the signal-to-noise ratio between two images. Different *SNR* measurements can be calculated.

The command line is:

USAGE: im_snr ReferenceImage Image

where options are:

- **[-s SNR_Type]**

1. the Least Square Error: LSE (db) = $-10\log_{10}(Variance(Error))$.
2. the Normalized LSE: NLSE (db) = $-10\log_{10}(Total(Error^2)/Total(Ref))$.
3. the Peak LSE: PLSE (db) = $-10\log_{10}(Variance(Error)/Max(Ref1^2))$.
4. the 255-Peak LSE: PLSE255(db) = $-10\log_{10}(Variance(Error)/255^2)$.
5. The variance ratio: VRVE = Variance(Reference) / Variance(Error).

Default is 4.

- **[-w]**

Write in *SNR.fits* the result.

- **[-v]**

Verbose mode. Gives all SNRs calculations.

Examples:

- `im_snr image1_f_256_256.d image2_f_256_256.d`
Gives the PSNR between the two images.

2.4.4 Extract a subimage: im_get

im_get extracts a part of an image:

USAGE: im_get options image_in image_out

where options are:

- **[-x Xstart:XEnd]**
Xstart and XEnd are the first and the last column to extract.
- **[-y Ystart:YEnd]**
Ystart and YEnd are the first and the last rows to extract.

Examples:

- `im_get -x 10:20 -y 2:5 ngc2997.fits small.fits`
Extract a small area (lines 2 to 5, and columns 10 to 20) from the image `ngc2997.fits`.
- `im_get -x 10:20 -y 2:5 ngc2997.fits - | im_info -`
Same as before, but send the extracted image to the standard output and pipe it to *im_info*.
A sample result is:

```
File name = -
Nl = 4  Nc = 11
Min = 219  Max = 352
Mean = 301.341  sigma = 28.5673
Flux = 13259  Energy = 4.03139e+06
```

2.4.5 Insert a subimage: im_put

im_put inserts an image into a larger one:

USAGE: im_put options image_in image_inout

where options are:

- **[-x Xstart]**
x-coordinate of the image to insert.
- **[-y Ystart]**
y-coordinate of the image to insert.

Examples:

- `im_put -x 10 -y 2 small.fits ngc2997.fits`
Insert a small image at position (10,2) in `ngc2997.fits`.

2.4.6 Operation on two images: im_op

im_op calculates an operation on two images.

USAGE: im_put image_in1 [+ - */] image_in2 image_out

Examples:

- **im_op image1.d - image2.d image_out.d**
image_out.d is equal to image1.d - image2.d.
- **im_op image1.d / image2.d image_out.d**
image_out.d is equal to image1.d / image2.d.
- **im_op image1.d * image2.d image_out.d**
image_out.d is equal to image1.d * image2.d. The “\” character is added in order to avoid the shell interpreting the “*” character.

2.4.7 Create a noise map: im_random

im_random creates a noisy image.

USAGE: im_random options image_out

where options are:

- **[-x Nc]**
Number of columns. Default is 256.
- **[-y Nl]**
Number of lines. Default is 256.
- **[-t DistribLaw]**
 1. Gaussian.
 2. Poisson.
 3. Rayleigh.
 4. Laplace.

Type of distribution used for the noise creation. Default is Gaussian.

- **[-s DistribParam]**
Distribution parameter. Default is 1.
- **[-I InitRandomVal]**
Value used for random value generator initialization.
Default is 100.

Examples:

- **im_random -t 1 -s 10 image_out.d**
image_out.d contains Gaussian noise, with a noise standard deviation equal to 1.
- **im_random -t 2 -s 1 image_out.d**
image_out.d contains Poisson noise, with a mean value equal to 1.

2.4.8 Image simulation: im_simu

im_simu adds noise (Poisson, Gaussian, or both kinds of noise) to the input image, and/or convolves it beforehand with a point spread function (PSF) which can either be read from a file, or created by the program. If the PSF is simulated, it is a Gaussian and a parameter fixes the full-width at half-maximum (FWHM).

USAGE: `im_simu options image_in image_out`

where options are:

- **[-p]**
Add Poisson Noise. Default is not to do this.
- **[-g sigma]**
Add Gaussian noise. *sigma* is the standard deviation of the added noise. Default is not to do this.
- **[-c sigma]**
Add Poisson noise and Gaussian noise. *sigma* is standard deviation of the Gaussian noise. Default is not to do this.
- **[-r psf_image]**
Convolve the input image with a PSF. The PSF is read from a file of name *psf_image*. Default is not to do this.
- **[-f FWHM]**
Convolve the input image with a PSF. The PSF is a Gaussian which has a full-width at half-maximum equal to *FWHM*. Default is not to do this.
- **[-I InitRandomVal]**
Value used for random value generator initialization.
Default is 100.
- **[-w PsfFileName]**
Write the calculated PSF to disk.
Valid only if “-f” option is set. Default is not to do this.

The r and f options cannot be used together.

The g, p, and c options cannot be used together.

Examples:

- `im_simu -g 200 image.d image_g200.d`
Add Gaussian noise to an image.
- `im_simu -f 3. -g 200 image.d image_g200.d`
Convolve the image with a Gaussian (FWHM=3), and add Gaussian noise of standard deviation equal to 200.
- `im_simu -r PSF.d -c 20 image.d image_g200.d`
Convolve the image with another one (PSF.d), and add Poisson noise and read-out noise (Gaussian noise).

2.4.9 Image segmentation: im_segment

The program *im_segment* applies a segmentation to the input image. The image is first thresholded.

USAGE: **im_segment options Threshold image_in image_out**

where options are:

- **[-b]**
Eliminates regions at the border. Default is no elimination.

2.4.10 Image mirror extension: im_mirror

The program *im_mirror* extends an image by interpolating the image borders by mirror effect. N_x, N_y are the number of rows and columns, the output image O is defined by $O(x, y) = I(x_1, y_1)$, where I is the input image, $x_1 = M(x, N_x)$, $y_1 = M(y, N_y)$, and M is defined by

$$M(k, N) = \begin{cases} k & \text{if } k \in [0, N - 1] \\ -k & \text{if } k < 0 \\ 2(N - 1) - k & \text{if } k \geq N \end{cases}$$

The size of the output image can be fixed by the “-x” and “-y” options. If neither of these options is set, the size is chosen to the smallest power of two larger than N_x and N_y .

The command line is:

USAGE: **im_mirror options image_in image_out**

where options are:

- **[-x Number_of_Columns]**
Number of columns of the output image. By default, it takes the next power of 2.
- **[-y Number_of_Lines]**
Number of rows of the output image. By default, it takes the next power of 2.

Example:

- `im_mirror -x 256 -y 256 image_in.d image_out_256.d`
Extend an image by mirror effect in order to get a new square image of size 256.

2.4.11 Image thresholding: im_threshold

The program *im_threshold* thresholds an image.

USAGE: **im_threshold options image_in image_out**

where options are:

- **[-t ThresholdMin]**
Threshold all values lower than ThresholdMin. All pixels with a value lower than *ThresholdMin* are set to 0. Default is 0 (negative values are thresholded).

- **[-t ThresholdMax]**

Set to ThresholdMax all values larger than ThresholdMax. Default is no.

- **[-a]**

Compare the absolute value to the threshold. Default is no.

Examples:

- im_threshold image_in.d image_out.d

Threshold all negative values in the data.

- im_threshold -t 10 image_in.d image_out.d

All pixels with values smaller than 10 are set to 0.

- im_threshold -t 10 -a image_in.d image_out.d

All pixels with absolute values smaller than 10 are set to 0.

- im_threshold -T 255 image_in.d image_out.d

All pixels with values larger than 255 are set to 255.

2.4.12 Image rotation: im_rot

The program *im_mirror* rotates an image by a given angle. Translation may also be done if needed. The rotation operator is given by:

$$R(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \quad (2.6)$$

A bilinear interpolation method is used by default by the program.

It has been shown [49, 207] that the whole transformation can be decomposed into an appropriate sequence of 1D signal translations that can all be implemented via simple convolutions using the following factorization:

$$\begin{aligned} R(\theta) &= \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} = \begin{bmatrix} 1 & -\tan \frac{\theta}{2} \\ 0 & 1 \end{bmatrix} \\ &\quad X \begin{bmatrix} 1 & 0 \\ \sin \theta & 1 \end{bmatrix} X \begin{bmatrix} 1 & -\tan \frac{\theta}{2} \\ 0 & 1 \end{bmatrix} \end{aligned} \quad (2.7)$$

The command line is:

USAGE: im_rot options image_in image_out

where options are:

- **[-r RotAngle]**

Rotation angle in degrees.

- **[-x Dx]**

x-axis translation offset.

- **[-y Dy]**

y-axis translation offset.

- **[-f]**

Use the factorization in order to have a reversible transformation. (Since the Fourier transform is used, the image size must be a power of 2.)

Examples:

- `im_rot -r 30 image_in.d image_out.d`
Rotate an image by 30 degrees.
- `im_rot -x 5 -r 30 image_in.d image_out.d`
Rotate an image by 30 degrees, but first shift the image by five pixels along the x-axis.

2.5 Fourier analysis

2.5.1 Introduction

The Fourier transform of a continuous function $f(x)$ is defined by:

$$\hat{f}(\nu) = \int_{-\infty}^{+\infty} f(x) \exp\{-i2\pi\nu x\} \quad (2.8)$$

and the inverse Fourier transform is:

$$f(x) = \int_{-\infty}^{+\infty} \hat{f}(\nu) \exp\{i2\pi\nu x\} \quad (2.9)$$

The discrete Fourier transform is given by:

$$\hat{f}(u) = \frac{1}{N} \sum_{k=-\infty}^{+\infty} f(k) \exp\{-i2\pi u k\} \quad (2.10)$$

and the inverse discrete Fourier transform is:

$$\hat{f}(k) = \sum_{u=-\infty}^{+\infty} f(u) \exp\{i2\pi u k\} \quad (2.11)$$

In the case of images (two variables), this is:

$$\hat{f}(u, v) = \frac{1}{MN} \sum_{l=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} f(k, l) \exp\{-2i\pi(\frac{uk}{M} + \frac{vl}{N})\} \quad (2.12)$$

$$f(k, l) = \sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} \hat{f}(u, v) \exp\{2i\pi(\frac{uk}{M} + \frac{vl}{N})\} \quad (2.13)$$

Since $\hat{f}(u, v)$ is generally complex, it can be written using its real and imaginary parts:

$$\hat{f}(u, v) = Re[\hat{f}(u, v)] + iIm[\hat{f}(u, v)] \quad (2.14)$$

with:

$$Re[\hat{f}(u, v)] = \frac{1}{MN} \sum_{l=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} f(k, l) \cos(2\pi(\frac{uk}{M} + \frac{vl}{N})) \quad (2.15)$$

$$Im[\hat{f}(u, v)] = -\frac{1}{MN} \sum_{l=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} f(k, l) \sin(2\pi(\frac{uk}{M} + \frac{vl}{N})) \quad (2.16)$$

$$(2.17)$$

It can also be written using its modulus and argument:

$$\hat{f}(u, v) = | \hat{f}(u, v) | \exp\{i \arg \hat{f}(u, v)\} \quad (2.18)$$

$| \hat{f}(u, v) |^2$ is called power spectrum, and $\Theta(u, v) = \arg \hat{f}(u, v)$ the phase.

Two other related transforms are the cosine and the sine transform. The discrete cosine transform is defined by:

$$\begin{aligned} DCT(u, v) &= \frac{1}{\sqrt{2N}} c(u)c(v) \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} f(k, l) \cos\left(\frac{(2k+1)u\pi}{2N}\right) \cos\left(\frac{(2l+1)v\pi}{2N}\right) \\ IDCT(k, l) &= \frac{1}{\sqrt{2N}} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} c(u)c(v) DCT(u, v) \cos\left(\frac{(2k+1)u\pi}{2N}\right) \cos\left(\frac{(2l+1)v\pi}{2N}\right) \end{aligned}$$

with $c(i) = \frac{1}{\sqrt{2}}$ when $i = 0$ and 1 otherwise.

2.5.2 Fourier transform (float): `fft_f_2d`

The program *fft_f_2d* computes the Fourier transform. Only FITS format is supported by FFT routines. The output file name must not have any suffix. The program adds “_re.fits” to the output file name for the real part of the Fourier transform, and “_im.fits” for the imaginary part.

USAGE: `fft_f_2d options image_in image_out`

where options are:

- [-r]
Apply an inverse Fourier transform. Default is not to do this.
- [-s]
Zero frequency is by default in the middle of the image. If the “s” option is set, zero frequency is at the bottom left.

The input image must be an integer or a floating image, and the output image is a complex one.

Examples:

- `fft_f_2d imag.fits cf_imag`
Computes the Fourier transform of `imag.fits`.
- `im_info cf_imag_re.fits`
Print information about the real part of the Fourier transform.
- `im_info cf_imag_im.fits`
Print information about the imaginary part of the Fourier transform.

2.5.3 Fourier transform (complex): `fft_cf_2d`

The program *fft_cf_2d* computes the Fourier transform.

USAGE: `fft_cf_2d options image_in image_out`

where options are:

- [-r]
Apply an inverse Fourier transform. Then the input and output images are complex.
Default is not to do this.
- [-s]
Zero frequency is by default in the middle of the image. If the “s” option is set, zero frequency is at the bottom left.

Examples:

- `fft_cf_2d imag inv_imag`
Computes the inverse Fourier transform of `imag.fits` (see previous section).
- `im_info inv_imag_re.fits`
Prints information about the real part of the inverse Fourier transform.
- `im_info cf_imag_im.fits`
Prints information about the imaginary part of the inverse Fourier transform.

2.5.4 Cosine transform: im_dct

The program `im_dct` computes the cosine transform of an $N \times N$ image, where N is a power of 2.

USAGE: im_dct options image_in image_out

where options are:

- `[-r]`
Apply an inverse cosine transform.
Default is not to do this.

Example:

- `im_dct imag.fits dct_imag`
Computes the cosine transform of `imag.fits`.

2.5.5 Sinus transform: im_dst

The program `im_dst` computes the sine transform of an $N \times N$ image, where N is a power of 2.

USAGE: im_dst options image_in image_out

where options are:

- `[-r]`
Apply an inverse sine transform.
Default is no.

Example:

- `im_dst imag.fits dst_imag`
Computes the sine transform of `imag.fits`.

2.6 Mathematical morphology

2.6.1 Introduction

Originally developed by Matheron [128, 129] and Serra [165], mathematical morphology is based on two operators: the *infimum* (denoted \wedge) and the *supremum* (denoted \vee). The infimum of a set of images is defined as the greatest lower bound while the *supremum* is defined as the least upper bound. The basic morphological transformations are erosion, dilation, opening and closing. For gray-level images, they can be defined in the following way:

- *dilation* consists of replacing each pixel of an image by the maximum of its neighbors.

$$\delta_B(f) = \bigvee_{b \in B} f_b$$

where f stands for the image, and B denotes the structuring element, typically a small convex set such a square or disk.

The dilation is commonly known as “fill”, “expand”, or “grow.” It can be used to fill “holes” of a size equal to or smaller than the structuring element. Used with binary images, where each pixel is either 1 or 0, dilation is similar to convolution. Over each pixel of the image, the origin of the structuring element is overlaid. If the image pixel is nonzero, each pixel of the structuring element is added to the result using the “or” operator.

- *erosion* consists of replacing each pixel of an image by the minimum of its neighbors:

$$\epsilon_B(f) = \bigwedge_{b \in B} f_{-b}$$

where f stands for the image, and B denotes the structuring element.

Erosion is the dual of dilation. It does to the background what dilation does to the foreground. This operator is commonly known as “shrink” or “reduce”. It can be used to remove islands smaller than the structuring element. Over each pixel of the image, the origin of the structuring element is overlaid. If each nonzero element of the structuring element is contained in the image, the output pixel is set to one.

- *opening* consists of doing an erosion followed by a dilations.

$$\alpha_B = \delta_B \epsilon_B \text{ and } \alpha_B(f) = f \circ B$$

- *closing* consists of doing a dilation followed by an erosion.

$$\beta_B = \epsilon_B \delta_B \text{ and } \beta_B(f) = f \bullet B$$

In a more general way, *opening* and *closing* refer to morphological filters which respect some specific properties [25].

The skeleton of an object in an image is a set of lines that reflect the shape of the object. The set of skeletal pixels can be considered to be the medial axis of the object. More details can be found in [175].

2.6.2 Erosion: im_erosode

The *im_erosode* program implements the erosion operator on grayscale images.

USAGE: im_erosode options image_in image_out

where options are:

- **[-n erosion_number]**
Number of times the erosion is repeated. Default is 1.
- **[-s structural_element]**
Type of structuring element:

1. square
2. cross
3. circle

Default is 3.

- **[-d Dim]**
Dimension of the structuring element. Only used for square and circle structuring elements.
Default is 3.

Example:

- `im_erosode imag.fits erode_imag`
Computes the erosion transform of `imag.fits`.

2.6.3 Dilation: im_dilate

The *im_dilate* program implements the dilation operator on grayscale images by the structuring element.

USAGE: im_dilate options image_in image_out

where options are:

- **[-n dilation_number]**
- **[-s structural_element]**
- **[-d Dim]**

See 2.6.2 for description of options.

Example:

- `im_dilate imag.fits dilate_imag`
Computes the dilation transform of `imag.fits`.

2.6.4 Opening: im_opening

The *im_opening* program implements the opening (erosion followed by a dilation) operator on grayscale images.

USAGE: **im_opening options image_in image_out**

where options are:

- [-n **opening_number**]
- [-s **structural_element**]
- [-d **Dim**]

See 2.6.2 for description of options.

Example:

- im_opening imag.fits opening_imag
Computes the opening transform of imag.fits.

2.6.5 Closing: im_closing

The *im_closing* program implements the closing (dilation followed by an erosion) operator on grayscale images.

USAGE: **im_closing options image_in image_out**

where options are:

- [-n **closing_number**]
- [-s **structural_element**]
- [-d **Dim**]

See 2.6.2 for description of options.

Example:

- im_closing imag.fits closing_imag
Computes the closing transform of imag.fits.

2.6.6 Morphological skeleton: im_thin

The *im_thin* program returns the “skeleton” of a bi-level image. The result is a byte type image in which skeletal pixels are set to 1 and all other pixels are zero.

USAGE: **im_thin options Threshold image_in image_out**

All values lower than *Threshold* are set to zero, and values higher are set to 1. The skeleton is calculated on this bi-level image.

Example:

- im_thin 50 imag.fits skel_imag
Computes the skeleton of imag.fits.

2.7 Principal component analysis

2.7.1 Introduction

Principal component analysis (PCA), also often referred to as the eigenvector, Hotelling, or Karhunen-Loëve transform [102, 115, 94] allows us to transform discrete signals into a sequence of uncorrelated coefficients. Considering a population $D(1..M, 1..N)$ of M signals or images of dimension N , the PCA method expresses the data set D by:

$$D = U\Lambda^{\frac{1}{2}}V^t \quad (2.19)$$

and

$$DD^t = U\Lambda U^t \quad (2.20)$$

$$D^t D = V\Lambda V^t \quad (2.21)$$

where Λ is the diagonal matrix of eigenvalues of the covariance matrix $C = DD^t$, the columns of U are the eigenvectors of DD^t , and the columns of V contain the eigenvectors of $D^t D$.

For a signal $d(1..N)$ we have,

$$d = \sum_{i=1}^M \sqrt{\lambda_i} u_i v_i^t \quad (2.22)$$

where λ_i are the eigenvalues of the covariance matrix and where u_i and v_i are the i th columns of U and V . The v_i vectors can also be calculated by [17]:

$$V(i, j) = v_i(j) = \frac{1}{\sqrt{\lambda_i}} \sum_k D(i, k) u_i(j) \quad (2.23)$$

In practice, we construct the matrix A whose rows are formed from the eigenvectors of C [85], ordered by decreasing order of associated eigenvalues. A vector $x(1..M)$ can then be transformed by:

$$y = \Lambda^{-\frac{1}{2}} A(x - m_x) \quad (2.24)$$

where m_x is the mean value of x . Because the rows of A are orthonormal vectors, $A^{-1} = A^t$, any vector x can be recovered from its corresponding y by:

$$x = \Lambda^{\frac{1}{2}} A^t y + m_x \quad (2.25)$$

The Λ matrix multiplication can be seen as a normalization. Building A from the correlation matrix instead of the covariance matrix leads to another kind of normalization, and the Λ matrix can be suppressed ($y = A(x - m_x)$ and $x = A^t y + m_x$). Then the norm of y will be equal to the norm of x .

2.7.2 Transform: im_pca

The *im_pca* program applies a PCA to a series of images, and prints the correlation matrix, its eigenvalues, and its eigenvectors. If the “-w” option is set, the eigenvector images are written to disk.

USAGE: im_pca options image_list

where options are:

- **[-w]**
Write to disk the eigenvector images.
File names are: pca_x
where x is the eigenvector number.
- **[-M]**
Do not subtract the mean.

Examples:

- **im_pca -w imag1.fits imag2.fits imag3.fits imag4.fits**
Print the correlation matrix, and the eigenvalues. Write to disk the following files (eigenvectors):
pca_1.fits, pca_2.fits, pca_3.fits, pca_4.fits.
- **im_pca -M -w imag1.fits imag2.fits imag3.fits imag4.fits**
Ditto, but do not subtract the mean before calculating the correlation matrix.

2.7.3 Transform: im_pca_rec

The command *im_pca_rec* carries out the same operations as *im_pca*, and performs a reconstruction from a subset of eigenvectors.

USAGE: im_pca_rec options image_list output_prefix_file_name

where options are:

- **[-w]**
Write to disk the eigenvector images.
File names are: pca_x
where x is the eigenvector number.
- **[-M]**
Do not subtract the mean.
- **[-F NbrEigenVect]**
Number of (largest) eigenvectors used for the reconstruction.
Default is set to the number of images (⇒ output = input).
- **[-K EigenVect_Number]**
Eigenvector number which will not be used for the reconstruction.

Examples:

- **im_pca_rec -F 1 imag1.fits imag2.fits imag3.fits imag4.fits imaout**
Apply the PCA on the four images, and reconstruct the four images from only the first eigenvector. The four reconstructed image names are imaout_1.fits, imaout_2.fits, imaout_3.fits, and imaout_4.fits.

- `im_pca_rec -K 2 -K 3 imag1.fits imag2.fits imag3.fits imag4.fits imaout`
Omit eigenvectors 2 and 3 in the reconstruction.

Chapter 3

MR/1 Multiresolution

3.1 Multiscale Transform

3.1.1 Introduction

The Morlet-Grossmann definition [87] of the continuous wavelet transform for a 1-dimensional signal $f(x) \in L^2(R)$, the space of all square integrable functions, is:

$$W(a, b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} f(x) \psi^* \left(\frac{x-b}{a} \right) dx \quad (3.1)$$

where:

- $W(a, b)$ is the wavelet coefficient of the function $f(x)$
- $\psi(x)$ is the analyzing wavelet
- $a (> 0)$ is the scale parameter
- b is the position parameter

Many discrete wavelet transform algorithms have been described [191]. The most widely-known one is certainly the orthogonal one, proposed by Mallat [124] and its bi-orthogonal version. Using the orthogonal wavelet transform (OWT), a signal s can be decomposed by:

$$s(l) = \sum_k c_{J,k} \phi_{J,l}(k) + \sum_k \sum_{j=1}^J \psi_{j,l}(k) w_{j,k} \quad (3.2)$$

with $\phi_{j,l}(x) = 2^{-j} \phi(2^{-j}x - l)$ and $\psi_{j,l}(x) = 2^{-j} \psi(2^{-j}x - l)$, where ϕ and ψ are respectively the scaling function and the wavelet function. J is the number of resolutions used in the decomposition, w_j the wavelet (or details) coefficients at scale j , and c_J is a coarse or smooth version of the original signal s . Thus, the algorithm outputs $J+1$ subband arrays. The indexing is such that, here, $j = 1$ corresponds to the finest scale (high frequencies). Coefficients $c_{j,k}$ and $w_{j,k}$ are obtained by means of the filters h and g :

$$\begin{aligned} c_{j+1,l} &= \sum_k h(k-2l) c_{j,k} \\ w_{j+1,l} &= \sum_k g(k-2l) c_{j,k} \end{aligned} \quad (3.3)$$

where h and g verify:

$$\begin{aligned}\frac{1}{2}\phi\left(\frac{x}{2}\right) &= \sum_k h(k)\phi(x-k) \\ \frac{1}{2}\psi\left(\frac{x}{2}\right) &= \sum_k g(k)\phi(x-k)\end{aligned}\quad (3.4)$$

and the reconstruction of the signal is performed with:

$$c_{j,l} = 2 \sum_k [\tilde{h}(k+2l)c_{j+1,k} + \tilde{g}(k+2l)w_{j+1,k}] \quad (3.5)$$

where the filters \tilde{h} and \tilde{g} must verify the conditions of dealiasing and exact reconstruction:

$$\begin{aligned}\hat{h}(\nu + \frac{1}{2})\hat{\tilde{h}}(\nu) + \hat{g}(\nu + \frac{1}{2})\hat{\tilde{g}}(\nu) &= 0 \\ \hat{h}(\nu)\hat{\tilde{h}} + \hat{g}(\nu)\hat{\tilde{g}}(\nu) &= 1\end{aligned}\quad (3.6)$$

By this algorithm, the transformation of an image is an image.

3.1.2 Methods

The application of the OWT to image compression and filtering has lead to impressive results compared to previous methods. However some problems related to the OWT may impact on their use in some applications, and include the following.

1. Edge representation: If the OWT performs better than the FFT at representing edges in an image, it is still not optimal. There is only a fixed number of directional elements independent of scale, and there is no highly anisotropic element [33]. For instance, the Haar 2D wavelet transform is optimal for finding features with a ratio $length/width = 2$, and a horizontal, vertical, or diagonal orientation.
2. Isotropic feature representation: the 2D OWT [124] leads to a wavelet transform with three wavelet functions (at each scale there are three wavelet coefficient sub-images) which does not simplify the analysis and the interpretation of the wavelet coefficients. An isotropic transform seems more appropriate for images containing features or objects with no favored orientation (such as astronomical or medical images).
3. Negative values: By definition, the wavelet coefficient mean is zero. Every time we have a positive structure at a scale, we have negative values surrounding it. These negative values often create artifacts during the restoration process, or complicate the analysis.
4. Point artifacts: For example, cosmic ray hits in optical astronomy can “pollute” all the scales of the wavelet transform. The wavelet transform is non-robust relative to such real or detector faults.
5. Integer values: The OWT produce floating values which are not easy to handle for lossless image compression.

These problems have lead to the development of other multiscale representations. Some of the other algorithms do not produce an image, but a pyramid, or cube. So we separate the different multiscale transform algorithms into classes, depending on the output data type. Other multiscale methods, which are nonlinear can also be categorized in this way. We distinguish between fives classes of multiscale transform:

1. transforms which produce *cubes*
2. transforms which produce *pyramids*
3. transforms which produce *half pyramids*
4. transforms which produce *images* (non-redundant transforms)
5. transforms which produce directional *cubes*

The first three, and the last, are redundant (i.e. there are more pixels in the transformation than in the input data).

Cube transform

For the first class, the input image I can be expressed as [21, 140, 138]:

$$I(x, y) = c_p(x, y) + \sum_{j=1}^p w_j(x, y) \quad (3.7)$$

w_j (for $j = 1 \dots p$) and c_p represent the transformation of I . c_p is a very smoothed version of the image I , and w_j is the image which contains information at scale j . The transformation is defined by $n = p + 1$ images (n = number of scales). The p images have zero mean (or approximately zero, for nonlinear transforms). Each of them corresponds to the information at a given scale, i.e. structure of a given size in the input image. Compact structures (with size of one or two pixels) will be found at the first scale ($j = 1$). For this class of transformation, which is very redundant, the amount of data is multiplied by the number of scales n . Therefore this takes a lot of memory, but the multiresolution coefficients ($w_j(x, y)$) are easy to interpret.

Pyramidal transform

The second class of transformation is less redundant. The first scale has the same size as the image, but for the other scales, the number of pixels is reduced by four at each resolution. Thus, if N^2 is the number of pixels of I , the number of pixels of the transformation is $4/3N^2$.

Half-pyramidal transform

This transform [20] is relatively close to the previous one, but the two first scales are not decimated (i.e. they have the same size as the input image). See Appendix D for more details.

Non-redundant transform

The last class is completely non-redundant, and the number of pixels is the same as in the input image. This means that it is an image. Figure 3.1 shows the representation of an image using the Mallat transform [124, 7, 51]. At a given resolution, the image is shared between four parts. Three subimages correspond to details of the image in the horizontal, vertical, and diagonal directions, and the last part corresponds to the image at a lower resolution. The process can then be repeated on the image at the lower resolution. The Haar wavelet transform, lifting scheme transform, and the G transform (which is a nonlinear transform based on the minimum and the maximum) produce the same kind of representation.

The Feauveau transform [72] (see Figure 3.2) is not redundant, but the representation is different. There is no prioritized direction, and we have an intermediate resolution (half resolution).

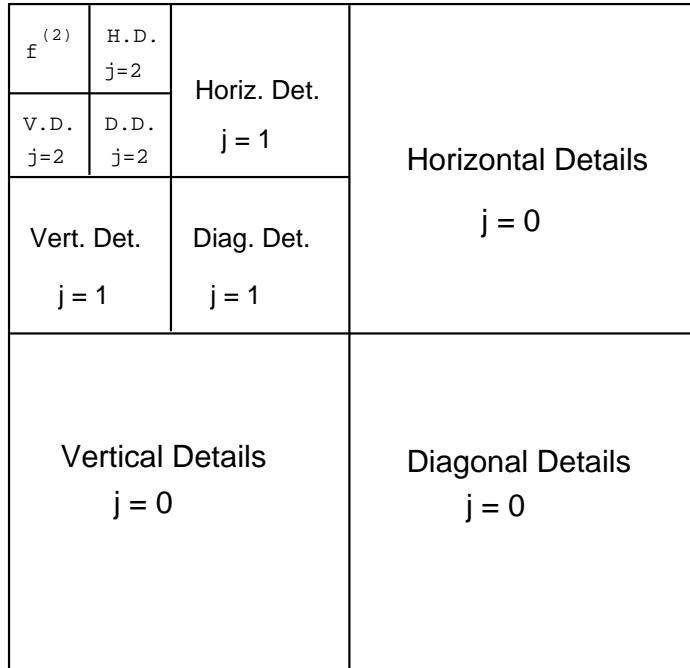


Figure 3.1: Mallat wavelet transform representation of an image.

Directional cube transform

In this case, just as for the orthogonal Mallat transform, the analysis is directional, but there is no decimation. This means that the number of output bands is equal to the number of scales multiplied by the number of directions, and each band has the same number of pixels as the input image. The dyadic wavelet transform uses two directions (vertical and horizontal), and the undecimated wavelet transform uses three directions (vertical, horizontal, and diagonal).

Algorithms

Since the output differs following the chosen algorithm, we will define the term *band* as a subimage included in a given dyadic scale. For the cube and pyramid transforms, the number of bands is equal to the number of scales. For the Mallat algorithm and the undecimated bi-orthogonal wavelet transform, we have three bands per scale. For the Feauveau and the dyadic wavelet transform, we have two bands per scale. Table 3.1 indicates for each MR/1 algorithm its class, the number of bands per scale, and if it is redundant and linear.

3.2 Multiresolution object

3.2.1 Multiresolution transform of image: `mr_transform`

The program `mr_transform` computes the multiresolution transform of an image. This transform can be a linear transform (wavelet transform) or a morphological transform (based on the median, or the minimum and the maximum). The output file which contains the transformation

Multiscale algorithm	Class	Redundant	Bands per scale	Linear
1: linear WT	Cube	Yes	1	Yes
2: B-spline WT	Cube	Yes	1	Yes
3: wavelet transform in Fourier space	Cube	Yes	1	Yes
4: morphological median transform (MT)	Cube	Yes	1	No
5: morphological minmax transform	Cube	Yes	1	No
6: pyramidal linear WT	Pyramid	Yes	1	Yes
7: pyramidal B-spline WT	Pyramid	Yes	1	Yes
8: pyramidal WT in Fourier space: alg. 1	Pyramid	Yes	1	Yes
9: pyramidal WT in Fourier space: alg. 2	Pyramid	Yes	1	Yes
10: pyramidal median transform	Pyramid	Yes	1	No
11: pyramidal Laplacian	Pyramid	Yes	1	No
12: morph. pyramidal minmax transform	Pyramid	Yes	1	No
13: decomposition on scaling function	Pyramid	Yes	1	No
14: (bi-) orthogonal wavelet transform 14-1: Antonini 7/9 filter 14-2: Daubechies filter 4 14-3: Biorthogonal 2/6 Haar filters 14-4: Biorthogonal 2/10 Haar filters 14-5: Odegard 7/9 filters 14-6: User's filters	Image	No	3	Yes
15: Feauveau WT	Image	No	2	Yes
16: Feauveau WT without undersampling	Cube	Yes	2	Yes
17: G transform (morph. min-max alg.)	Image	No	3	No
18: Haar wavelet transform	Image	No	3	Yes
19: Half-pyramidal transform	Half-Pyramid	Yes	1	Yes
20: Mixed Half-pyramidal and MT	Half-Pyramid	Yes	1	No
21: Dyadic wavelet transform	Dir. Cube	Yes	2	Yes
22: Mixed WT and PMT method	Pyramid	Yes	1	No
23: Undecimated Haar transform	Dir. Cube	Yes	1	Yes
24: Undecimated bi-orthog. wavelet trans. (filters 14-1 to 14-6 are available)	Dir. Cube	Yes	3	Yes
25: Wavelet transform via lifting scheme 25-1: CDF WT 25-2: Median prediction 25-3: integer Haar WT 25-4: integer CDF WT 25-5: integer (4,2) interpolating transform 25-6: Antonini 7/9 filter 25-7: integer Antonini 7/9 filter	Image	No	3	Yes No No No No Yes No

Table 3.1: Multiscale transform algorithms.

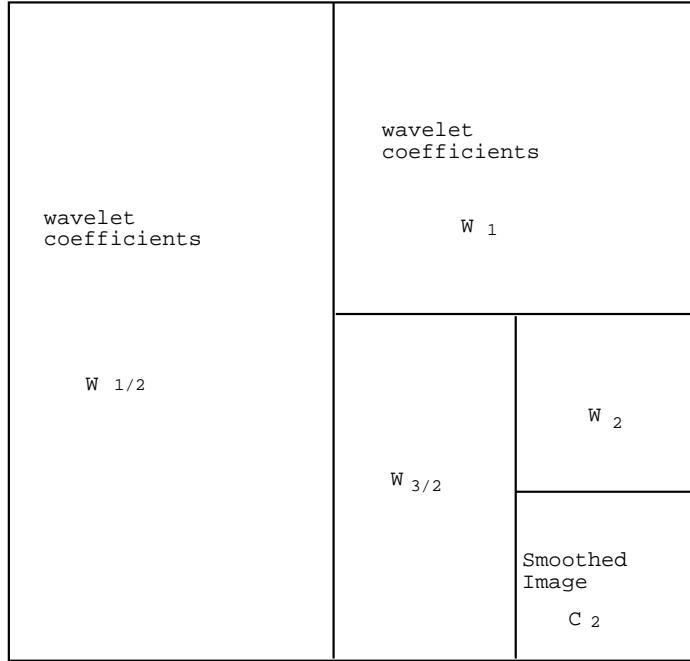


Figure 3.2: Feauveau wavelet transform representation of an image.

has a suffix, .mr. If the output file name given by the user does not contain this suffix, it is automatically added. The “.mr” file is a FITS format file, and can be manipulated by any package dealing with FITS format, or using the *mr_extract* program.

USAGE: `mr_transform option image_in multiresolution_transform_out`

where options are:

- `[-t type_of_multiresolution_transform]`
 1. linear wavelet transform: à trous algorithm
 2. B-spline wavelet transform: à trous algorithm
 3. wavelet transform in Fourier space
 4. morphological median transform
 5. morphological minmax transform
 6. pyramidal linear wavelet transform
 7. pyramidal B-spline wavelet transform
 8. pyramidal wavelet transform in Fourier space: wavelet = between two resolutions
 9. pyramidal wavelet transform in Fourier space: wavelet = difference between the square of two resolutions

10. pyramidal median transform
11. pyramidal Laplacian
12. morphological pyramidal minmax transform
13. decomposition on scaling function
14. (bi-) orthogonal wavelet transform.
Antonini 7/9 filters [7] are used by default, with an L_1 normalization. The filters can be changed using the “-T” option, and an L_2 normalization is obtained by “-L” option.
15. Feauveau wavelet transform
16. Feauveau wavelet transform without undersampling
17. G transform (non-redundant morphological min-max algorithm)
18. Haar wavelet transform (L_2 normalization).
19. Half-pyramidal wavelet transform (HPWT)
20. Mixed HPWT and Median method
21. dyadic wavelet transform
22. Mixed WT and PMT method (WT-PMT)
23. Undecimated Haar transform: à trous algorithm
24. Undecimated (bi-) orthogonal wavelet transform.
Antonini 7/9 filters [7] are used by default, with an L_1 normalization. The filters can be changed using the “-T” option, and an L_2 normalization is obtained by “-L” option.
25. Wavelet transform via lifting scheme

Default is 2.

- **[-T type_of_filters]**

1. Antonini 7/9 filters.
2. Daubechies filter 4.
3. Biorthogonal 2/6 Haar filters.
4. Biorthogonal 2/10 Haar filters.
5. Odegard 7/9 filters.
6. User’s filters.

Default is Antonini 7/9 filters.

This option is only available if the chosen transform method is the (bi-) orthogonal transform (-t 14 or -t 24).

- **[-L]**

Use an L_2 normalization. Default is L_1 .

- **[-u]**

Number of undecimated scales used in the undecimated wavelet transform. Default is all scales.

- **[-l type_of_lifting_transform]**

1. Lifting scheme: CDF WT.
2. Lifting scheme: median prediction.
3. Lifting scheme: integer Haar WT.
4. Lifting scheme: integer CDF WT.

5. Lifting scheme: integer (4,2) interpolating transform.
6. Lifting scheme: Antonini 7/9 filters.
7. Lifting scheme: integer Antonini 7/9 filters.

Default is Lifting scheme: integer Haar WT.

This option is only available if the chosen transform method is the lifting scheme (-t 24).

- **[-n number_of_scales]**

Number of scales used in the multiresolution transform. Default is 4.

- **[-x]**

Write all bands separately as images with prefix “band_j”, $j=1,\dots,NbrBand$ (j being the band number). Note also that $j = 1$ corresponds to the smallest scale (highest frequency band), and j increases for larger scales.

- **[-B]**

Same as x option, but interpolate the bands by block in order to have the same size as the original image.

- **[-c iter]**

An exact reconstruction cannot be achieved from all pyramidal transformations, and a few iterations may be necessary to refine the result. Therefore, for transformations (6,7,8,9,10,12), this option is valid, and *iter* sets the number of iterations. Generally, three is enough. Default is no iteration.

- **[-u number_of_undecimated_scales]**

Number of undecimated scales used in the undecimated wavelet transform (-t 24). By default, all scales are undecimated.

The distribution of the transforms into classes is the following:

1. cube: transform 1 to 5, and 16, and 23.
2. pyramid: transforms 6 to 13, 22.
3. half-pyramid: transform 19,20.
4. image: 14, 15, 17, 18, 25.
5. directional cube: 21,24.

Options “-x” and “-B” are mutually exclusive. If one of these options is set, the second parameter is used also as prefix for the creation of the band-images.

Option “-u” is only valid if the undecimated wavelet transform (-t 24) is selected.

Wavelet transforms using the Fourier transform require the input image to be a square image, with a number of pixels which is an integer power of 2.

The result is stored in a file (suffix “.mr”), and images (or bands) of the transformation can be extracted by using the *mr_extract* program.

Examples:

- `mr_transform -t 10 image.d pyr_trans.mr`

Apply the pyramidal median transform to an image, and store the result in *pyr_trans.mr*.

- `mr_transform -t 10 -x image.d pyr_trans.mr`
Same as before, but write all scales separately in files:
(band_1_pyr_trans.d, band_2_pyr_trans.d, etc.).
- `mr_transform -t 8 -B image.d pyr_wave.mr`
Apply a pyramidal wavelet transform, and each scale is interpolated to the input image size before being saved in a file of name “band_j_pyr_wave.d”, where j is the scale number.
- `mr_transform -t 14 -T 4 image.d bio_wave.mr`
Bi-orthogonal wavelet transform using Odegard 7/9 filters.
- `mr_transform -t 25 -l 3 image.d bio_wave.mr`
Integer Haar wavelet transform.

Figure 3.3 shows the galaxy NGC2997, and Figure 3.4 its wavelet transform using the à trous algorithm with four scales (three wavelet scales, and the last smooth array). Each scale has the same size as the original image.

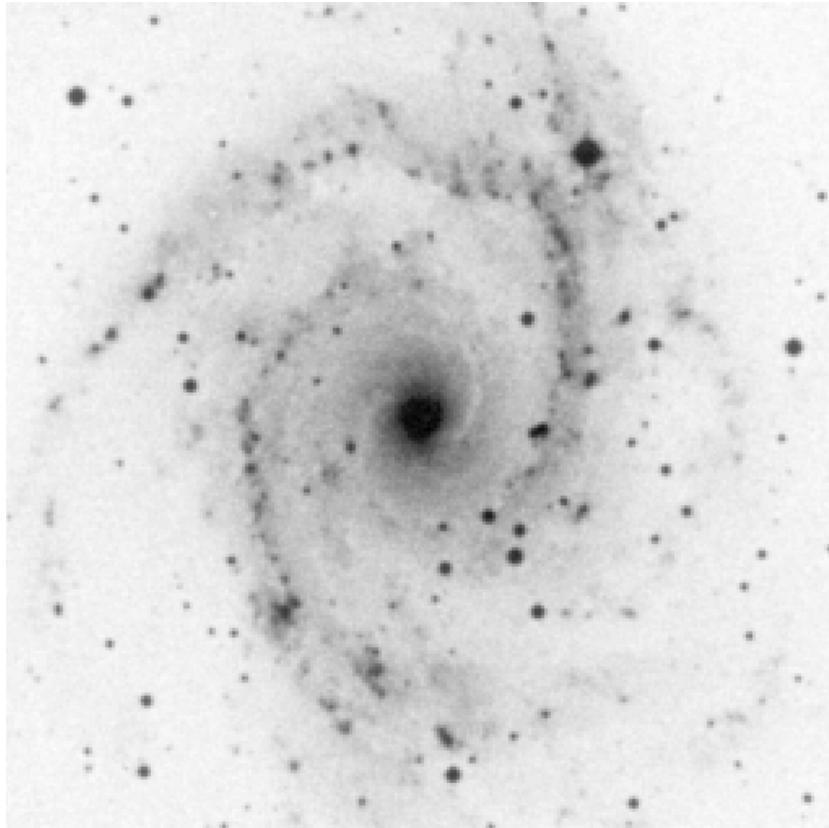


Figure 3.3: Galaxy NGC2997.

3.2.2 How to select a filter bank?

The (bi-) orthogonal wavelet transform (OWT) and the undecimated wavelet transform (UWT) allow the user to select between five different filter banks (see Table 3.1, from 14-1 to 14-5), or to use one's own filter bank. In this case, a file must contain the coefficient values of the filters

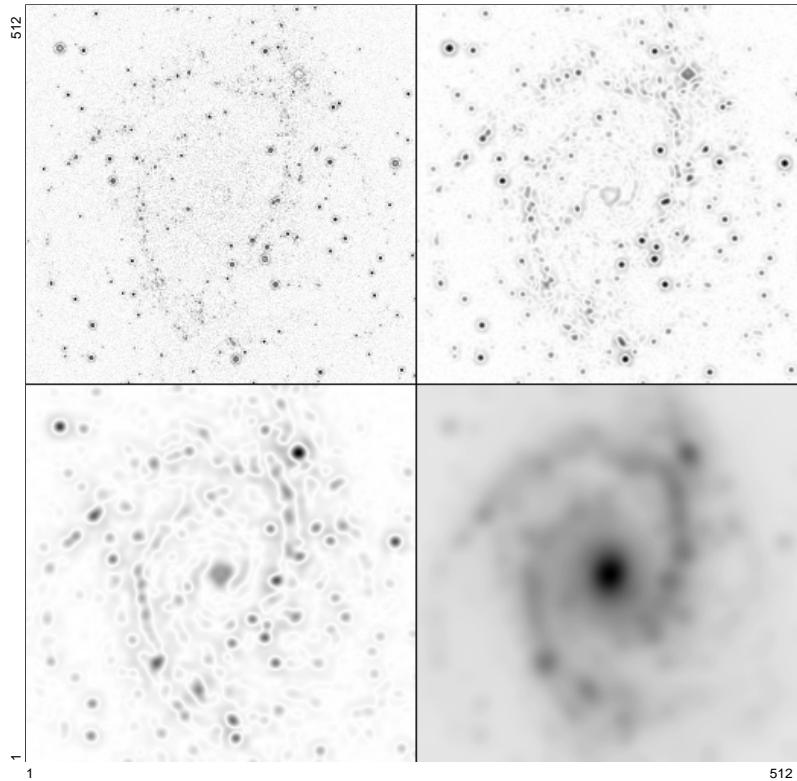


Figure 3.4: Wavelet transform of NGC 2997 by the à trous algorithm.

h and \tilde{h} . The file format is the Bath Wavelet Warehouse file format (see <http://dmsun4.bath.ac.uk/wavelets/warehouse.html> for more information). The file is an ASCII file, with the “.wvf” extension, and the general format for describing wavelet filter coefficients is the following:

```
[Range low] [Range high]
[Analysis LP filter coefficients]
.
.
.
[Range low] [Range high]
[Synthesis LP filter coefficients]
```

For example, the Daubechies filter 4 is (the filename is dau4.wvf):

```
0 3
0.4829629131445341
0.8365163037378079
0.2241438680420134
-0.1294095225512604
```

```
0 3  
0.4829629131445341  
0.8365163037378079  
0.2241438680420134  
-0.1294095225512604
```

More than thirty different filters are available at the Bath Wavelet Warehouse. In the MR/1 programs, when the option “-t 14” (for the OWT) or “-t 24” (for the UWT) are chosen, the option “-T 6” or “-T 6,filename” is used for defining a user filter. If no file is given, the MR/1 programs first check if the default filename “**mr1.wvf**” exists in the current directory, and if it does not exist, the environment variable **CEA_FILTER** is tested. Examples of command line syntax for the wavelet transform program are:

- `mr_transform -t 14 -T 6,dau4 image.d dwt_trans.mr`
Bi-orthogonal wavelet transform, using the filters defined in the file “dau4.wvf”.
- `mr_transform -t 24 -T 6 image.d uwt_trans.mr`
Undecimated wavelet transform, using the file “**mr1.wvf**” if it exists, otherwise using the file name contained in the environment variable **CEA_FILTER**.

3.2.3 Extraction of a scale: `mr_extract`

The program `mr_extract` allows the user to extract a scale or a band from a multiresolution transform file (suffix “.mr”).

USAGE: `mr_extract options multiresolution_file output_image`

where options are:

- **[-b band_number]**
Band number to extract. Default is 1.
- **[-s scale_number]**
Scale number to extract. Default is 1.
- **[-x]**
Extract all the bands. The second parameter is used as a prefix. If this option is set, the second parameter **must** only contain the file name, and not the complete path of the file.
- **[-B]**
Interpolate the scale by block, in order to have the same size as the original image. For scale extraction, this option is valid only if the multiresolution transform is pyramidal, and is always valid for band extraction.

Examples:

- `mr_extract -s 1 mr_file.mr scale_1.d`
Extract the first scale of the wavelet transform, and write as a .d image of name “scale_1.d”. If the transform is an orthogonal transform, it will contain all of the transform.
- `mr_extract -b 1 mr_file.mr band_1.d`
Extract the first band of the wavelet transform, and write as a .d image of name “band_1.d”. If the transform is an orthogonal transform, it will contain only the first band (in one single direction) of the transform.
- `mr_extract -x mr_file.mr toto.d`
Creates the files “toto_band_1.d”, ..., “toto_band_j.d”.

3.2.4 Insertion of an image: `mr_insert`

The program `mr_insert` replaces a scale or a band by some image, by inserting it in the multiresolution transform file. The scale (or the band) and the image must have the same size.

USAGE: `mr_insert options multiresolution_file input_image`

where options are:

- **[-b band_number]**
Band number to insert. Default is 1.
- **[-s scale_number]**
Scale number to insert. Default is 1.

multiresolution_file is the file (.mr) which contains the multiresolution transformation.

input_image is the image which must replace the band image.

band_number specifies the band number to be replaced. Default is 1. The multiresolution transform is updated.

Example:

- `mr_insert -b 3 mr_file.mr image.d`
Insert an image at the third scale of the multiresolution file.

3.2.5 Reconstruction: `mr_recons`

The program `mr_recons` reconstructs an image from its multiresolution transform.

USAGE: `mr_recons multiresolution_file image_out`

multiresolution_file is the file (.mr) which contains the multiresolution transformation, *output_image* is the output reconstructed image.

3.3 Multiresolution tools**3.3.1 Visualization: `mr_visu`**

All scales of the multiresolution transform are normalized to 1, and inserted in one image, which can be visualized by a graphics tool (xv, IDL, etc.). The input file can either be an image or a multiresolution file. In the second case options “`-t`, `-n`”, which define the type of transformation and the number of scales, are not used. By default the output image is represented in graylevel, with each scale normalized to 255. The normalization can be avoided using the “`-c`” option. Two other representations are also available, the perspective one (all scales are represented in a 3-dimensional way), and the contour one. In the contour representation, one contour is plotted per scale. The contour level is determined from the standard deviation of the first scale and the *nsigma* parameter (which can be modified by the “`-s`” option). Non-isotropic transforms cannot be used for perspective or contour representation.

Figure 3.5 shows the wavelet transform of the galaxy NGC2997, with five scales, and using a perspective representation.

USAGE: `mr_visu image_in_or_MRfile image_out`

where options are:

- `[-t type_of_multiresolution_transform]`
see 3.2.1.
- `[-T type_of_filters]`
see 3.2.1.
- `[-L]`
see 3.2.1.
- `[-n number_of_scales]`
see 3.2.1.
- `[-V Type_Visu]`
 1. Graylevel
 2. Contour
 3. Perspective

Default is 1.

- **[-b]**
Save output image as bi-level. Only used if *Type_Visu* is equal to 2 or 3. Default is not to save.
- **[-c]**
Do not apply a normalization on the multiresolution coefficient. Only used if *Type_Visu* is equal to 1.
- **[-i Increment]**
Number of lines of the image which will be used.
If Increment = 3, only one line out of 3 is used.
Only used if *Type_Visu* is equal to 3.
The default value is 1.
- **[-s nsigma]**
Plot contour at $nsigma \cdot \text{Sigma}$ if *Type_Visu* equal to 2.
Threshold value upper $nsigma \cdot \text{Sigma}$ if *Type_Visu* equal to 3.
Default is 3.

3.3.2 Multiresolution support creation: mr_support

The multiresolution support of an image describes in a logical or boolean way if an image I contains information at a given scale j and at a given position (x, y) [184, 139, 143]. If $M^{(I)}(j, x, y) = 1$ (or *true*), then I contains information at scale j and at the position (x, y) . M depends on several parameters:

- The input image.
- The algorithm used for the multiresolution decomposition.
- The noise.
- All constraints we additionally want the support to satisfy.

Such a support results from the data, the treatment (noise estimation, etc.), and from knowledge on our part of the objects contained in the data (size of objects, linearity, etc.). In the most general case, a priori information is not available to us.

The multiresolution support of an image is computed in several steps:

- Step 1 is to compute the wavelet transform of the image.
- Binarization of each scale leads to the multiresolution support.
- A priori knowledge can be introduced by modifying the support.

The last step depends on the knowledge we have of our images. For instance, if we know there is no interesting object smaller or larger than a given size in our image, we can suppress, in the support, anything which is due to that kind of object. This can often be done conveniently by the use of mathematical morphology. Negative detections can sometimes be suppressed too. In the most general setting, we naturally have no information to add to the multiresolution support.

A complete description of how to derive the multiresolution support from the multiresolution coefficients is given in [184, 191]. Several types of noise are taken into account:

1. Gaussian noise (default case).

2. Poisson noise.
3. Poisson and Gaussian noise

For the last two transforms, a variance stabilization transform is applied to the input image, which allows us to obtain a transformed image which contains Gaussian noise with a standard deviation equal to one. After transformation of the image (or the variance stabilized image), each wavelet coefficient $w_j(x, y)$ is compared to the standard deviation at the scale j , σ_j multiplied by a coefficient N_σ . This coefficient specifies the confidence interval we want. If $|w_j(x, y)| > N_\sigma \sigma_j$, the probability that $w_j(x, y)$ is due to noise is less than 1%.

USAGE: `mr_support options image_in multiresolution_file_out`

where options are:

- **`[-t type_of_multiresolution_transform]`**
see 3.2.1.
- **`[-T type_of_filters]`**
see 3.2.1.
- **`[-L]`**
see 3.2.1.
- **`[-u]`**
see 3.2.1.
- **`[-g sigma]`**
The image contains Gaussian noise, and the standard deviation is given by *sigma* . The option should be set only if the user knows the standard deviation of the noise.
- **`[-p]`**
If this option is set, the image is assumed to contain Poisson noise. The default is Gaussian noise, and its standard deviation is automatically estimated.
- **`[-c gain,sigma,mean]`**
The noise is composed of a Gaussian and Poisson component. This is the case for a CCD detector.
noise = Poisson noise and read-out noise
gain = inverse of the CCD gain (unit: DN/e^-)
sigma = standard deviation of the read-out noise (unit: DN) = CCD RON / CCD Gain
mean = mean of the read-out noise
If this option is set,
Noise = Poisson and Gaussian read-out noise
This is generally the case with a CCD.
Note: these parameters must be separated by a comma without spaces.
Example: `-c 0.133,1.733,0`.
If the mean, or sigma and the mean, are omitted, default values are 0.
The gain cannot be omitted if parameter -c is used.
- **`[-n number_of_scales]`**
Number of scales used in the multiresolution transform. Default is 4.
- **`[-s NSigma]`**
The detection level at each scale is determined by the product of the standard deviation of the noise by the *NSigma*. *NSigma* fixes the confidence interval we want. By default, *NSigma* is equal to 3.

- **[-k]**
If this option is set, isolated pixels in the multiresolution support are suppressed. If the PSF is large, then isolated pixels are certainly residual noise, or spurious impulse noise objects, or artifacts. Then we can suppress these pixels in the support. This option can only be used with transformations of class 1 and 2 (which produce cubes and pyramids). Default is not to do this.
- **[-l]**
If this option is set, the morphological dilation operator is applied on each scale of the multiresolution support. This option can only be used with transformations of class 1 and 2 (which produce cubes and pyramids). Default is not to do this.
- **[-w support_file_name]**
If this option is set, a synthetic image is created from the multiresolution support. Default is not to do this.

Examples:

- `mr_support -w support.d -t 14 image.d mr_file.mr`
Creates the multiresolution support of an image, assuming the noise is Gaussian (its standard deviation is automatically estimated). A .d image is created from the support.
- `mr_support -w support.d -p image.d mr_file.mr`
Creates the multiresolution support of an image, assuming the noise follows a Poisson distribution. The chosen multiresolution transform is the default one (à trous algorithm). A .d image is created from the support. This is not a boolean image as in the previous example, because it is obtained by addition of all boolean scales of the multiresolution support.

3.3.3 Statistical information: `mr_info`

Program `mr_info` gives statistical multiresolution information for an image (min, max, sigma, mean, skewness and kurtosis at each scale). If the transform belongs to class 1 or 2 (output is a cube or a pyramid), then additional options allow the significant coefficients to be analyzed, and furnish

- the percentage of significant coefficients at each scale,
- the number of maxima at each scale,
- the number of structures detected at each scale,
- the size in pixels of the largest detected structure at each scale.

If an output file name is given, a 2D array containing statistical information is saved on the disk in the FITS format. The output file is a fits file containing a two dimensional array $T[J - 1, 5]$ (J being the number of bands), with the following the syntax:

- $T[j, 0]$ = standard deviation of the j th ridgelet band.
- $T[j, 1]$ = skewness of the j th ridgelet band.
- $T[j, 2]$ = kurtosis of the j th ridgelet band.
- $T[j, 3]$ = minimum of the j th ridgelet band.
- $T[j, 4]$ = maximum of the j th ridgelet band.

USAGE: `mr_info option image_in [out_statfile]`

where options are

- `[-t type_of_multiresolution_transform]`
- `[-T type_of_filters]`
- `[-L]`
- `[-u]`
- `[-p]`
- `[-g sigma]`
- `[-c gain,sigma,mean]`
- `[-n number_of_scales]`
- `[-s NSigma]`
- `[-k]`
- `[-l]`
- `[-a]`

Significant structures analysis. Default is not to do this.

Options `[t,T,L,u,p,g,c,n,s,k,l]` are the same as those described in section 3.3.2.

Options `[a,p,g,c,s,k,l]` are not allowed for transforms producing images (class 4).

If any of the options `[a,p,g,c,s,k,l]` is set, a significant structures analysis is performed.

Examples:

- `mr_info image.d`
Writes on the standard output the minimum value, the maximum value, the mean value, and the standard deviation of each scale.
- `mr_info -a image.d`
Same as before, but computes also the percentage of significant pixels per scale, and the number of maxima, the number of structures, and the size (in pixels) of the bigger structures.

3.3.4 Multiresolution segmentation: `mr_segment`

Program `mr_segment` binarizes all the scales of the wavelet transform of an image, and applies to each one a segmentation procedure. Each scale of the output multiresolution file contains an image in which values are ranged between 0 and R_j , where R_j is the number of regions found at this scale.

USAGE: `mr_segment option image_in mr_file_out`

where `mr_file_out` is the multiresolution file (“.mr”) which contains the segmented scales.
Options are

- `[-t type_of_multiresolution_transform]`
- `[-T type_of_filters]`

- [-L]
- [-u]
- [-p]
- [-g sigma]
- [-c gain,sigma,mean]
- [-n number_of_scales]
- [-s NSigma]
- [-k]
- [-l]

These options are the same as those described in section 3.3.2.

Example:

- `mr_segment -g 1. image.d mr_file`
Applies the default multiresolution transform (à trous algorithm) to the input image, thresholds the scale assuming Gaussian noise with a standard deviation equal to 1, and applies a segmentation on each scale.

3.3.5 Noise analysis: mr_sigma

Program *mr_sigma* estimates the standard deviation of Gaussian noise in an image [189]. Several methods can be used. One of them (Multiresolution support) is based on the multiresolution. If the option “-p” is set, the noise is considered to be a combination of both Gaussian and Poisson noise. In this case, the program seeks the standard deviation of the Gaussian component of the noise using the generalized Anscombe transform.

USAGE: mr_sigma options image

where options are:

- **[-m type_of_methods]**
 1. 3_sigma_clipping
 2. Median + 3_sigma_clipping
 3. Bspline + 3_sigma_clipping
 4. Multiresolution support
 5. Block method
 6. MAD method (Median of Absolute Deviation)

Default is 2.

- **[-n number_of_scales]** Number of scales used in the multiresolution transform. Default is 4.
- **[-p gain]**
Computes the standard deviation of the Gaussian part of the noise (RON), knowing the gain, in the case where the noise is composed of Poisson and Gaussian noise.

Examples:

- **mr_sigma image.d**
Computes the standard deviation of the Gaussian noise contained in the input image, using the default method.
- **mr_sigma -p 1 image.d**
Computes the standard deviation of the read-out noise, assuming a gain equal to 1.

3.3.6 Background subtraction: mr_background

Program *mr_background* subtracts the background from an image. The background is considered as being the last scale of the pyramidal median transform. The number of scales is automatically calculated in order to have the last scale with a size lower than or equal to 16×16 pixels. This value of 16 can be modified by the “-n” option.

USAGE: mr_background option image_in image_out

where options are

- **[-n number_of_pixels]**
Number of pixels used for background estimation. Default is 16.
- **[-w background_file_name]**
Creates the background image and writes it on disk.
- **[-W MedianWindowSize]**
Median window size using in the Pyramidal Median Transform.

Examples:

- **mr_background image.d im_bgr_free.d**
Computes the pyramidal median transform of the input image, interpolates the last scale to the image, and subtracts it from the input image.
- **mr_background -n 32 -w bgr.d image.d**
The number of scales is calculated so that the last scale will have less than 32 pixels in at least one dimension.

3.3.7 Image comparison: mr_compare

Program *mr_compare* compares a set of images to a reference image. These comparisons are performed both in the direct space and in the wavelet space. By default, only structures above the noise are considered for the comparison. If the *NSigma* parameter is set to zero, then all wavelet coefficients are taken into account. If the *number_of_scales* parameter is set to 1, then the comparison is only done in the direct space. For all images, the minimum, the maximum, the mean and standard deviation are printed. For each image *ima_i*, the wavelet transform of *ima_i* and *ima_error* ($= \text{ref_image} - \text{ima_i}$) is calculated. For each scale, several results are given:

- the percentage of significant wavelet coefficients of *ima_i*.
- the correlation between the wavelet coefficients of *ima_i* and *ref_image*.
- the standard deviation of absolute values of the wavelet coefficients of *ima_error*.
- the root mean square (RMS) of the wavelet coefficients of *ima_error*.

- the minimum value of the wavelet coefficients of *ima_error*.
- the maximum value of the wavelet coefficients of *ima_error*.
- the signal-to-noise ratio (SNR) using the wavelet coefficients of the reference image and those of the error image:

$$SNR = \frac{\sigma^2(W_{ImaRef})}{\sigma^2(W_{Error})}$$

- the signal-to-noise ratio in dB (SNRb)

$$SNRb = 10\log_{10}(SNR)$$

USAGE: `mr_compare option ref_ima ima1 [ima2, [ima3, ...]]`

where options are

- **[-t type_of_multiresolution_transform]**
See 3.2.1. But only transforms 1 to 13 (isotropic transforms) are available.
- **[-p]**
- **[-g sigma]**
- **[-c gain,sigma,mean]**
- **[-n number_of_scales]**
- **[-s NSigma]**

These options are the same as those described in section 3.3.2.

Examples:

- `mr_compare -n 1 image_ref.d ima_1.d ima_2.d`
Computes the images *ima_1* and *ima_2* to *image_ref* in the direct space only.
- `mr_compare -s 0. image_ref.d ima_1.d ima_2.d`
Computes the images *ima_1* and *ima_2* to *image_ref* taking into account all wavelet coefficients.

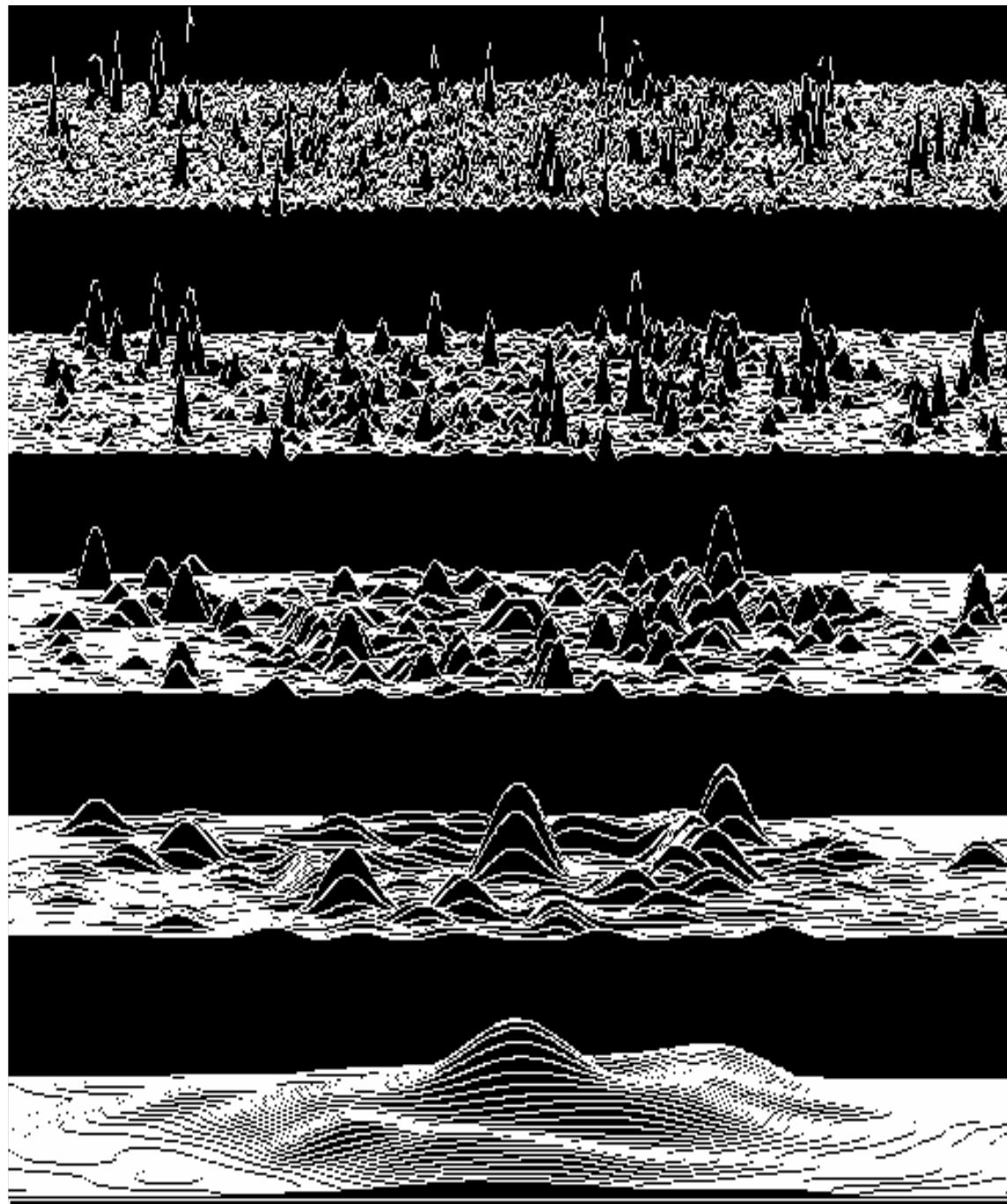


Figure 3.5: 3D perspective view of NGC 2997 wavelet scales.

Chapter 4

MR/1 Image Restoration

4.1 Introduction

4.1.1 Statistical significance test

Images generally contain noise. Hence the wavelet coefficients are noisy too. For filtering, it is necessary to know if a coefficient is due to signal (i.e. it is significant) or to noise. We introduce a statistical significance test for wavelet coefficients. Let \mathcal{H}_j be the hypothesis that the image is locally constant at scale j . Rejection of hypothesis \mathcal{H}_j depends (for a positive coefficient value) on:

$$P = \text{Prob}(W_N > w_j(x, y))$$

and if the coefficient value is negative

$$P = \text{Prob}(W_N < w_j(x, y))$$

Given a threshold, ϵ , if $P > \epsilon$ the null hypothesis is not excluded. Although non-null, the value of the coefficient could be due to noise. On the other hand, if $P < \epsilon$, the coefficient value cannot be due only to the noise alone, and so the null hypothesis is rejected. In this case, a significant coefficient has been detected.

Our noise modeling in the wavelet space is based on the assumption that the noise in the data follows a distribution law, which can be:

- a Gaussian distribution
- a Poisson distribution
- a Poisson + Gaussian distribution (noise in CCD detectors)
- Poisson noise with few events (galaxy counts, X-ray images, point patterns)
- Speckle noise
- Root Mean Square map: we have a noise standard deviation of each data value.

If the noise does not follow any of these distributions, we can derive a noise model from any of the following assumptions:

- it is stationary, and we have a subimage containing a realization of the noise,
- it is additive, and non-stationary,

- it is multiplicative and stationary,
- it is multiplicative, but non-stationary,
- it is undefined but stationary,
- it is additive, stationary, and correlated.

4.1.2 Noise modeling

We summarize here the different noise modeling strategies implemented in MR/1.

1. Gaussian noise

Given stationary Gaussian noise, it suffices to compare $w_j(x, y)$ to $k\sigma_j$.

$$\begin{aligned} \text{if } |w_j| \geq k\sigma_j &\text{ then } w_j \text{ is significant} \\ \text{if } |w_j| < k\sigma_j &\text{ then } w_j \text{ is not significant} \end{aligned} \quad (4.1)$$

2. Poisson noise

If the noise in the data I is Poisson, the transform

$$t(I(x, y)) = 2\sqrt{I(x, y) + \frac{3}{8}} \quad (4.2)$$

acts as if the data arose from a Gaussian white noise model (Anscombe, 1948), with $\sigma = 1$, under the assumption that the mean value of I is large. The image is first transformed, and the same processing is performed as in the Gaussian case. This processing works if the number of photons per pixel is greater than 30. Otherwise the detection levels will be over-estimated, and the case “Poisson noise with few events” should instead be used.

3. Poisson noise + Gaussian

The generalization of the variance stabilizing is:

$$t(I(x, y)) = \frac{2}{\alpha} \sqrt{\alpha I(x, y) + \frac{3}{8}\alpha^2 + \sigma^2 - \alpha g}$$

where α is the gain of the detector, and g and σ are the mean and the standard deviation of the read-out noise.

4. Multiplicative noise

The image is first log-transformed. Then the transformed image is treated as an image with Gaussian additive noise.

5. Non-stationary additive noise

The noise is assumed to be locally Gaussian. So we must consider one noise standard deviation per pixel. The Root Mean Square (RMS) map $R_\sigma(x, y)$ can be furnished by the user, or automatically calculated by estimating for each pixel the standard deviation in a box around it.

From $R_\sigma(x, y)$, we have to compute the noise standard deviation $\sigma_j(x, y)$ for any wavelet coefficient $w_j(x, y)$. $w_j(x, y)$ is obtained by the correlation product between the image I and a function g_j : $w_j(x, y) = \sum_k \sum_l I(x, y)g_j(x + k, y + l)$.

Then we have: $\sigma_j^2(x, y) = \sum_k \sum_l R_\sigma^2(x, y) g_j^2(x + k, y + l)$.

In the case of the à trous algorithm, the coefficients $g_j(x, y)$ are not known exactly, but they can easily be computed by taking the wavelet transform of a Dirac w^δ . The map σ_j^2 is calculated by correlating the square of the wavelet scale j of w^δ by $R_\sigma^2(x, y)$.

6. Non-stationary multiplicative noise

The image is first log-transformed. Then the transformed image is treated as an image with non-stationary additive noise.

7. Undefined stationary noise

A k-sigma clipping is applied at each scale.

8. Undefined noise

The standard deviation is estimated for each wavelet coefficient, by considering a box around it, and the calculation of σ is done in the same way as for non-stationary additive noise. The latter determines a map of variances for the image, and then derives the variances for the wavelet coefficients. “Undefined noise” does not assume additivity of the noise, and so calculates the noise from local variance in the resolution scales.

9. Stationary correlated noise

The noise is stationary, but correlated. This noise modeling requires a noise map, containing a realization of the noise. The threshold at a scale j S_j is found by computing the wavelet transform of the noise map, and using the histogram of S_j to derive the noise probability density function, pdf, of S_j .

10. Poisson noise with few events

This case corresponds to noise with a very small number of photons per pixel (this is the case for instance of X-ray images where the number of photons per pixel is often lower than 1). This special case requires more processing time, due to the fact that a set of autoconvolutions of the histogram of the wavelet function must be calculated. For faster filtering, the table can be pre-computed using the “mr_abaque” program, and then the restoration is carried out using “mr_pfilter” (see section 4.4.1).

11. Speckle noise

See section 4.3.

4.1.3 Filtering Methods

Hard and Soft Thresholding

Many filtering methods have been proposed in the last ten years. *Hard thresholding* consists of setting to 0 all wavelet coefficients which have an absolute value lower than a threshold T_j :

$$\tilde{w}_{j,k} = \begin{cases} w_{j,k} & \text{if } |w_j| \geq T_j \\ 0 & \text{otherwise} \end{cases}$$

where $w_{j,k}$ is a wavelet coefficient at scale j and at spatial position k .

Soft thresholding consists of replacing each wavelet coefficient by the value \tilde{w} where

$$\tilde{w}_{j,k} = \begin{cases} sgn(w_{j,k})(|w_{j,k}| - T_j) & \text{if } |w_j| \geq T_j \\ 0 & \text{otherwise} \end{cases}$$

When the discrete orthogonal wavelet transform is used, it is interesting to note that the hard and soft thresholded estimators are solutions of the following minimization problems:

$$\begin{aligned}\tilde{w} &= \arg_w \min \frac{1}{2} \| y - \mathcal{W}^{-1}w \|_{l^2}^2 + \lambda \| w \|_{l^0}^2 && \text{hard threshold} \\ \tilde{w} &= \arg_w \min \frac{1}{2} \| y - \mathcal{W}^{-1}w \|_{l^2}^2 + \lambda \| w \|_{l^2}^2 && \text{soft threshold}\end{aligned}$$

where y is the input data, \mathcal{W} the wavelet transform operator, and l^0 indicates the limit of l^δ when $\delta \rightarrow 0$. This counts in fact the number of non-zero elements in the sequence.

Several approaches have been proposed for deriving the T_j thresholds.

k-Sigma Thresholding

The k-Sigma approach consists of deriving T_j from the probability of false detection ϵ

$$\text{Prob}(w > T_j) < \epsilon$$

Given stationary Gaussian noise, it suffices to compare $w_{j,k}$ to $k\sigma_j$, where σ_j is the noise standard deviation in band j . Often k is chosen as 3, which corresponds approximately to $\epsilon = 0.002$.

Iterative Filtering

When a redundant wavelet transform is used, the result after a simple hard thresholding can still be improved by iterating. Indeed, we want the wavelet transform of our solution s to reproduce the same significant wavelet coefficients (i.e., coefficients larger than T_j). This can be expressed in the following way:

$$(\mathcal{W}s)_{j,k} = w_{j,k} \text{ if } |w_{j,k}| > k\sigma_j \quad (4.3)$$

where $w_{j,k}$ are the wavelet coefficients of the input data y . Denoting M the multiresolution support (i.e. $M(j, k) = 1$ if $|w_{j,k}| > k\sigma_j$, and 0 otherwise), we want:

$$M.\mathcal{W}s = M.\mathcal{W}y$$

The solution can be obtained by the following Van Cittert iteration [191]:

$$\begin{aligned}s^{n+1} &= s^n + \mathcal{W}^{-1}(M.\mathcal{W}y - M.\mathcal{W}s^n) \\ &= s^n + \mathcal{W}^{-1}(M.\mathcal{W}R^n)\end{aligned} \quad (4.4)$$

where $R^n = y - s^n$. Another approach consists of minimizing the functional

$$J(s) = \| M.\mathcal{W}y - M.\mathcal{W}s \|^2 \quad (4.5)$$

using a minimization method such as the fixed step gradient or the conjugate gradient.

Iterative Filtering using Smoothness Constraint

This method consists in adding a smoothness constraint \mathcal{S} when deriving the solution from the significant coefficients. We solve the following optimization problem [186]:

$$\min \mathcal{S}(\tilde{s}), \quad \text{subject to } \tilde{s} \in C, \quad (4.6)$$

where C is the set of vectors \tilde{s} which obey the linear constraints

$$\begin{cases} \tilde{s} \geq 0, \\ |\mathcal{W}s - \mathcal{W}\tilde{s}| \leq e; \end{cases} \quad (4.7)$$

Here, the second inequality constraint only concerns the set of significant coefficients. We consider two possible functions for \mathcal{S} :

$$\begin{aligned} \mathcal{S}_w(\tilde{s}) &= \min \|\mathcal{W}\tilde{s}\|_{\ell_1} \\ \mathcal{S}_{tv}(\tilde{s}) &= \min \|\tilde{s}\|_{TV} \end{aligned} \quad (4.8)$$

where $\|\cdot\|_{TV}$ is the Total Variation norm, i.e. the discrete equivalent of the integral of the euclidian norm of the gradient.

Universal threshold

Universal thresholding consists of using a threshold [65, 63] $T_j = \sqrt{2 \log(n)}\sigma_j$, where n is the number of pixels in the input data. It ensures with a probability tending to one that all noise is removed. The drawback is that it tends to give an oversmoothed estimator.

SURE Threshold

The SURE threshold minimizes Stein's unbiased risk estimator [66].

MULTI-SURE Thresholding

The SURE method is applied independently on each band of the wavelet transform.

MAD Thresholding

The Median Absolute Deviation (MAD) threshold on a given band j is:

$$T_j = k\sigma_{j,m}$$

where $\sigma_{j,m}$ is the Median Absolute Deviation ($\sigma_{j,m} = \text{MED}(|w_j|)/0.6745$, where MED is the median function). The MAD method does not require any knowledge about the noise such as the noise standard deviation. It is considered as a very good method to denoise data contaminated by correlated noise.

Multiscale Wiener Filtering

A multiresolution Wiener filtering [181] consists of multiplying all coefficients w_j of a given scale j by

$$\alpha_j = \frac{S_j}{S_j + N_j} \quad (4.9)$$

where S_j and N_j are respectively the variance of the signal and of the noise at the scale j .

Hierarchical Multiscale Wiener Filtering

A hierarchical Wiener filtering [181] tries to introduce a prediction into the estimation of \tilde{w}_j . This prediction is obtained from the coefficient w_h at the same position but at the following scale.

$$\tilde{w}_j = \frac{H_j}{N_j + H_j + Q_j} w_j + \frac{N_j}{N_j + H_j + Q_j} w_h \quad (4.10)$$

with:

$$Q_j = \frac{H_j N_j}{S_j} \quad (4.11)$$

where H_j is the variance of the image obtained by taking the difference of the scale j and the following one $j + 1$.

Hierarchical thresholding

For each wavelet coefficient w , the threshold is derived from $N\sigma$ and the signal-to-noise ratio of the wavelet coefficient at the same position but at the next scale. The threshold used here, T_h [181], is equal to $T_j = k\sigma_j$ if $|w_j| \geq T$, and $T_h = Tf(|\frac{w_h}{S_h}|)$ otherwise (S_h is the standard deviation of w_h). The function $f(a)$ must return a value between 0 and 1. The chosen function f is:

- $f(a) = 0$ if $a \geq k$
- $f(a) = 1 - \frac{1}{k}a$ if $a < k$

4.2 Filtering: mr_filter

Program *mr_filter* filters an image. Several methods can be used [181, 188, 187]. Those using multiresolution need noise modeling. For the case of Poisson noise with few events, the *mr_pfilter* program can also be used (see section 4.4.1). The case of speckle noise is treated in section 4.3.

USAGE: **mr_filter option image_in image_out**

where options are

- **[-f type_of_filtering]**
 1. Multiresolution Hard k-Sigma Thresholding.
 2. Multiresolution Soft k-Sigma Thresholding.
 3. Iterative Multiresolution Thresholding.
(see equation 4.4).
 4. Adjoint operator applied to the multiresolution support.
Minimize a functional from the detected wavelet coefficients (see equation 4.5). It is the same method as this which is used by *mr_detect* for individual object reconstruction.
 5. Hierarchical Hard Thresholding.
 6. Hierarchical Wiener filtering.
 7. Multiresolution Wiener filtering.
 8. Median filtering
Standard median filtering
 9. Average filtering
Standard average filtering
 10. B-spline filtering
Convolve the image with a B-spline
 11. Universal Hard Thresholding.
 12. Universal Soft Thresholding.
 13. SURE Hard Thresholding.
 14. SURE Soft Thresholding.
 15. MULTI-SURE Hard Thresholding.
 16. MULTI-SURE Soft Thresholding.
 17. Median Absolute Deviation (MAD) Hard Thresholding.
 18. Median Absolute Deviation (MAD) Soft Thresholding.
 19. Total Variation + Wavelet Constraint
 20. Wavelet Constraint Iterative Methods

Default is multiresolution thresholding.

- **[-t type_of_multiresolution_transform]**
- **[-T type_of_filters]**
- **[-u]**
- **[-g sigma]**
- **[-c gain,sigma,mean]**
- **[-m type_of_noise]**

1. Gaussian noise
The standard deviation is either provided by the user using the option “-g” or it is automatically calculated.
2. Poisson noise
3. Poisson noise + Gaussian noise
Parameter (gain, readout noise, etc.) can be given by the “-c” option. The generalized Anscombe transform is used.
4. Multiplicative noise
5. Non-stationary additive noise
The standard deviation is estimating in a box around each pixel. The size of the box can be fixed by the “-S” option. By default, the standard deviation of the pixel values inside the box is taken as the noise standard deviation, but by using the “-N” option, the standard deviation can be calculated by k-sigma clipping. An alternative with this model is to use a pre-defined root mean square map (see option “-R”).
6. Non-stationary multiplicative noise
7. Undefined stationary noise
8. Undefined noise
9. Stationary correlated noise
The noise map must be given using the “-R” option. Option “-E” allows us to fix the confidence interval of the detection (this replaces the “-s” option used by other noise models).
10. Poisson noise with few events
Option “-E” allows us to fix the confidence interval of the detection (this replaces the “-s” option used by other noise models). With this noise model, only the à trous wavelet transform algorithm can be used.

Default is Gaussian noise.

- **[-n number_of_scales]**
- **[-s NSigma]**
Only used with *type_of_filtering* in [1,2,3,4,5,17,18].
- **[-e Epsilon]**
Convergence parameter (used only by the filtering method 2). Default is 1e-5 in the case of Poisson noise with few events, and 1e-3 in other cases.
- **[-i number_of_iterations]**
Maximum number of iterations (used only by iterative filtering methods). Default is 10.
- **[-w support_file_name]**
If this option is set, two files are created. The first one (“.mr”) contains the multiresolution support, and the second one contains an image which is created from the multiresolution support. Default is not to do this. The suffix must not be specified.
- **[-k]**
Suppress isolated pixels in the multiresolution support. Default is no suppression.
- **[-K]**
Suppress the last scale. The last scale is not used during the restoration process. This leads to a filtered image which does not contain a background, or extended structures (depending on the number of scales used). Default is no suppression.
- **[-p]**
Detect only positive structures. Wavelet coefficients can be either positive or negative. Using this option, only positive coefficients are considered as significant. Default is to take both positive and negative coefficients.

- **[-E Epsilon]**

Precision for computing thresholds, used only in the case of Poisson noise with few events. Default is 1e-3, which is equivalent to a 3.1 sigma detection in the Gaussian case.

- **[-S SizeBlock]**

Size of the blocks used for local variance estimation. Default is 7.

- **[-N NiterSigmaClip]**

Iteration number used for local variance estimation. Default is 1.

- **[-F first_detection_scale]**

If this option is set, all wavelet coefficients detected at scales lower than *first_detection_scale* are considered as significant.

- **[-R RMS_Map_File_Name]**

Root mean square map. If this option is set, the noise model is automatically fixed to: “Non-stationary additive noise”, and the RMS image is used as the local standard deviations image.

- **[-P]**

By default, a positivity constraint is applied to the solution. This means that the solution is forced to be positive everywhere. By using this option, this constraint is suppressed.

- **[-b]**

Add the maximum level constraint (maximum = 255). For one-byte images (GIF, JPEG, etc.), this constraint limits the range values between 0 and 255.

- **[-W WindowSize]**

Window size for median and average filtering. Default is 5. This option is only valid if *type_of_filtering* is equal to 4 or 5.

- **[-v]**

Verbose.

For median, average, and B-spline filtering, all options (except “-W”) have no effect.

Examples:

- `mr_filter image_in.d ima_out.d`
Filters an image by multiresolution thresholding, assuming Gaussian noise (its standard deviation is automatically estimated).
- `mr_filter -t24 image_in.d ima_out.d`
Hard thresholding using the undecimated bi-orthogonal wavelet transform.
- `mr_filter -t24 -u1 image_in.d ima_out.d`
Hard thresholding using the partially undecimated bi-orthogonal wavelet transform.
- `mr_filter -f 3 image_in.d ima_out.d`
Iterative filtering using the à trous algorithm.
- `mr_filter -m 2 -s 4 -p image_in.d ima_out.d`
Same as before, but considering Poisson noise, and the thresholding is done at 4 sigma (instead of 3). All negative coefficients are thresholded.
- `mr_filter -K -n 3 -s 7 ngc2997.fits stars.fits`
Subtracts from the input image (see Figure 3.3) the noise and the last scale. As only three scales were used, all large structures disappear (see Figure 4.1 top). The difference (see Figure 4.1 bottom) between the input and the output images shows the galaxy, in which all high small-scale structures have been removed.

Filtering Strategy:

The *mr_filter* programs allows the user to perform a filtering by many methods, and using many different transforms. Depending on the application, the optimal filtering method may change. For astronomical applications, for instance, methods 3 and 4 lead to very good results, especially for the photometry of the objects. However, some general aspects can be noted:

- Directional transforms (i.e., option “-t” in [14,21,24]) are well adapted to images which contain edges, lines, etc., while isotropic transforms (à trous algorithm, etc.) are better adapted for images which contain isotropic features.
- Redundant transforms do a better job than non-redundant transforms, because they respect the translation invariance property. So the undecimated bi-orthogonal wavelet transform (i.e. option -t 24) for images containing edges, and the à trous algorithm (i.e. option -t 2) for images with isotropic features, should always be preferred to decimated transforms.
- If the computation time is important, or if the image is very large, a good trade-off between quality and memory or computation time is to let the first scale remain undecimated, while decimating the others. This is done by using the “-t 24 -u 1” option for the bi-orthogonal WT, or the “-t 19” option for the à trous algorithm.
- The number of scales (i.e., option -n *ScaleNumber*) which can be used depends on the image size. The larger the image, the larger the number of scales. The user must keep in mind that increasing by 1 the number of scales corresponds to adding a new scale in which the number of independent pixels is divided by two. In theory, the number of scales could be $n = \log_2(N_s) + 1$, where N_s is the image size in its smallest direction, but in practice, it is preferable to use a lower value ($\log_2(N_s) - 2$ or $\log_2(N_s) - 3$).

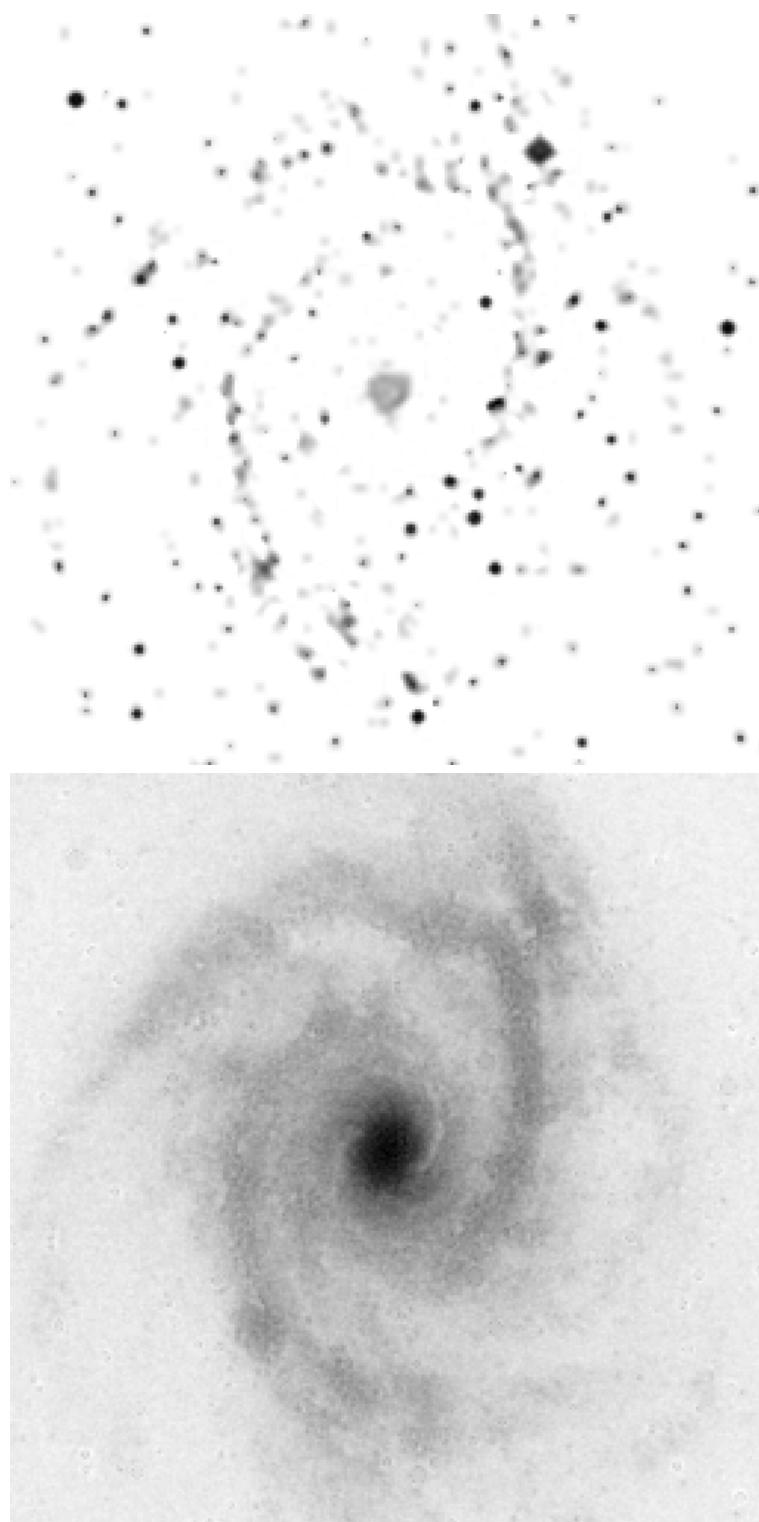


Figure 4.1: Top, output image produced by *mr_filter*. The noise and the large structures of the galaxy have been removed. Bottom, difference between NGC2997 and the output image.

4.3 Image with Speckle Noise

4.3.1 Speckle noise statistics

Speckle occurs in all types of coherent imagery such as synthetic aperture radar (SAR) imagery, acoustic imagery and laser illuminated imagery [48]. When an object is illuminated by a coherent source of radiation and the object has a surface that is roughly the order of the wavelength of the incident radiation, the wave scattered from the surface consists of contributions from many independent scattering areas. The signal is assumed to be a complex Gaussian distributed noise of zero mean and standard deviation σ . The value of σ is a function of the backscattering coefficient σ_0 of the observed surface. The modulus ρ of the signal gives an image (see Figure 4.3) from which one can derive some properties about the object surface by measuring the local value of σ . The probability density function (PDF) of the modulus of a homogeneous scene is a Rayleigh distribution:

$$p(\rho) = \frac{\rho}{\sigma^2} e^{-\frac{\rho^2}{2\sigma^2}} \quad (4.12)$$

of mean M_ρ and standard deviation σ_ρ :

$$M_\rho = \sqrt{\frac{\pi}{2}\sigma} \quad \sigma_\rho = \sqrt{\frac{4-\pi}{2}\sigma} \quad (4.13)$$

The ratio σ_ρ/M_ρ is a constant of value $\sqrt{\frac{4-\pi}{\pi}}$. This means that the speckle is multiplicative noise. For this reason some filtering methods, called homomorphic techniques [74], use a logarithmic transform of the modulus to convert the multiplicative noise into additive noise. The PDF of the modulus of log-transformed speckle noise is:

$$p(\ell) = \frac{e^{2\ell}}{\sigma^2} e^{-\frac{e^{2\ell}}{2\sigma^2}} \quad (4.14)$$

The mean M_ℓ and the standard deviation σ_ℓ are:

$$M_\ell = 0.058 + \log(\sigma) \quad \sigma_\ell = \sqrt{\frac{\pi^2}{24}} = 0.641 \quad (4.15)$$

When using the logarithm, the standard deviation is independent of the local value of the signal, but the estimation of σ_0 from the filtered image has a bias which is not minimum according to the Cramer-Rao theorem. The minimum variance bound estimator of a Rayleigh distribution is the energy (i.e. the square of the modulus $I = \rho^2$) which is known to have an exponential (i.e. Laplace) distribution of parameter $a = 2\sigma^2$ [90]:

$$p(I) = \frac{1}{a} e^{-\frac{I}{a}} \quad (4.16)$$

With this transformation the noise is still multiplicative, but it has been shown [14] that the ratio R between the energy of the image to be filtered and a spatially Laplace-distributed image of local mean value $a(k_x, k_y)$ has a Laplace PDF of mean 1. In that case, the filtering algorithm consists of finding the local value of a which is iteratively obtained by rejecting the non-significant wavelet coefficients found in the image ratio R according to a simulated Laplace noise of mean 1.

4.3.2 Speckle filtering: mr_rfilter

General description

The filtering algorithm can be applied on the modulus, the logarithm of the modulus or the square of the modulus of the image. Visually the results are quite similar, but the use of a quadratic transform allows more accurate radiometric estimation from the filtered image. When using the logarithmic transform the general structure of the algorithm is the following (for more details see [14]):

1. Simulate a log-Rayleigh noise image model.
2. Compute the wavelet transform of this simulated image by using the à trous algorithm.
3. According to the chosen statistical decision level ϵ , compute the corresponding positive and negative thresholds for each scale as shown in Figure 4.2.
4. Compute the logarithm of the original image.
5. Threshold the wavelet transform of the log-transformed original image according to the thresholds computed with the noise model.
6. Reconstruct the image from the thresholded wavelet transform.

If the reconstructed image is correct (i.e. the reconstructed image is the filtered image) the residual between this image and the original one must contain only noise. If not, we have to iterate from step 5 to step 6 to extract the significant structures from the residual and refine the filtered image.

For multiplicative noise (i.e. Rayleigh or Laplace noise) we have to test the significant part of the ratio R between the original transformed image and a reference image. This reference image is the filtered image if the wavelet transform of the ratio contains no significant wavelet coefficients. The algorithm starts with any approximation of the image to be filtered as the reference image. Generally, the last smooth image of the wavelet transform is used to initialize the first reference image. The reference image is iteratively refined by introducing the significant part detected from the ratio R until this ratio has a Laplace PDF of parameter 1. In the case of Rayleigh distributed noise (i.e. no transformation) the test is carried out on the variation of the standard deviation of the residual between two successive iterations. For many reasons (precision of the convergence test, estimation of the scattering parameter σ_0 from the filtered image) the Laplace noise model (i.e. square-transformed Rayleigh noise) gives the best results.

Thresholding function

The thresholded wavelet coefficients $\bar{w}(i, k_x, k_y)$ are computed from the original wavelet coefficients $w(i, k_x, k_y)$ by applying the following equation:

$$\bar{w}(i, k_x, k_y) = S(w(i, k_x, k_y))w(i, k_x, k_y) \quad (4.17)$$

where $S(w)$ is the thresholding function. For hard thresholding (see Figure 4.2-a), $S(w)$ is defined by:

$$S(w) = \begin{cases} 0 & \text{if } w \in [x_n, x_p] \\ 1 & \text{elsewhere} \end{cases} \quad (4.18)$$

For soft thresholding (see Figure 4.2-b) the thresholding function is:

$$S(w) = \begin{cases} 0 & \text{if } w \in [xn_1, xp_1] \\ \frac{w - xp_1}{xp_2 - xp_1} & \text{if } w \in [xp_1, xp_2] \\ \frac{w - xn_1}{xn_2 - xn_1} & \text{if } w \in [xn_2, xn_1] \\ 1 & \text{elsewhere} \end{cases} \quad (4.19)$$

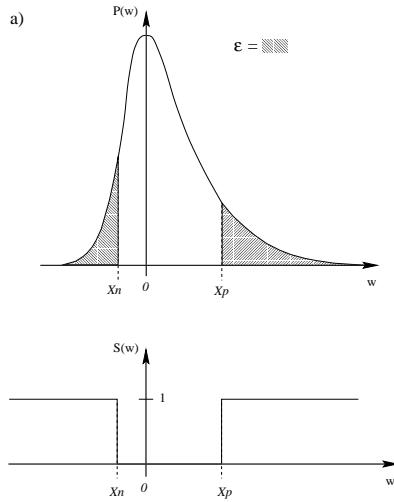


Figure 4.2: Threshold determination and thresholding functions; (a) hard thresholding, (b) soft thresholding.

Hierarchical thresholding

With the hierarchical thresholding, the thresholded wavelet coefficients at a given scale i are such that:

$$\bar{w}(i, k_x, k_y) = S(w(i, k_x, k_y))S(w(i + 1, k_x, k_y))w(i, k_x, k_y) \quad (4.20)$$

This kind of thresholding is generally used to reduce the number of false detections on the first scale.

mr_rfilter

Program *mr_rfilter* filters an image that contains speckle noise by using a multiscale à trous algorithm.

USAGE: mr_rfilter options image_in image_out

where options are :

- [-t type_of_noise]

0. Rayleigh noise:

The filtering algorithm is applied to the modulus of the image.

1. Log Rayleigh:

The filtering algorithm is applied after a logarithmic transform of the image.

2. Laplace:

The filtering algorithm is applied after a quadratic transform of the image.

Default is Laplace noise model.

- **[-n number_of_scales]**

- **[-i number_of_iterations]**

Maximum number of iterations. Default is 10.

- **[-e Epsilon]**

Convergence parameter. Default is 0.001.

- **[-E decision_level]**

Statistical decision level ϵ used for each scale. If 0, the value of ϵ is set interactively for each of the first four scales. If the maximum number of scales (scale_max) set by the parameter -n is greater than 4, the statistical decision levels from scale 5 to scale_max is set to the value of the statistical decision level used for scale 4.

Default is 0.001.

- **[-N number_of_images]**

If the input image is a multi-look image (i.e. the input image is the average of N single-look images).

Default is 1.

- **[-r Ratio]**

ratio=Epsilon2/Epsilon1 used for soft thresholding. If set, a hierarchical filtering is used for the first scale. The value of Epsilon1 is set with the -E option. If ratio=1, a standard hard thresholding is used for all scales. Default is 10.

Examples

- `mr_rfilter -t1 -E.001 -i5 image.fits result`

Filter the image *image.fits* by considering an additive log-Rayleigh noise model (logarithmic transformation) with 5 iterations and a statistical decision level $\epsilon = 0.001$ for all scales.

- `mr_rfilter -t1 -E.001 -e0.01 image.fits result`

The same as above, but the iterations are stopped when the difference between the standard deviation of the residual and the theoretical value (i.e. 0.64 for the log-Rayleigh distribution) is less than 0.01.

- `mr_rfilter -t2 -E0 -i5 -n8 image.fits result`

Filter the image *image.fits* by considering a multiplicative Laplace noise model (quadratic transformation) with 5 iterations and 8 scales. The program will ask the user to enter the statistical decision level *Epsilon*[*i*] at each scale. Typical values are:

Epsilon[0] = 0.0001

Epsilon[1] = 0.001

Epsilon[2] = 0.01

Epsilon[3] = 0.01

Epsilon[4] = 0.1

The values from *Epsilon*[5] to *Epsilon*[8] are automatically set to 0.1.

- `mr_rfilter -t3 -E0 -r10 -i5 -n8 image.fits result`

The same as above but a soft thresholding is used with a ratio=10, and a Rayleigh noise model.



Figure 4.3: Top, raw synthetic aperture radar (SAR) image from the ERS1 satellite (CNES image). Bottom, image produced by *mr_rfilter -t2 -n6 -i5 -E0 -r10*, with $\text{Epsilon}[1]=\text{Epsilon}[2]=0.0001$ and $\text{Epsilon}[3]$ to $\text{Epsilon}[6]=0.001$.

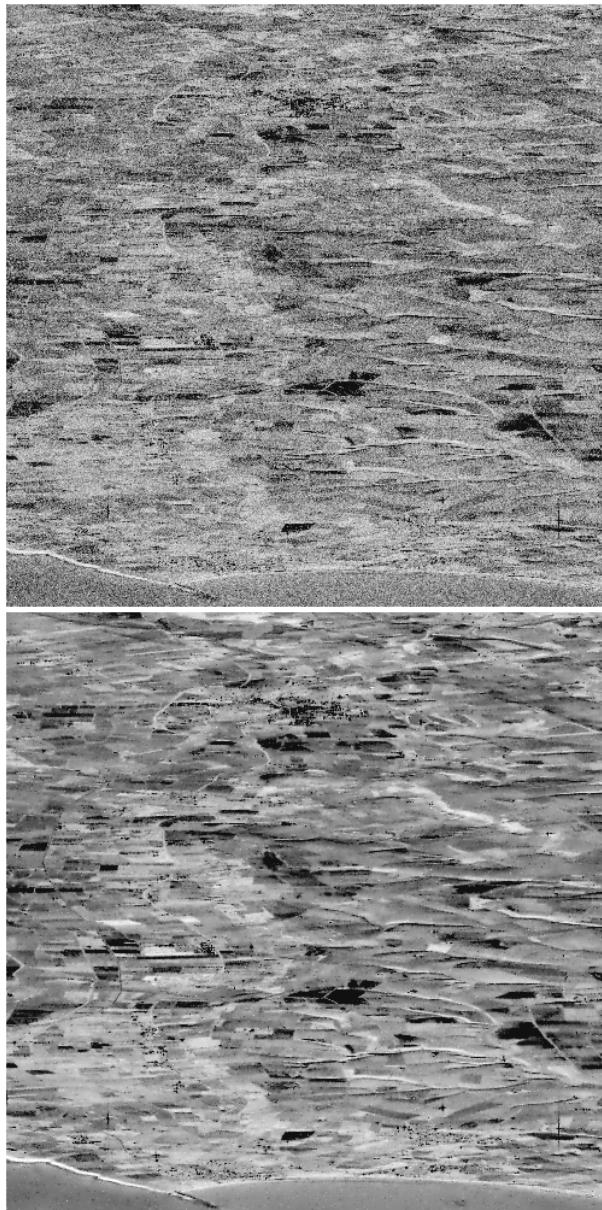


Figure 4.4: Top, multi-look image generated by averaging 7 raw SAR images. Bottom, output image produced by `mr_rfilter -t2 -N7 -n6 -i5 -E0 -r10`, with Epsilon[1]=Epsilon[2]=0.0001 and Epsilon[3] to Epsilon[6]=0.001.

Epsilon	NSigma
$1e^{-1}$	1.28160
$1e^{-2}$	2.32630
$1e^{-3}$	3.09020
$1e^{-4}$	3.71900
$1e^{-5}$	4.26490
$1e^{-6}$	4.75340
$1e^{-7}$	5.19930
$1e^{-8}$	5.61200
$1e^{-9}$	5.99780

Table 4.1: Correspondence between confidence level and Gaussian detection level.

4.4 Sparse Images: à trous Algorithm

In this section we are concerned with the restoration of images with few photons or counts by the à trous algorithm.

4.4.1 Initialization: mr_abaque

Program *mr_abaque* precomputes a table which is used by *mr_psupport* and *mr_pfilter*. This table contains the threshold levels for a wavelet coefficient with a confidence interval ϵ . These levels are a function of the number of events used for the calculation of the wavelet coefficient. The table is saved by default in the FITS table format. The levels are calculated from the autoconvolution of the histogram of the wavelet function. The algorithm first calculates the histogram, then performs the autoconvolution, derives the probability distribution of a wavelet coefficient (depending on the number of events), the distribution function, and finally derives the thresholds for a confidence interval equal to *Epsilon*.

USAGE: mr_abaque option file_out

where options are:

- **[-e Epsilon]**

Epsilon = confidence level. Default is 1e-3. The correspondence between the confidence level and $N\sigma$ detection in the Gaussian case is as shown in Table 4.1.

- **[-n Number]**

Number = Number of events in the data, given as an integer power of 2. Default is 25 (i.e. 2^{25}).

- **[-w]**

Write the following files:

- Aba_histo.fits: contains all histograms
 $h(3*i)$ = histogram values
 $h(3*i+1)$ = reduced coordinates
 $h(3*i+2)$ = histogram values for reduced coordinates
- Aba_bspline.d: contains the 1D B-spline used
- Aba_wavelet.d: contains the 2D wavelet used

- **[-d]**

Use all default parameters.

Example:

- **mr_abaque -d**
Creates the table (with filename “Abaque.fits”) with all default options. The table is saved in an image format with 26 lines and 2 columns (by default). *Abaque*(*i*, 0) corresponds to the negative detection level for a wavelet coefficient calculated from 2^i events (photons), and *Abaque*(*i*, 1) corresponds to the positive threshold.
- **mr_abaque -e 1e-04**
Creates the table for a detection with a confidence interval of 1e-04.

4.4.2 Support creation: mr_psupport

Program *mr_psupport* applies a wavelet transform using the à trous algorithm to data, assuming that the noise follows a Poisson distribution, and in the case where we have only few events per pixel [153, 197, 172, 174, 137]. Data can either be given by an ASCII table of coordinates in real values, or by an image. In the first case, the data must begin with a line containing the size (number of rows and number of columns separated by a space) of the image in which the events can be inserted, and all other rows must contain one position (real coordinates separated by a space). The wavelet transform is then thresholded and stored in the output multiresolution file (“.mr”). If the option “-s” or “-t” is set, an analysis of the detected structures is performed. Results are stored in a file.

USAGE: mr_psupport option mr_file.out

where options are:

- **[-a ascii_file_in]**
Read the input data from an ASCII table. An example of a table with three events in an image of size 10×10 is:
10 10
2.5 3.8
4. 4.
0.3 9.2
The first coordinates must all have values between 0 and the number of rows minus 0.5, and the second coordinate must have values between 0 and the number of columns minus 0.5.
- **[-I image_file_in]**
Read the input data from an image.
- **[-F first_detection_scale]**
First scale used for the detection. Default is 1.
- **[-e minimum_of_events]**
Minimum number of events for a detection. Default is 4. If a wavelet coefficient has been calculated from fewer events than this minimum value, it is not considered as significant.
- **[-w]**
write the following file
xx_Wavelet.mr: contains the wavelet transform of the image.
- **[-s SignifStructureAnalysis_FileName]**
Write in xx_Segment.mr the segmented scales.
Analyze the detected wavelet coefficients, and write in the file:
 - Number of detected structures per scale

- Percentage of significant wavelet coefficients
 - Mean deviation of shape from sphericity
 - For each detected structure, its surface area, its perimeter, and
 - its deviation of shape from sphericity,
 - its angle, its elongation in both axis directions.
- **[-t SignifStructureAnalysis_FileName]**
Same as -s option, but results are stored in an ASCII table format. The table contains: scale number, structure number, Max_x, Max_y, Surface, Perimeter, Morpho, Angle, Sigma_X, Sigma_Y.
 - **[-p]**
Detect only positive structure.
 - **[-q abaque_file]**
Default is Abaque.d
 - **[-n number_of_scales]**
Number of scales used in the multiresolution transform. Default is 6.

Examples:

- mr_psupport -a tabevent.ascii support.mr
Read the table, create the associated image, apply the wavelet transform, and threshold the non-significant wavelet coefficients.
- mr_psupport -a tabevent.ascii -s analysis.txt support.mr
Same as before, but also apply a segmentation to each scale. Results of the segmentation are stored in “xx_Segment.mr”, and each region is analyzed. Results of this analysis are saved in “analysis.txt”.

4.4.3 Filtering: mr_pfilter

Program *mr_pfilter* filters an image (as does *mr_filter*) assuming that the noise follows a Poisson distribution and in the case where we have only few events per pixel. Data can either be given by an image or by an ASCII table of coordinates in real values. As for *mr_psupport*, the pre-computed table must exist. By default, the program searches for a file of name “Abaque.fits”. If the “-c” option is used, the user gives a second input image file name. Then the multiresolution support is estimated from the first image (given by option “-a” or “-I”), and the second image is filtered using the multiresolution support of the first.

USAGE: mr_pfilter option image_out

where options are:

- **[-a ascii_file_in]**
See section 4.4.2.
- **[-I image_file_in]**
- **[-c ImageFileName]**
Image to be filtered using the multiresolution support derived from the image given by option “-a” or “-I”. By default, the image to filter is the same as the image used for the multiresolution support calculation.

- **[-w]**
write the following files
xx_Wavelet.mr: contains the wavelet transform of the image.
xx_Support.mr: contains the thresholded wavelet transform.
- **[-p]**
Detect only positive structure.
- **[-F first_detection_scale]**
First scale used for the detection. Default is 1.
- **[-q abaque_file]**
Default is Abaque.d
- **[-n number_of_scales]**
Number of scales used in the multiresolution transform. Default is 6.
- **[-e minimum_of_events]**
Minimum number of events for a detection. Default is 4.
- **[-f type_of_filtering]**
 1. Iterative multiresolution thresholding
 2. Adjoint operator applied to the multiresolution support
 3. Multiresolution Hard K-Sigma Thresholding

Default is Iterative multiresolution thresholding.
- **[-l]**
Dilate the support.
- **[-k]**
Suppress isolated pixels in the support.
- **[-K]**
Suppress the last scale.
- **[-i number_of_iterations]**
Maximum number of iterations. Default is 50.

The a and I options cannot be used together, and one of them must be set.

Examples:

- mr_abaque -e 1e-4
Creates the table (file=Abaque.fits) for a confidence interval of 1e-4.
- mr_pfilter -a tabevent.ascii filter_imag.fits
Read the table, create the associated image, and apply the filtering using the multiresolution support.
- mr_pfilter -I inpu_image.fits -f 2 -i 10 -F 3 -p output_image.fits
Filtering using the second method, with 10 iterations, without taking account of negative wavelet coefficients, and starting the detection at the third scale.

4.5 Images with Poisson Noise: Haar Transform

In this section, we are concerned with restoration of Images with Poisson noise by the Haar Transform.

4.5.1 Introduction

Several authors [104, 105, 203, 148, 18] have recently suggested independently that the Haar wavelet transform was very well suited to treat data with Poisson noise. Indeed, since a Haar wavelet coefficient is just the difference between two random variables following a Poisson distribution, it is easier to derive mathematical tools to remove the noise than with other wavelet methods.

The means used to filter the noise is however different following different authors. In [148], a kind of Wiener filter has been implemented. Timmermann and Nowak [203] have used a Bayesian approach with an a priori model on the original signal. Kolaczyk [104] proposed to use the Haar transform for gamma-ray burst detection in a one-dimensional signal, and has extended his method to images [105]. In his method, the thresholds are calculated from the PDF of the wavelet coefficients. A similar approach has been developed by Jammal and Bijaoui [18] for medical image filtering.

As far back as 1910, Haar described the following function as providing an orthonormal basis. The analyzing wavelet of a continuous variable is a step function.

$$\begin{aligned}\psi(x) &= 1 && \text{if } 0 \leq x < \frac{1}{2} \\ \psi(x) &= -1 && \text{if } \frac{1}{2} \leq x < 1 \\ \psi(x) &= 0 && \text{otherwise}\end{aligned}$$

The Haar wavelet constitutes an orthonormal basis. Two Haar wavelets of the same scale (i.e. value of m) never overlap, so we have scalar product $\langle \psi_{m,n}, \psi_{m,n'} \rangle = \delta_{n,n'}$ (the Kronecker delta, equal to 1 when the subscripts are the same, otherwise equal to zero). Overlapping supports are possible if the two wavelets have different scales, e.g. $\psi_{1,1}$ and $\psi_{3,0}$ (see [52], pp. 10–11). However, if $m < m'$, then the support of $\psi_{m,n}$ lies wholly in the region where $\psi_{m',n'}$ is constant. It follows that $\langle \psi_{m,n}, \psi_{m',n'} \rangle$ is proportional to the integral of $\psi_{m,n}$, i.e. zero.

4.5.2 Poisson noise and Haar wavelet coefficients

Thresholding assuming a uniform background

Assuming a constant background with a background rate λ , Kolaczyk and Dixon [105] proposed to use the normalized Haar transform (L2-normalization) with the following threshold, corresponding a false detection rate of α :

$$t_j = 2^{-(j+1)}[z_{\alpha/2}^2 + \sqrt{z_{\alpha/2}^4 + 4\lambda_j z_{\alpha/2}^2}] \quad (4.21)$$

where j is the scale level ($j = 1..J$, J being the number of scales), $\lambda_j = 2^{2j}\lambda$ is the background rate over $n_j = 2^{2j}$ pixels, and $z\alpha/2$ is the point under the Gaussian density function for which there falls $\alpha/2$ mass in the tails beyond each $z\alpha/2$. An upper bound limit for the threshold limit, valid also with other filters, is [105]:

$$t_j = 2^{-(j+1)} \log(n_j) + \sqrt{\log^2(n_j) + 2\lambda_j \log(n_j)} \quad (4.22)$$

This formula results from substituting $z = \sqrt{2\log(n_j)}$ in the previous equation.

Jammal and Bijaoui [18] have calculated the probability density function of an unnormalized wavelet coefficient which is given by

$$p(w_j = \nu) = e^{-2^{2j}\lambda} I_\nu(2^{2j}\lambda) \quad (4.23)$$

where $I_\nu(x)$ is the modified Bessel function of integer order ν . For a given false detection rate α , the threshold t_j can be derived from this PDF.

Thresholding with non-uniform background

In many cases, the background cannot be considered to be constant, and an estimation of $\lambda_{j,k,l}$ (the background rate at scale j and position k, l) is necessary. Several approaches can be used to take into account the background variation:

- Image model: if a model image M can be provided by the user, then $\lambda_{j,k,l}$ is easily obtained by integrating M over the correct surface area.
- Lower resolution: the filtering must be started from the coarser scale S_N . The solution is refined scale by scale in order to obtain $S_{N-1}, S_{N-2}, \dots, S_1, S_0$. The $\lambda_{j,k,l}$ values are obtained from $\lambda_{j,k,l}$ ($\lambda_{j,k,l} = \frac{\lambda_{j,k/2,l/2}}{4}$).
- iterative procedure: a filtering can first be performed assuming a constant background (with rate equal to the mean of the image), and the filtered image can be used as a model image. This process can be repeated several times until convergence.

Reconstruction

The Haar transform is known to produce block-artifacts. Two approaches may be used to resolve this problem

- Cycle-spinning method [43].
- Iterative constraint reconstruction [23].

The cycle-spinning method precedes the restoration algorithm (Haar transform + thresholding + reconstruction) at every version of the original image data obtainable by combinations of left-right and upwards-downwards translations. The final image is simply the average of all images resulting from each set of translation sequence.

The iterative constraint reconstruction consists of considering the reconstruction as an inverse problem, where the solution must respect some constraints. Constraints on the solution are:

- The positivity.
- The range of variation of the Haar coefficients.
- The smoothness at all resolution levels.

If $w_{j,k,l}$ and $s_{j,k,l}$ are respectively a Haar coefficient at the scale j and at position k, l of the data and the solution, then $s_{j,k,l}$ must verify:

$$\begin{cases} s_{j,k,l} \in [-t_j, 0] & \text{if } w_{j,k,l} \in [-t_j, 0] \\ s_{j,k,l} \in [0, t_j] & \text{if } w_{j,k,l} \in [0, t_j] \\ s_{j,k,l} \in [w_{j,k,l} - t_j/2, w_{j,k,l} + t_j/2] & \text{if } |w_{j,k,l}| > t_j \end{cases} \quad (4.24)$$

The smoothness constraint consists of minimizing the gradient of the solution S_j at the scale j in both vertical and horizontal directions:

$$C(S) = \| D_x S(x, y) \|^2 + \| D_y S(x, y) \|^2 \quad (4.25)$$

where D_x and D_y are the gradient operators in both directions. A full description of the algorithm can be found in [23].

MMI model

The Multiscale Multiplicative Innovations (MMI) model was proposed in [203], and introduces a prior model f_Λ , which is a beta-mixture density functions of the form:

$$f(\delta) = \sum_{i=1}^M p_i \frac{(1-\delta^2)^{s_i-1}}{B(s_i, s_i) 2^{2s_i-1}} \quad (4.26)$$

for $-1 \leq \delta \leq 1$, where B is the Euler beta function, $-1 \leq p_i \leq 1$ is the weight of the i -th beta density $\frac{(1-\delta^2)^{s_i-1}}{B(s_i, s_i) 2^{2s_i-1}}$ with parameter $s_i \geq 1$, and $\sum_{i=1}^M p_i = 1$. The final algorithm consists of multiplying each wavelet coefficient $w_{j,k,l}$ by a term which is derived from the model.

PRESS-optimal filter

The PRESS-optimal filter shrinks the noisy wavelet coefficient toward zero according to the estimated signal to signal-plus-noise ratio. The PRESS-optimal filter is given by:

$$h_{j,k,l} = \frac{\tilde{w}_{j,k,l}^2}{\tilde{w}_{j,k,l}^2 + \sigma_{j,k,l}^2} \quad (4.27)$$

where $\sigma_{j,k,l}^2$ and $\tilde{w}_{j,k,l}$ are the noise and signal power. The noise power is proportional to an estimate of the local intensity of the image falling under the support of the Haar coefficient $w_{j,k,l}$. The signal power can be estimated from a model, or directly from the data by:

$$\tilde{w}_{j,k,l}^2 = w_{j,k,l}^2 - \sigma_{j,k,l}^2 \quad (4.28)$$

4.5.3 Filtering: mr_hfilter

Program *mr_hfilter* filters an image using the undecimated Haar wavelet transform assuming that the noise follows a Poisson distribution.

USAGE: mr_hfilter option image_in image_out

where options are:

- **[-n number_of_scales]**

Number of scales used in the multiresolution transform. By default, the number of scales is calculated from the image size by the relation:

$$Nscale = \log_2(MIN(Nl, Nc)) - 2 \quad (4.29)$$

- **[-s Nsigma]**

False detection rate. The false detection rate for a detection is given

$$\epsilon = erfc(NSigma/\sqrt{2}) \quad (4.30)$$

Nsigma parameter allows us to express the false detection rate as if it were Gaussian noise. Default is 3.

- **[-F first_detection_scale]**

First scale used for the detection. Default is 1.

- **[-M BackgroundModel]**

1. Flat background
2. Multiresolution background estimation
3. Background image model
4. Iterative background estimation

Default is 2.

- **[-T ThresholdingMethod]**

1. Kolaczyk-Dixon threshold.
2. Kolaczyk-Dixon upper bound threshold.
3. Jammal-Bijaoui threshold.
Using this method, the ϵ false detection rate must be in the interval in $[10^{-6}, 10^{-2}]$.
4. PRESS-optimal filter

Default is 1.

- **[-h Haar_FilterBank]**

1. Haar filter
2. Biorthogonal 2/6 Haar filters
3. Biorthogonal 2/10 Haar filters

Default is Biorthogonal 2/6 Haar filters.

- **[-B FileName]**

Background Image Model File Name. Only valid if the *BackgroundModel* is set to 3 (Background image model).

- **[-I NiterBgr]**

Number of iteration for the iterative background estimation. Only valid if the *BackgroundModel* is set to 4 (Iterative background estimation).

Default is 4.

- **[-S]**

Apply a soft thresholding instead of a hard-thresholding.

- **[-L LambdaValue]**

Lambda Value. Only valid if the *BackgroundModel* is set to 1 (Flat background). Default value is set to mean of the input image.

- **[-P PsfInFile]**

Input Point Spread Function. If set, a deconvolution is performed.

- **[-G RegulParam]**

Regularization parameter for the deconvolution.

When using PRESS-optimal filter, options “s,F,M,B,I,S,L” have no effect.

Examples:

- `mr_hfilter imag.fits filter_imag.fits`
Image filtering using all default options.
- `mr_hfilter -T 3 image.fits output_image.fits`
Filtering using the Jammal-Bijaoui Threshold.
- `mr_hfilter -T 3 -M 4 image.fits output_image.fits`
Filtering using the Jammal-Bijaoui Threshold, and an iterative background estimation.

Experiments

From our experiments on astronomical images, we found that:

- the à trous algorithm is significantly better than any of the Haar-based methods.
- PRESS-optimal filter is not optimal at all!
- The lower resolution furnishes a good estimation of the background. Iterating does not improve significantly the results.
- The Jammal-Bijaoui threshold is a little better than the Kolaczyk one for compact source detection, and is equivalent for more extended sources. But the Kolaczyk threshold requires less computation time and is easier to implement.

Such study shows clearly that the Haar transform is less efficient for restoring X-ray astronomical images than the à trous algorithm. But its simplicity, and the time computation, may be attractive in many cases.

4.6 Image deconvolution

4.6.1 Introduction

See chapter 14.

4.6.2 Standard methods: im_deconv

Program *im_deconv* deconvolves an image, assuming a space invariant point spread function (PSF). The PSF is automatically centered in the image, and its flux is renormalized to unity. The output resolution to achieve can be limited using the concept of ICF (Intrinsic Correlation Function), either by selecting a Gaussian ICF function (and fixing its full-width at half maximum with the “-f” option), or by giving an image ICF file name (option “-I”). The main methods are iterative, and some of them can be optimized using the “-O” option, which activates the automatic convergence parameter calculation. For iterative methods, there is a stopping criterion, which can be modified using the “-e” option. A fixed number of iterations can be specified by the options “-e 0 -i NIter”. For MEM methods, the regularization parameter “-G” controls the smoothness of the solution. Increasing “-G” produces a smoother image. In order to avoid divergence for large values, it is recommended to decrease at the same time the convergence parameter by use of the “-C” option.

USAGE: `im_deconv option image_in psf_in image_out`

where options are

- **[-d type_of_deconvolution]**
 1. Deconvolution by Van Cittert’s algorithm
Standard Van Cittert algorithm
 2. Deconvolution by gradient algorithm
Minimization of a functional by the fixed step gradient method.
 3. Deconvolution by division in Fourier space
Divides the Fourier transform of the image by the Fourier transform of the PSF. If the “-f” option is set, the solution is convolved with a Gaussian.
 4. Deconvolution by Richardson-Lucy algorithm
Standard Richardson-Lucy (Shepp-Vardi, maximum likelihood expectation-maximization, EM) algorithm. This method fails when the input image contains negative values.
 5. Deconvolution by CLEAN algorithm
Standard CLEAN algorithm. The “-G” option fixes the loop gain factor, and the “-f” option fixes the size of the clean beam (full-width at half-maximum). By default, the residual map is not added to the clean map, and the user can get it using the “-r resi_filename” command. Criteria used for stopping the CLEAN loop are
 - (a) the maximum of the residual map is lower than the average value plus $NSigma \times Noise$. $NSigma$ and $Noise$ can be modified using “-s” and “-g” options,
 - (b) the maximum number of iterations is reached.
 6. Deconvolution by the MEM method (Frieden entropy)
 7. Deconvolution by the MEM method (Gull entropy)
 8. Deconvolution using Tikhonov regularization
 9. Deconvolution using MAP method
 10. Deconvolution by the Gradient algorithm + Markov Random Field Regularization
 11. Deconvolution by Lucy’s algorithm + Markov Random Field Regularization

Default is deconvolution by the gradient algorithm.

- **[-i number_of_iterations]**
Maximum number of iterations. Default is 1e6 for CLEAN and 500 for other deconvolution methods.
- **[-P]**
Suppress the positivity constraint.
- **[-G RegulParam]**
Regularization parameter. Only used with methods 5 to 8, and 10, 11. Default is 0.1.
- **[-C ConvergParam]**
Convergence parameter.
Default is 1.
- **[-f ICF_Fwhm]**
Fwhm = full-width at half-maximum.
- **[-I ICF_FileName]**
Intrinsic correlation function file.
- **[-F First_Guess]**
Input solution file name. This image is the starting point of the iterative methods. This option has no effect with deconvolution methods 3 and 5.
- **[-O]**
Optimization. The iteration parameter is calculated at each iteration in order to optimize the minimization of the functional. Only used with deconvolution methods 2, 4, and 8. For method 4, the optimization is done under the assumption of Poisson noise, and for methods 2 and 8 under the assumption of Gaussian noise.
- **[-M Model_Image]**
Input model file name for MEM method. Only used with method 7.
- **[-r residual_file_name]**
If this option is set, the residual is written to the file of name *residual_file_name*. By default, the residual is not written. The residual is equal to the difference between the input image and the solution convolved with the PSF.
- **[-e epsilon]**
Convergence parameter. Default is $1e - 3$.
- **[-S]**
Do not shift automatically the maximum of the PSF to the center.
- **[-v]**
Verbose.

Examples:

- `im_deconv -d 3 image_in.d psf_in.d ima_out.d`
Deconvolves an image by the regularized Richardson-Lucy algorithm, assuming Gaussian noise (its standard deviation is automatically estimated).
- `im_deconv -d 3 -f 2. image_in.d psf_in.d ima_out.d`
Ditto, but limit the resolution to achieve. The solution will be the convolution product of a hidden solution by a Gaussian of full-width at half maximum equal to 2.
- `im_deconv -F InputSol.d image_in.d psf_in.d ima_out.d`
Deconvolves an image by the gradient method. The program uses the image given by option “-F” as entry point to the iteration.

- `im_deconv -d 5 -G 0.1 -r residual.d -f 3 image_in.d psf_in.d ima_out.d`
`im_op ima_out.d + residual.d clean_map.d`
 Use the CLEAN algorithm. The loop-gain parameter is set to 0.1, the clean beam has a FWHM equal to 3. The residual is added to the output in order to form the clean map.
- `im_deconv -d 6 -G 40 -C 0.1 image_in.d psf_in.d ima_out.d`
 Deconvolution by MEM method. The regularization parameter is set to 40, and the iteration parameter to 0.1.
- `im_deconv -d 7 -G 1 -C 0.5 -M ModelFileName image_in.d psf_in.d ima_out.d`
 Deconvolution by MEM method, using the Gull and Skilling entropy function. The model image (i.e. normally close to the background level) is given using “-M” option.
- `im_deconv -d 8 -G 0.1 image_in.d psf_in.d ima_out.d`
 Deconvolves an image by the Tikhonov method. Regularization parameter is set to 0.1.

4.6.3 Multiresolution methods: mr_deconv

Program `mr_deconv` deconvolves an image using the multiresolution support, assuming a space invariant point spread function (PSF) [183, 182, 152, 196].

USAGE: `mr_deconv option image_in psf_in image_out`

where options are

- **[`-d type_of_deconvolution`]**
 1. Deconvolution by multiresolution Van Cittert algorithm
 Regularization of Van Cittert’s algorithm using the multiresolution support. if “-G” is set, a Total Variation smoothness constraint is added.
 2. Deconvolution by multiresolution gradient algorithm
 Regularization of fixed step gradient algorithm using the multiresolution support. if “-G” is set, a Total Variation smoothness constraint is added.
 3. Deconvolution by multiresolution Richardson-Lucy algorithm
 Regularization of Richardson-Lucy algorithm using the multiresolution support. if “-G” is set, a Total Variation smoothness constraint is added.
 4. Deconvolution by multiresolution MAP algorithm
 Regularization of MAP algorithm using the multiresolution support.
 5. Deconvolution by the division in Fourier space + Wavelet filtering
 This method is also called Wavelet-Waguelette decomposition. It consists of first dividing the image by the PSF in Fourier space, and then applying spatial filtering using the wavelet transform, and considering the correct noise behavior. Indeed, after the division, the noise is not white anymore.

Default is deconvolution by the multiresolution Lucy algorithm (3).

- **[`-t type_of_multiresolution_transform`]**
 See 3.2.1. Transforms 1 to 24 are available.

- **[`-T type_of_filters`]**
- **[`-u`]**
- **[`-g sigma`]**
- **[`-c gain,sigma,mean`]**
- **[`-m type_of_noise`]**

- **[-n number_of_scales]**
- **[-s NSigma]**
- **[-i number_of_iterations]**
Maximum number of iterations. Default is 500.
- **[-e Epsilon]**
Convergence parameter.
Default is 1e-3.
- **[-R RMS_Map_File_Name]**
- **[-f ICF_Fwhm]**
Intrinsic correlation function.
Fwhm = Full Width at Half Maximum.
- **[-P]**
Suppress the positivity constraint.
- **[-I ICF_FileName]**
Intrinsic correlation function file.
- **[-F First_Guess]**
Input solution file name.
- **[-r residual_file_name]**
If this option is set, the residual is written to the file of name *residual_file_name*. By default, the residual is not written. The residual is equal to the difference between the input image and the solution convolved with the PSF.
- **[-S]**
Do not shift automatically the maximum of the PSF to the center.
- **[-p]**
Detect only positive structure. Default is no.
- **[-k]**
Suppress isolated pixels in the support. Default is no.
- **[-K]**
Suppress the last scale. This option is available only for deconvolution methods 1 and 2.
Default is no.
- **[-O]**
Optimization. Only for deconvolution methods 1 to 3.
Default is no.
- **[-G RegulParam]**
Regularization parameter (only valid for deconvolution methods 1,2 and 3). Default is 0.
- **[-v]**
Verbose.

Examples:

- `mr_deconv image_in.d psf_in.d ima_out.d`
Deconvolves an image by the regularized Richardson-Lucy algorithm, assuming Gaussian noise (its standard deviation is automatically estimated).
- `mr_deconv -F image_in.d image_in.d psf_in.d ima_out.d`
Ditto, but the used entry point is the data (it is a flat image by default). The solution will contain the noise present in the raw data, while the solution is smooth in the previous example.
- `mr_deconv -i 50 -e 0 image_in.d psf_in.d ima_out.d`
Ditto, but force the number of iterations to be equal to 50.
- `mr_deconv -K -d 2 -m 2 -r resi.d image_in.d psf_in.d ima_out.d`
Deconvolves an image by the regularized gradient method. The deconvolved image will be background free. Noise is assumed to be Poisson. The residual is saved in the file “resi.d”.
- `mr_deconv -d 1 -s 5 -p -r resi.d image_in.d psf_in.d ima_out.d
im_op ima_out.d + resi.d clean_map.d`
Run the deconvolution by the regularized Van Cittert method. Only positive wavelet coefficients are used, and the detection is done at 5σ , assuming Gaussian noise.

4.6.4 Wavelet CLEAN: mr_mrc

Program *mr_mrc* deconvolves an image using the Wavelet CLEAN method [191, 190].

USAGE: mr_mrc option image_in psf_in image_out

where options are

- `[-g sigma]`
- `[-m type_of_noise]`
 1. Gaussian noise.
 2. Undefined stationary noise.
 3. Stationary correlated noise.

Default is Gaussian noise.

- `[-n number_of_scales]`
- `[-s NSigma]`
- `[-i number_of_iterations]`
Maximum number of iterations. Default is 500.
- `[-e Epsilon]`
Convergence parameter.
Default is 1e-4.
- `[-R RMS_Map_File_Name]`
- `[-f ICF_Fwhm]`
Intrinsic correlation function.
Fwhm = Full Width at Half Maximum.
- `[-G LoopGain]`
Loop Gain parameter. Default is 0.01.

- **[-E]**
Use the Dirty Map instead of the Energy Dirty Map.
- **[-L]**
Apply CLEAN also on the last scale. Default is Van-Cittert.
- **[-A]**
Do not add the residual to the CLEAN map.
- **[-r residual_file_name]**
If this option is set, the residual is written to the file of name *residual_file_name*. By default, the residual is not written. The residual is equal to the difference between the input image and the solution convolved with the PSF.
- **[-K]**
Suppress the last scale. Default is no.
- **[-v]**
Verbose.

Example:

- `mr_mrc image_in.d psf_in.d ima_out.d`
Deconvolves an image by the Wavelet-CLEAN method, assuming Gaussian noise (its standard deviation is automatically estimated).

Chapter 5

MR/1 Image Compression

5.1 Introduction

Image compression can be necessitated for different reasons, which may in practice motivate different compression strategies. Cases which can be easily distinguished include: quicklook, very large images (e.g. the all-sky atlas, ALADIN, with links to cataloged information on tens of millions of astronomical objects [11]); fast access to large pictorial databases; and Web-based and other types of data transmission (a number of references to past work on progressive image transmission can be found in [79]).

Following the kind of images and the application needs, different strategies can be used:

1. Lossy compression: in this case the compression ratio is relatively low (< 5).
2. Compression without visual loss. This means that one cannot see the difference between the original image and the decompressed one. Generally, compression ratios between 10 and 20 can be obtained.
3. Good quality compression: the decompressed image does not contain any artifact, but some information is lost. Compression ratios up to 40 can be obtained in this case.
4. Fixed compression ratio: for some technical reasons, one may decide to compress all images with a compression ratio higher than a given value, whatever the effect on the decompressed image quality.
5. Signal/noise separation: if noise is present in the data, noise modeling can allow very high compression ratios just by including filtering in wavelet space during the compression.

Following the image type, and the selected strategy, the optimal compression method may differ. A major interest in using a multiresolution framework is to get, in a natural way, the possibility for progressive information transfer.

According to Shannon's theorem, the number of bits we need to code an image I without distortion (losslessly) is given by its entropy H . If the image (with N pixels) is coded with L intensity levels, each level having a probability p_i to appear, the entropy H is

$$H(I) = \sum_{i=1}^L -p_i \log_2 p_i \quad (5.1)$$

The probabilities p_i can be easily derived from the image histogram. The compression ratio is given by:

$$\mathcal{C}(I) = \frac{\text{number of bits per pixel in the raw data}}{H(I)} \quad (5.2)$$

and the distortion is measured by

$$R = \| I - \tilde{I} \| = \sum_{k=1}^N (I_k - \tilde{I}_k)^2 \quad (5.3)$$

where \tilde{I} is the decompressed image.

A Huffman, or an arithmetic, coder is generally used to transform the set of integer values into the new set of values, in a reversible way.

Compression methods try to use the redundancy contained in the raw data in order to reduce the number of bits. The main efficient methods belong to the transform coding family, where the image is first transformed into another set of data where the information is more compact (i.e. the entropy of the new set is lower than the original image entropy). The typical steps are:

1. transform the image (for example using a discrete cosine transform, or a wavelet transform),
2. quantize the obtained coefficients, and
3. code the values by a Huffman or an arithmetic coder.

The first and third points are reversible, while the second is not. The distortion will depend on the way the coefficients are quantized. We may want to minimize the distortion with the minimum of bits, and a trade-off will then be necessary in order to also have “acceptable” quality. “Acceptable” is evidently subjective, and will depend on the application. Sometimes, any loss is unacceptable, and the price to pay for this is a very low compression ratio (often between one and two).

5.2 Compression Methods

We review briefly here different methods:

- **HCOMPRESS:** HCOMPRESS [214] was developed at Space Telescope Science Institute (STScI), Baltimore, and is commonly used to distribute archive images from the Digital Sky Survey. It is based on the Haar wavelet transform. The algorithm consists of
 1. applying a Haar transform to the data,
 2. quantizing the wavelet coefficients linearly as integer values,
 3. applying a quadtree to the quantized values, and
 4. using a Huffman coder.
- **HCOMPRESS + iterative decompression:** Iterative decompression was proposed [15] to decompress files which were compressed using HCOMPRESS. The idea is to consider the decompression problem as a restoration problem, and to add constraints on the solution in order to reduce the artifacts.
- **FITSPRESS:** FITSPRESS [155] is based on a wavelet transform, using Daubechies-4 filters. It was developed at the Center for Astrophysics, Harvard.
- **JPEG:** JPEG is the standard video compression software for single frame images [80]. It decorrelates pixel coefficients within 8 by 8 pixel blocks using the Discrete Cosine Transform and uniform quantization.
- **Wavelet:** Bi-orthogonal wavelet transform. The 7/9 filter is one of the best filters. [7]
- **Fractal:** The image is decomposed by block, and each block is represented by a fractal. See [73] for more explanation.
- **Mathematical morphology:** This method is based on mathematical morphology [95] (erosion and dilation). It consists of detecting structures above a given level, the level being equal to the background plus three times the noise standard deviation. Then, all structures are compressed using erosion and dilation, and additionally a quadtree and Huffman coding. This method is based on object detection, and leads to a high compression ratio if the image does not contain a lot of information, as is often the case in astronomy.
- **Pyramidal median transform:** The principle of this compression method is to select the information we want to keep, by using the pyramidal median transform, and to code this information without any loss. Thus the first phase searches for the minimum set of quantized multiresolution coefficients which produce an image of “high quality”. The quality is evidently subjective, and we will define by this term an image such as the following:
 - there is no visual artifact in the decompressed image,
 - the residual (original image – decompressed image) does not contain any structure.

Lost information cannot be recovered, so if we do not accept any loss, we have to compress what we take as noise too, and the compression ratio will be low (3 or 4 only).

The PMT is similar to the mathematical morphology method in the sense that both try to understand what is in the image, and to compress only what is considered as significant. The PMT uses a multiresolution approach, which allows more powerful separation between signal and noise. Both methods have been developed for astronomical image compression. They certainly can also be used for medical, or biological images.

Concerning computation time for such methods, it is in general very competitive, of the order of a few seconds for a 1024×1024 integer (16 bits) image (machine Sun Ultra-Enterprise,

250 MHz and 1 processor). Only the methods producing a fixed compression ratio take more time, because they need to iterate in order to obtain a good compression ratio.

The MR/1 compression program allows us to use any of the previously described strategies. User parameters define the chosen compression ratio, the noise model, etc. For large images, compression can be performed by blocks, and block size is a parameter. This allows us to decompress a given block (or area) at a given resolution, without having to read the whole file. The file format has been specified in order to gain direct access to a given block at a given resolution using the C command “fseek”. Available methods are bi-orthogonal wavelet transform, Haar and Feauveau wavelet transform, PMT, mathematical morphology.

5.3 Large Image Visualization Environment: LIVE

With new technology developments, images furnished by detectors are larger and larger. For example, current astronomical projects plan to deal with images of sizes larger than 8000 by 8000 pixels (VLT: 8k × 8k, MegaCam 16k × 16k, etc.). A digitized mammogram film may lead to images of about 5k × 5k. Analysis of such images is obviously not easy, but the main problem is clearly that of archiving and network access to the data by users.

In order to visualize an image in a reasonable amount of time, the transmission is based on two concepts:

- data compression
- progressive decompression

The concept of progressive transmission has appeared in recent years. It consists of visualizing quickly a low resolution image, and then increasing the quality with the arrival of new bits. Wavelet based methods become very attractive for this, because they integrate in a natural way this multiresolution concept. A coarse resolution image can be directly obtained without having to decompress the whole file.

But with very large images, a third concept is necessary, which is the region of interest. Indeed, images are becoming so large it is impossible to display them in a normal window (typically of size 512 × 512), and we need to have the ability to focus on a given area of the image at a given resolution. To move from one area to another, or to increase the resolution of a part of the area is a user task, and is a new active element of the decompression. The goal of LIVE is to furnish this third concept.

An alternative to LIVE for extracting a region of the image would be to let the server extract the selected area, compress it, and send it to the user. This solution is simpler and does not need any development, but presents several drawbacks:

- Server load: the server must decompress the full size image and re-compress the selected area on each user request.
- Transfer speed: indeed, to improve the resolution of a 256 × 256 image to a 512 × 512 image, the number of bits using the LIVE strategy is of the order of 10 to 100 less than if the full 512 × 512 is transferred. The lower the compression ratio is, the more the LIVE strategy becomes important.

The principle of LIVE is to use the previously described technology, and to add the following functionalities:

- Full image display at a very low resolution.

- Image navigation: the user can go up (the quality of an area of the image is improved) or down (return to the previous image). Going up or down in resolution implies a four-fold increase or decrease in the size of what is viewed.

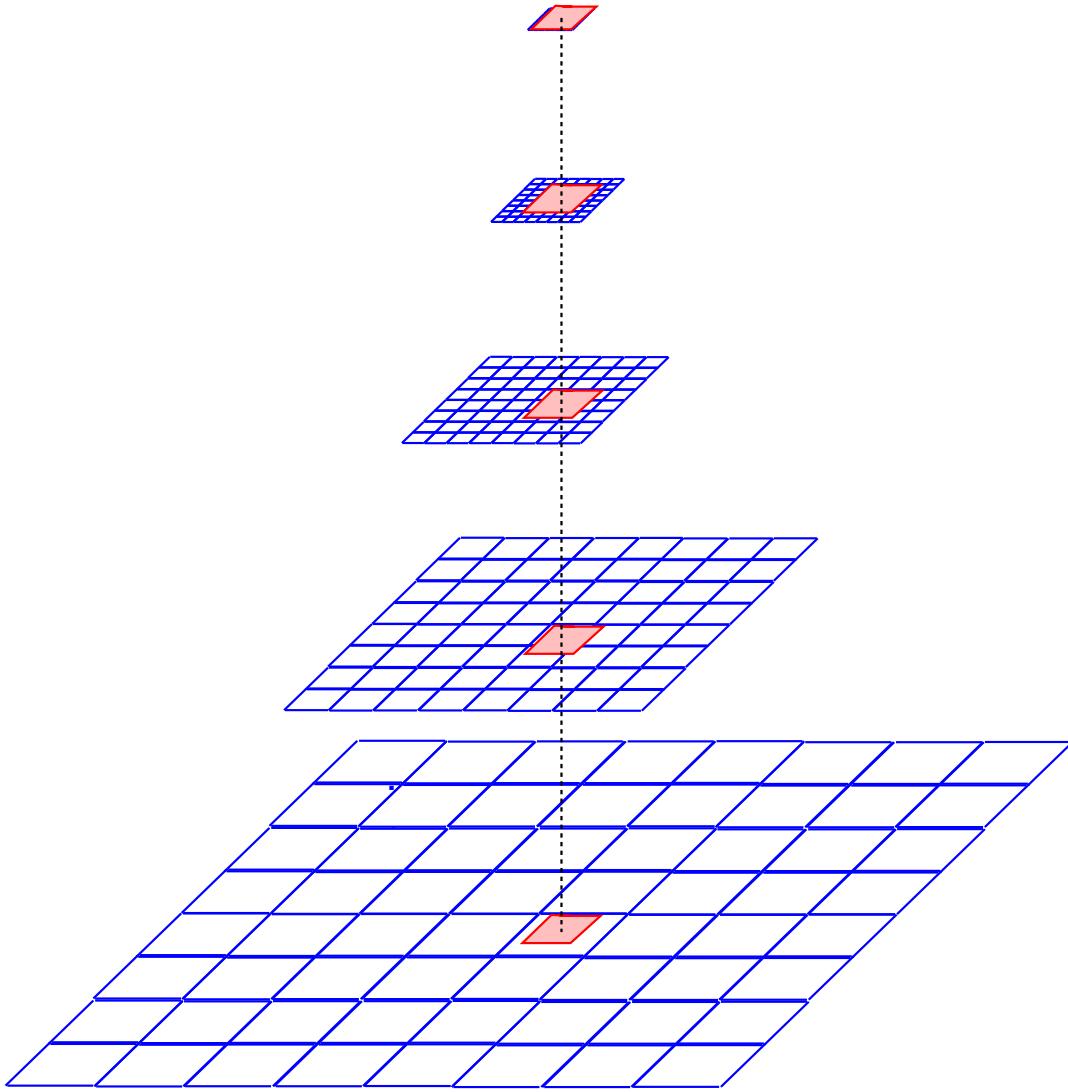


Figure 5.1: Example of large image, compressed by block, and represented at five resolution levels. At each resolution level, the visualization window is superimposed at a given position. At low resolution, the window covers the whole image, while at the full resolution level, it covers only one block.

In order to illustrate this concept, see Figure 5.1. A large image (say 4000×4000), which is compressed by blocks (8×8 , each block having a size of 500×500), is represented at five resolution levels. The visualization window (of size 256×256 in our example) covers the whole image at the lowest resolution level (image size 250×250), but only one block at the full resolution (in fact between one and four, depending on the position in the image). The LIVE concept consists of moving the visualization window into this pyramidal structure, without having to load into memory the large image. The image is first visualized at low resolution,

and the user can indicate (using the mouse) which part of the visualized subimage he wishes to zoom on. At each step, only wavelet coefficients of the corresponding blocks and of the new resolution level are decompressed.

An IDL program is available (XLIVE) in MR/1 which allows manipulation and visualization of large images. Future versions of MR/1 will also include the LIVE concept in Java, allowing use over the network in client-server mode.

5.4 Compression/Decompression Programs

Three image compression programs are available:

- *mr_comp*: lossy image compression by several methods. This program is well adapted to the compression of images contaminated by noise. The pyramidal median transform (default method in *mr_comp*) is very efficient for astronomical image compression.
- *wcomp*: lossy image compression by the bi-orthogonal wavelet transform (7/9 filters), and non-uniform quantization. This program is similar to *mr_comp* with specific options (i.e. “-N -m 5 -s 1”). It also allows the user to evaluate the PSNR between the original image and the decompressed one.
- *mr_lcomp*: lossless image compression using the lifting scheme method.

These programs furnishes a “.MRC” file format, and the decompression can be performed using the *mr_decomp* program.

5.4.1 Image Compression: wcomp

Program *wcomp* compresses an image by the bi-orthogonal wavelet transform.

For very large images, the compression can be carried out by block using the “-C” option. Then, each block is considered as an independent image, and compressed. Only one block need be in memory for this approach. Another advantage is the fact that a block can be decompressed alone, without decompression of the whole image.

USAGE: wcomp option image_in [compressed_file_out]

where options are:

- **[-g QuantifParam]**
Quantization parameter.
- **[-n number_of_scales]**
Number of scales used in the multiresolution transform. Default is 6.
- **[-E]**
Evaluation mode. Once the compression is finished, decompression is performed, and the PSNR between the original and decompressed image is calculated.
- **[-C BlockSize]**
Compress by block. *BlockSize* is the size of each block. Default is not to compress block-wise.
- **[-v]**
Verbose. Default is no verbose output.

Examples:

- *wcomp image.d*
Compress the image, and store the result “image.d.MRC”
- *wcomp -g 30 image.d*
Ditto, but increase the compression rate.

- `mr_comp -E image.d comp.MRC`
Compress the image, store the result in “comp.MRC”, and calculate the PSNR between the original and the decompressed image.
- `mr_comp -C 512 image.d`
Compress the image by block of size 512x512.

5.4.2 Noisy image Compression: `mr_comp`

Program `mr_comp` compresses an image by the pyramidal median transform (PMT) [195, 193, 142, 141], by mathematical morphology [95], or by wavelet transforms.

The PMT method uses noise modeling, assuming Gaussian, Poisson, or Gaussian + Poisson noise. By default, the noise is not compressed, and the decompressed image will look like a filtered image. Option “-s” fixes the detection level, for the separation between signal and noise, and option “-q” specifies the quality we want when the detected multiresolution coefficients are coded. If the user wants to keep the noise, two options allow him or her to do this. Option “-l” for lossless compression (see also the `mr_lcomp` program), and option “-r” when an error on the noise reconstruction is acceptable. For the second case, option “-e” fixes the quality of the noise reconstruction. By default, all operations are done using float. By using the “-O” option, all operations are done using integer (the program runs faster), and the input image is first converted into an integer image. If the input image is coded with floating values, some information may be lost during this conversion. If no output file name is given, the program uses the input file name, and adds to it the suffix “.MRC”.

For the mathematical morphology method, only “-O”, “-S” and “-D” options are valid, “-S”, “-D” allowing us to change the structuring element and its size.

For other compression methods, options are the same as for PMT, but “-O” is not valid.

If the data does not contain any noise, option “-N” must be set. In this case, the compression ratio cannot be determined from the significant coefficients. Then, the parameter “-q” can be used for increasing the compression ratio, or the compression ratio can also be fixed, using the “-R” option. This option implies that we need to iterate in order to get an optimal distortion for the given compression ratio. For this reason the “-R” option requires more computation time.

Depending on the kind of data, `mr_comp` allows different strategies to be taken. For astronomical and medical images, which contain noise and diffuse objects, the PMT transform gives good results. For an image without any noise and containing edges, the bi-orthogonal wavelet transform (option -m 5) is better. For lossless compression (i.e. output decompressed image = input image), the min-max transform can also be considered.

For very large images, the compression can be carried out by block using the “-C” option (see previous section).

USAGE: `mr_comp option image_in [compressed_file_out]`

where options are:

- **[-m Compression_Method]**
 1. Compression by the pyramidal median transform
 2. Compression by mathematical morphology
 3. Compression by the Haar transform
 4. Compression by the min-max transform (or G transform)
 5. Compression by the Mallat-Daubechies transform

6. Compression by the Feauveau transform
7. Compression by mixed WT and PMT method

Default is Compression by the pyramidal median transform.

- **[-g SigmaNoise]**

SigmaNoise = Gaussian noise standard deviation. Default is automatically estimated.

- **[-p]**

Poisson Noise. Default is no Poisson component (hence just Gaussian).

- **[-c gain,sigma,mean]**

See section 3.3.2.

- **[-n number_of_scales]**

Number of scales used in the multiresolution transform. Default is 6.

- **[-s NSigma]**

Thresholding at NSigma * SigmaNoise. Default is 3.

- **[-r]**

If -r option is given, the noise is compressed with a step of sigma_noise/2. By default, the noise is not conserved.

- **[-k]**

Keep isolated pixels in the support at the first scale. Default is not to do this. If the point spread function is defined on only one pixel, this option should be set.

- **[-l]**

Save the noise (for lossless compression). Default is not to do this.

- **[-q SignalQuantif]**

The signal is quantized by $Sq = S / (\text{SignalQuantif} * \text{Sigma_Noise})$. By default, SignalQuantif = 1.5

- **[-e NoiseQuantif]**

The signal is quantized by $Nq = S / (\text{NoiseQuantif} * \text{Sigma_Noise})$. By default, NoiseQuantif = 0.5.

- **[-f]**

Keep all the header in the case of FITS format files. Default is not to do this.

- **[-b bad_pixel_value]**

All pixels with this value will be considered as bad pixels, and not used for the noise modeling.

- **[-O]**

Optimization. If set, the program works with integer instead of float, and the execution time is faster.

- **[-B]**

Optimization without BSCALE (rescaling) operation in the case of FITS images.

- **[-P]**

Keep only positive coefficients.

- **[-W]**

Set the Median window size equal to 5. Default is 3.

- **[-S]**

Use a square structuring element. (Only for mathematical morphology compression). Default structuring element is a circle.

- **[-D Dim]**
Dimension of the structuring element (Only for mathematical morphology compression). Default is 5.
- **[-R Compression_Ratio]**
Fixes the compression ratio. Default is not to do this.
- **[-C BlockSize]**
Compress by block. *BlockSize* is the size of each block. Default is No.
- **[-i NbrIter]**
Method 3 (using Haar transform) can be improved by iterating. *NbrIter* fixes the number of iterations. Default is not to do this.
- **[-N]**
Do not use noise modeling.
- **[-v]**
Verbose. Default is no verbose output.

Examples:

- mr_comp image.d
Compress the image, and store the result “image.d.MRC”
- mr_comp -f image.fits icompress
Compress the image, and store the result “icompress.MRC”. The complete FITS header is saved in the compressed file.
- mr_comp -m 5 -R 30 image.d
Compress the image with a compression ratio of 30, using a bi-orthogonal wavelet transform.
- mr_comp -m 5 -q 30 image.d
Compress the image, fixing the quantization parameter, and using a bi-orthogonal wavelet transform.
- mr_comp -O -m 4 -N image.d
Apply optimized lossless compression.

5.4.3 Compression: mr_lcomp

Program *mr_lcomp* compress an image by an integer wavelet transform via the lifting scheme. The integer wavelet coefficients are coded without losing any information. This leads to a lossless compression method (if the input image contains integer values!). Five transforms are available.

USAGE: **mr_lcomp option image_in [compressed_file_out]**

where options are:

- **[-m Compression_Method]**

1. Lifting scheme: median prediction.
2. Lifting scheme: integer Haar WT.
3. Lifting scheme: integer CDF WT .
4. Lifting scheme: integer (4,2) interpolating transform.
5. Lifting scheme: integer 7/9 WT.

Default is Lifting scheme: integer Haar WT.

- **[-n number_of_scales]**

Number of scales used in the multiresolution transform. Default is 6.

- **[-f]**

Keep all the header in the case of FITS format files. Default is not to do this.

- **[-B]**

Optimization without BSCALE (rescaling) operation in the case of FITS images.

- **[-v]**

Verbose. Default is no verbose output.

- **[-C BlockSize]**

Compress by block. *BlockSize* is the size of each block. Default is No.

5.4.4 Compression strategies

Compression method

If the images contain isotropic features (the case of medical, biomedical, and astronomical images), the PMT (with *mr_comp*) transform is a good choice. If the image contains edges, a bi-orthogonal transform should be used (option “-m 5” with *mr_comp*, and “-m 2” or “-m 3” with *mr_lcomp*).

Lossless compression

For lossless compress, *mr_lcomp* is normally a better choice than *mr_comp*. However, *mr_comp* separates signal from noise, which can be in many cases useful. The Haar lifting scheme method has also an advantage, which is that a pixel in the decompressed image at a lower resolution corresponds to the average of the pixel values in a square defined by the resolution level. This allows the user to make correct scientific measurement without having to decompress the whole image. For this reason, the Haar lifting scheme compression method would be a good candidate for a new data format.

Noise model

By default, *mr_comp* tries to separate the noise from the signal. If the image is not noisy, this operation should not be carried out, and the option “-N” must be set.

Compression ratio

To obtain a fixed compression ratio, the option “-R x” (x being the compression ratio value) has to be given. The price to be paid is that the compression takes more time because it iterates until the correct compression ratio is obtained.

Large images

For very large images, the “-C” option implies that block compression will be performed. The image is separated into $N \times M$ blocks and each block is compressed separately. Blocks are directly read from disk, so the compression takes more time, but requires less memory space. A block can be decompressed individually and at any resolution level, a property used by the IDL XLIVE program.

5.4.5 Decompression: `mr_decomp`

Program `mr_decomp` decompresses a file compressed with `mr_comp` or `mr_lcomp`. We are not always interested in decompressing the image at full resolution. The option “-r” allows an image at lower resolution to be extracted from the compressed file, and produces in this way a smaller image than the original. Even if the compressed file contains the noise of the input image, the noise will not be decompressed. Note that for lower resolution decompression, only the necessary part of the file is read and decompressed (and so the decompression is particularly efficient). This is possible because of the pyramidal compression method. The option “-g” recreates a noise map, and adds it to the decompressed image. The final image is evidently not closer to the original one, but the appearance will be very similar.

USAGE: `mr_decomp` option CompressedFileName OutputFileName

where options are:

- **[-B BlockNbr]**
Decompress only one block. *BlockNbr* is the block number to decompress. Default is no.
- **[-r resolution]**
resolution must be ≥ 0 and $<$ number of scales of the transform. By default, the image is reconstructed at full resolution with its noise if this exists in the compressed file.
- **[-t output_type]**
If the input image was a FITS format image, the image output type can be fixed by the user to “i” for integer, or “s” for short. By default, the output type is the same as the type of the original image.
- **[-g]**
Add simulated noise to the decompressed image with the same properties as in the original image. This preserves appearances.
- **[-v verbose]**
Default is no.
- **[-I IterRecNbr]**
Use an iterative reconstruction. Only used with orthogonal transforms. Default is no iteration.

Examples:

- `mr_comp ngc2997.fits`
Compress the image, and store the result “ngc2997.fits.MRC”
- `mr_decomp ngc2997.fit.MRC decngc.fits`
Decompress the file, and store the result “decngc.fits”.
- `cat ngc2997.fits.MRC | mr_decomp - - > decngc.fits`
Unix command, doing the same job.
- `mr_decomp -r 1 ngc2997.fit.MRC decngc.fits`
Decompress the file at a resolution lower. The decompressed image has size 128×128 when the original one had a size of 256×256 .
- `mr_decomp -r 3 -i ngc2997.fit.MRC decngc.fits`
Same as before, but the decompressed image has size 32×32 .

5.4.6 Decompression: mr_upresol

From an image which has been compressed by *mr_comp* or *mr_lcomp*, we can extract a low resolution image. The principle of *mr_upresol* is to improve the resolution, by a factor two, of the low resolution image. Only wavelet coefficients of the new scale are extracted from the compressed file (MRC format). This program is used by the IDL widget program XLIVE.

If the image was compressed by block (“-C” option using *mr_comp* or *mr_lcomp*), then *mr_upresol* allows us to focus on a region of the image. The coordinates of the center of the region are given either by the “-x” and “-y” options, or by “-X” and “-Y” options. In the first case, the coordinates must begin in the pixel units of the low resolution image, while in the second case, units are those of the full resolution image. The minimum size of the region to visualize is given by the option “-W”. Only blocks covering the selected region will be decompressed.

Two files are created. The new image file, and an information file (suffix “.inf”) which is an ascii file and which contains information about the decompressed image. It contains the three following lines:

```
L_Nl L_Nc Nl Nc
Resol BlockSize KeepResi
FirstBlockNl LastBlockNl FirstBlockNc LastBlockNc
```

where $L_Nl \times L_Nc$ is the size of the full resolution image, $Nl \times Nc$ is the size of the decompressed image, Resol its resolution level, BlockSize is the block size used for the compression, KeepResi indicates whether the noise map is stored in the compressed file, and FirstBlockNl,LastBlockNl,FirstBlockNc, LastBlockNc indicates which blocks are decompressed.

USAGE: mr_upresol options CompressedFileName OutputFileName

where options are:

- **[-t output_type]**

If the input image was a FITS format image, the image output type can be fixed by the user to “i” for integer, or “s” for short. By default, the output type is the same as the type of the original image.

- **[-I IterRecNbr]**

Use an iterative reconstruction. Only used with orthogonal transforms. Default is no iteration.

- **[-l LowResol_Image_FileName]**

Image at a lower resolution.

- **[-x PosX]**

Center pixel position (x) of the region to zoom in the input low resolution image. Default value is 0.

- **[-y PosY]**

Center pixel position (y) of the region to zoom in the input low resolution image. Default value is 0.

- **[-X PosX]**

Center pixel position (x) of the region to zoom in the full resolution image (real pixel coordinate). Default value is 0.

- **[-Y PosY]**

Center pixel position (y) of the region to zoom in the full resolution image (real pixel coordinate). Default value is 0.

- **[-W WindowSize]**
Window size. Default is 256.
- **[-v verbose]**
Default is no.

Examples:

- mr_lcomp -n 4 ngc2997.fits ngc2997.fits.MRC
mr_upresol ngc2997.fits.MRC resol3.fits
mr_upresol -l resol3.fits ngc2997.fits.MRC resol2.fits
mr_upresol -l resol2.fits ngc2997.fits.MRC resol1.fits
mr_upresol -l resol1.fits ngc2997.fits.MRC resol0.fits
Compress first image, decompress it resolution by resolution.
- mr_lcomp -n 4 -C 512 ngc2997.fits ngc2997.fits.MRC
mr_upresol -X 200 -Y 250 -W 400 -l resol3.fits ngc2997.fits.MRC resol2.fits
mr_upresol -X 200 -Y 250 -W 400 -l resol2.fits ngc2997.fits.MRC resol1.fits
mr_upresol -X 200 -Y 250 -W 400 -l resol1.fits ngc2997.fits.MRC resol0.fits
Compress first image by block of 512. An image is created for each resolution level, which contains the region around the pixels of coordinates (200,250). The “-W” options specifies the size of the region of interest (here 400 pixels). At each level, only blocks covering this surface area are decompressed.

Chapter 6

MR/1 Object Detection

6.1 Multiscale Vision Model

6.1.1 Introduction

After applying the wavelet transform on the image, we have to detect, to extract, to measure and to recognize the significant structures. The wavelet space of a 2D direct space is a 3D one. An object has to be defined in this space. A general idea for object definition lies in the connectivity property. An object occupies a physical region, and in this region we can join any pixel to other ones. The connectivity in the direct space has to be transported to the wavelet transform space (WTS). In order to define the objects we have to identify the WTS pixels we can attribute to the objects.

Figure 6.1.1 shows an example of connectivity between detected structures at different scales.

6.1.2 Definition

The Multiscale Vision Model (MVM) [19, 160] described an object as a hierarchical set of structures. It uses the following definitions:

- **Structure:** a structure S_j is a set of significant connected wavelet coefficients at the same scale j .
- **object:** an object is a set of structures.
- **object scale:** the scale of an object is given by the scale of the maximum of its wavelet coefficients.
- **interscale-relation:** the rule which allows us to connect two structures into a single object is called “interscale relation”. Let us consider two structures at two successive scales, S_j^1 and S_{j+1}^2 . Each structure is located in one of the individual images of the decomposition and corresponds to a region in this image where the signal is significant. Noting p_m the pixel position of the maximum wavelet coefficient value of S_j^1 , S_j^1 is said to be connected to S_{j+1}^2 if S_{j+1}^2 contains the pixel position p_m (i.e. the maximum position of the structure S_j^1 must also be contained in the structure S_{j+1}^2). Several structures appearing in successive

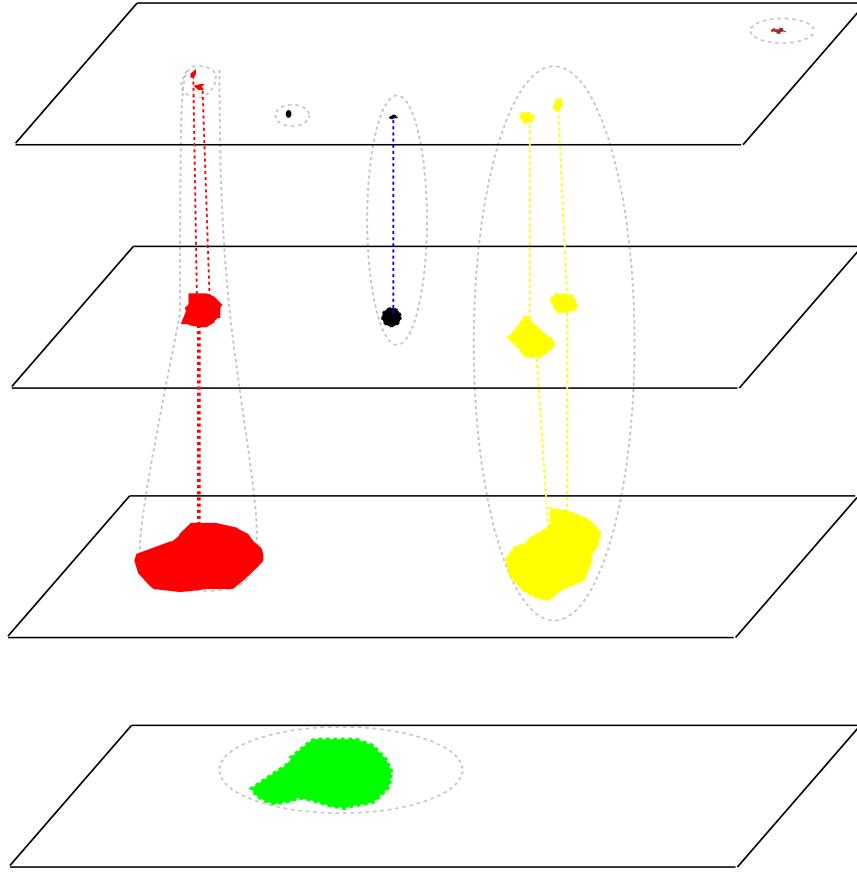


Figure 6.1: Example of connectivity in the wavelet space: contiguous significant wavelet coefficients form a structure, and following an interscale relation, a set of structures form an object. Two structures S_j, S_{j+1} at two successive scales belong to the same object if the position pixel of the maximum wavelet coefficient value of S_j is included in S_{j+1} .

wavelet coefficient images can be connected in such a way, which we call an object in the interscale connectivity graph.

- **sub-object:** a sub-object is a part of an object. It appears when an object has a local wavelet maximum. Hence, an object can be composed of several sub-objects. Each sub-object can also be analysed.

6.1.3 Reconstruction

The problem of reconstruction [19, 160] consists of searching for a signal O such that its wavelet coefficients are the same as those of the detected structures. If \mathcal{T} describes the wavelet transform operator, and P_w the projection operator in the subspace of the detected coefficients (i.e. all coefficients set to zero at scales and positions where nothing was detected), the solution is found by minimization of

$$J(O) = \| W - (P_w \circ \mathcal{T})O \|$$

where W represents the detected wavelet coefficients of the data.

The MVM presents many advantages compared to the standard approach:

- faint extended objects can be detected as well as point sources,
- the analysis does not require background estimation,
- the Point Spread Function (PSF) is not needed.

The second point is relatively important. Indeed, if the background varies spatially, its estimation becomes a non-trivial task, and may produce large errors in the object photometry.

The last point is an advantage when the PSF is unknown, or difficult to estimate, which occurs relatively often when it is space variant. However, when the PSF is well determined, it becomes a drawback because known information is not used for the object reconstruction. Such a situation leads to systematic errors in the photometry, which depends on PSF and on the source signal to noise ratio. In order to correct this error, a kind of calibration must be performed using simulations [180]. The next section shows how the PSF can be used in the MVM, leading to a deconvolution.

6.2 Detection and Deconvolution

6.2.1 Object reconstruction using the PSF

A reconstructed and deconvolved object can be obtained by searching for a signal O such that the wavelet coefficients of $P * O$ are the same as those of the detected structures. If \mathcal{T} describes the wavelet transform operator, and P_w the projection operator in the subspace of the detected coefficients, the solution is found by minimization of

$$J(O) = \| W - (P_w \circ \mathcal{T})P * O \| \quad (6.1)$$

where W represents the detected wavelet coefficients of the data, and P is the point spread function. By this approach, each object is deconvolved separately. The flux related to the ring of the PSF will be taken into account. For a point source, the solution will be close to that obtained by PSF fitting. This problem is different from global deconvolution in the sense that it is well-constrained. Except for the positivity of the solution which is always true and must be used, no other constraint needs to be introduced. This is due to the fact that the reconstruction is performed from a small set of wavelet coefficients (those above a detection limit). The number of objects are the same as those obtained by the MVM, but the photometry and the morphology are different. The position may also change a little.

6.2.2 The algorithm

Any minimizing method can be used to obtain the solution O . As we did not meet any problem of convergence, noise amplification, or ringing effect, we choose the Van Cittert method, which is certainly the simplest one. For each detected object, we apply the following algorithm:

$$O^{n+1} = O^n + \mathcal{T}^{-1}(W - (P_w \circ \mathcal{T})P * O^n) \quad (6.2)$$

where \mathcal{T}^{-1} is the inverse wavelet transform.

1. Set n to 0.

2. Find the initial estimation O^n by applying an inverse wavelet transform to the set W corresponding to the detected wavelet coefficients in the data.
3. Convolve O^n with the PSF P : $I^n = P * O^n$.
4. Determine the wavelet transform $W(I^n)$ of I^n .
5. Threshold all wavelet coefficients in $W(I^n)$ at position and scales where nothing has been detected (i.e. P_w operator). We get $W_t(I^n)$.
6. Determine the residual $W(R) = W - W_t(I^n)$.
7. Reconstruct the residual image R^n applying an inverse wavelet transform.
8. Add the residual to the solution: $O^{n+1} = O^n + R^n$.
9. Threshold negative values in O^{n+1} .
10. if $\sigma(R^n)/\sigma(O^0) < \epsilon$ then $n = n + 1$ and goto step 3.
11. O^{n+1} contains the deconvolved reconstructed object.

In practice, the convergence is very fast (less than 20 iterations). The reconstructed image (not deconvolved) can also be obtained, just by reconvolving the solution with the PSF.

6.3 Object Detection: mr_detect

Program *mr_detect* detects all objects present in an image [19, 160]. Several output files are created:

- name_out.tex: LaTeX file which contains a table describing all detected objects. Values given in the table are respectively the object number, the coordinates in pixel units, the standard deviation in x and y , the orientation, the maximum value, the flux and the signal to noise ratio of the maximum wavelet coefficient of the object. This file can be compiled by the LaTeX command. The pixel coordinates are given using the first pixel of the image as the reference pixel (0,0). The rotation angle gives the angle between the main axis of the object and the x-axis following the trigonometric convention. The object number is defined by numbers: the scale number which indicates at which scale the object is detected, and a number which indicates the object number at this scale.
- name_out_obj.ps: a Postscript file giving the position of all objects in the image.
- name_out.fits: this image contains the sum of all objects. It looks like a filtered image, without any background.
- name_out.mes: ASCII file describing all detected objects. Each object is described by five lines:
 - the first line contains the scale where the maximum of the object has been detected, and the object number at this scale.
 - the second line contains respectively the coordinates in pixel units (x and y), and the standard deviation in x and y .
 - the third line contains the orientation, the maximum value, the flux, and the magnitude.
 - the fourth line contains the flux error, the signal to noise ratio (SNR) of the maximum of the wavelet coefficient, and the SNR of the object. The SNR of the wavelet coefficient gives the ratio between the maximum of the wavelet coefficients belonging to the object, and the standard deviation of the noise at the scale of this maximum. The SNR of the object is obtained by calculating the standard deviation in a box containing 90% of the flux of the object in the reconstructed image, and by taking the ratio between this standard deviation and the noise standard deviation. In the case of Poisson noise with few events, the SNR of the wavelet coefficient is not calculated, but instead the probability that the wavelet coefficient is not due to noise (i.e. due to signal), and the SNR of the object, is calculated by taking the ratio between 90% of the flux of the source and the square root of the flux contained in the same box in the original data.
 - the last line contains the position of the maximum wavelet coefficient.

Furthermore, if the option “-w 1” is set, an image is created for each object individually. For a FITS image, if the “-C” option is set, another TeX table is created containing the object coordinates (RA, Dec), the flux and the SNR.

If the Point Spread Function (PSF) is available, it can be given to the program using the “-P” option. It generally improves the object photometry. Furthermore an object deconvolution can be performed using the “-D” option. In this case, each object is also deconvolved.

The photometry is by default estimated by integrating the flux in the reconstructed object. An alternative is to use an aperture photometry (“-a” option). In this case, a background image is needed. It can be:

- automatically calculated using pyramidal transform (PMT) (“-a 1”). It is found by interpolating the last scale of the PMT to the input image size. The number of scales is derived from the input image size and the number of pixels in the last scale (default is 16, and can be changed with the “-b” option). The background image size must be large enough to take into account background variations, and small enough to no take into account the information relative to the significant signal.
- a flat image (“-a 3”), with a value equal to *BgrValue*.
- an image given by the user (“-a 2” + “-B” option).

An object is characterized by its second order moments σ_x, σ_y in the two principal directions. The aperture photometry is done in a box of size $kMAX(\sigma_x, \sigma_y)$, where k is defaulted to 3. k can be modified by the “-l” option.

USAGE: mr_detect option image_in name_out

image_in is the input image and **name_out** the name which will be used for the output files. This name must not contain any suffix.

Options are:

- **[-t type_of_multiresolution_transform]**
 1. B-spline wavelet transform: à trous algorithm
 2. Half-pyramidal transform
 3. Pyramidal B-spline wavelet transform
 4. Mixed WT and PMT method (WT-PMT)
 5. Mixed Half-pyramidal WT and Median method (WT-HPMT)

Default is 1.

- **[-V Multiscale_Vision_Model]**
 1. No vision model.
 2. Blinded Objects.
 3. Rue-Bijaoui Vision Model for blinded + embedded Objects.

Default is Rue-Bijaoui Vision Model for blinded + embedded Objects.

- **[-n number_of_scales]**
Number of scales used in the multiresolution transform. Default is 5.
- **[-m type_of_noise]**
Description in section 4.2.
- **[-g SigmaNoise]**
SigmaNoise = noise standard deviation. Default is automatically estimated.
- **[-c gain,sigma,mean]**
Description in section 3.3.2.

- **[-s NSigma]**
Thresholding at $N\sigma * \text{SigmaNoise}$. Default is 3.
- **[-E Epsilon]**
 $\text{Epsilon} = \text{precision for computing thresholds}$. (Only used in the case of Poisson noise with few events). Default is 1e-03.
- **[-e minimum_of_events]**
Minimum number of events for a detection. Default is 4.
For Poisson noise with few events only (-m 9).
- **[-S SizeBlock]**
Size of the blocks used for local variance estimation. Default is 7.
- **[-N NiterSigmaClip]**
Iteration number used for local variance estimation. Default is 1.
- **[-F first_detection_scale]**
First scale used for the detection. Default is 1.
- **[-R RMS_Map_File_Name]**
RMS Map. If this option is set, the noise model is automatically fixed to:
Non-stationary additive noise
- **[-L last_detection_scale]**
Last scale used for the detection.
- **[-i number_of_iterations]**
Iteration number per object reconstruction. Default is 10.
- **[-u object_reconstruction_error]**
Default is 1e-5.
- **[-k]**
Keep isolated objects. Default is no.
- **[-K]**
Keep objects which touch the border. Default is no.
- **[-A FluxMult]**
Flux in TeX tables are multiplied by $FluxMul$. Default is 1.
- **[-w writing_parameter]**
 1. Write each object separately in an image.
The image file name of the object will be:

ima_obj_xx_yy.fits

2. Two synthetic images
 $xx_ellips.fits$: an ellipse is drawn around each object
 $xx_simu.fits$: image created only from the morphological parameters
3. equivalent to 1 and 2 together

- **[-U]**
Sub-segmentation. If two objects are close, and detected as a single object, sub-segmentation will try to separate them (deblending) into two objects.
- **[-p]**
Detect also negative structures. Default is no.
- **[-q]**
Define the root of an object from the maximum position and its value.

- **[-d DistMax]**

Maximum distance between two max positions of the same object at two successive scales.
Default is 1.

- **[-o Sub-object analysis]**

If set, sub-objects are also analysed. Several files are created:

- name_out_sub_obj.ps: a Postscript file giving the position of all sub-objects in the image.
- name_out_sub.mes: ASCII file describing all detected sub-objects. The syntax is the same as for the object ASCII file.
- name_out_sub.tex: LaTeX file which contains a table describing all detected sub-objects. The syntax is the same as for the object TeX file.
- name_out_subobj_radec: sub-object coordinates (only if “-C” option is set).

- **[-D]**

Perform a deconvolution. Default is no.

- **[-P PsfFileName]**

PSF file name.

- **[-f Fwhm]**

Full Width at Half Maximum.

- **[-O PSF_Sampling]**

PSF over-sampling value.

- **[-a BgrMethod]**

Aperture photometry:

1. Aperture photometry using a background image model
2. Aperture photometry using an estimated background image
3. Aperture photometry using a constant background

Default is no aperture photometry.

- **[-B BgrFileName]**

Background image file name.

- **[-G BgrValue]**

Constant background value. Default is: 0.

- **[-b BGR_Size]**

Background image size for automatic background estimation. Default is 16.

- **[-l KSigmaAperture]**

Aperture photometry size parameter. Default is 3.

- **[-M object_reconstruction_method]**

1. reconstruction from the fixed step gradient method
2. reconstruction from the optimum step gradient method
3. reconstruction from the conjugate gradient method
4. reconstruction using the PSF

Default is: reconstruction from the conjugate gradient method

- **[-C RADEC_Table_Order]**

1. TeX table ordered by object number.

2. TeX table ordered by the right ascension.
3. TeX table ordered by object SNR.

A TeX table of name **name_out_radec.tex** is created, which contains the object number, the right ascension (in degrees and in HH MN SEC), the declination in degrees and in DEG MN SEC), the flux, and the SNR. The object order depends on the *RADEC_Table_Order* parameter.

- [-v]
Verbose. Default is no.

6.4 Examples and Strategies

- `mr_detect -v -w 2 field_g10.fits detect_field`

Figure 6.2 presents the result of applying such a treatment. The input image (top right) has been obtained by adding Gaussian noise to a simulated image (top left). The output image shows that all objects (even very faint objects) have been detected. The file “detect_field.tex” contains Table 6.4.1, and can be compiled by the simple command “`latex detect_field`”.

- `mr_detect -P PsfFileName field_g10.fits detect_field`

Apply the detection with the same parameters, but using the PSF.

- `mr_detect -D -P PsfFileName field_g10.fits detect_field`

This time, a deconvolution is performed on each extracted object.

- `mr_detect -D -P -u 0 -i 20 PsfFileName field_g10.fits detect_field`

Force the number of iterations for each object reconstruction to be equal to 20.

- `mr_detect -v -m 2 -n 7 field_g10.fits detect_field`

Assume Poisson noise, and use more wavelet scales.

- `mr_detect -t 5 -q field_g10.fits detect_field`

The fifth transform type is used. This means that the reconstruction is not iterative (it runs faster). When two objects are blended, it may not reconstruct them as well as with other transforms. The Postscript file “detect_field_obj.ps” (see Figure 6.3) shows the position of each object in the map.

6.4.1 Choice of multiscale transform

Linear versus nonlinear transform

Five multiscale transforms are available. The first three are wavelet transforms, while the last two are nonlinear transform. Wavelet transforms should normally be preferred to nonlinear transforms, but in some cases, non-linear transforms can be useful. The main difference between the two types of transform is that nonlinear transforms allow us to have very fast object reconstruction, without iterating. But the separation between two blended objects is better when we iterate. The PSF can also not be used when using nonlinear methods, and the consequence is that the photometry may be systematically underestimated.

Decimation or not?

The default transform (transform 1) produces cubes, while transforms 2 and 5 produce half pyramidal data structures, and 3 and 4 pyramidal data structures. It is clear that in using pyramidal transforms we save computation time and memory space, but the quality is less good than with the two other sets of transforms. Half pyramidal transforms may be a good compromise between quality and both computation time and memory.

Photometry

By default, mr_detect estimates the object photometry without any information about the Point Spread Function (PSF). If the PSF presents a tail, it may not be detected in the wavelet space and the photometry may have a systematic error which depends on the PSF and on the flux of the source. If objects are sparse, an aperture photometry will resolve this problem (“-a” option). In the case where the PSF is known, a more accuracy photometry can also be obtained using the “-P” option. The PSF knowledge can be used for the object reconstruction, and the tail arround the objects is correctly restored. This approach produces normally goor results, even for crowded fields, where an aperture photometry produces poor results.

Object	x	y	σ_x	σ_y	θ	I_{max}	Flux	SNR_WaveCoef
3-1	166.78	171.85	3.90	3.54	6.4	64.62	3155.54	26.88
3-2	167.23	47.96	2.16	1.59	23.1	26.30	533.53	24.34
3-3	230.21	83.97	4.34	4.78	-14.6	84.77	6107.79	10.59
3-4	129.85	88.28	4.47	6.89	15.0	81.99	8839.90	22.36
3-5	63.78	185.28	5.49	2.85	2.4	91.62	6830.69	32.21
2-1	119.96	17.71	1.93	3.12	-18.0	85.43	2910.22	14.89
2-2	129.27	29.65	4.54	2.33	-5.5	398.30	16277.82	60.05
2-3	154.99	145.35	2.66	2.91	-35.5	181.33	6070.46	31.37
2-4	16.67	160.69	2.78	2.40	-11.6	55.63	1562.33	8.37
2-5	103.16	75.34	2.98	2.12	-11.6	47.76	1368.29	12.32
2-6	114.21	231.37	2.35	3.13	-41.1	72.64	2538.67	12.43
2-7	233.17	101.44	2.91	3.02	0.5	79.43	3552.28	8.54
2-8	101.13	98.60	2.80	3.51	42.9	79.17	3228.94	9.36
2-9	231.91	127.28	3.15	2.74	5.7	56.52	2015.41	8.71
2-10	36.05	153.08	2.33	3.03	-39.5	32.58	1162.67	15.26
2-11	227.54	207.51	3.36	2.98	-0.8	54.97	2549.64	12.13
2-12	152.97	209.01	3.54	3.09	35.1	95.40	4358.75	10.36
1-1	176.37	131.80	1.48	1.37	2.4	133.07	1653.93	5.66
1-2	154.00	146.00	1.01	1.01	0.0	122.96	1004.52	13.05
1-3	13.92	145.08	1.38	1.42	-9.0	299.54	3488.15	5.54

Table 6.1: Detection table.

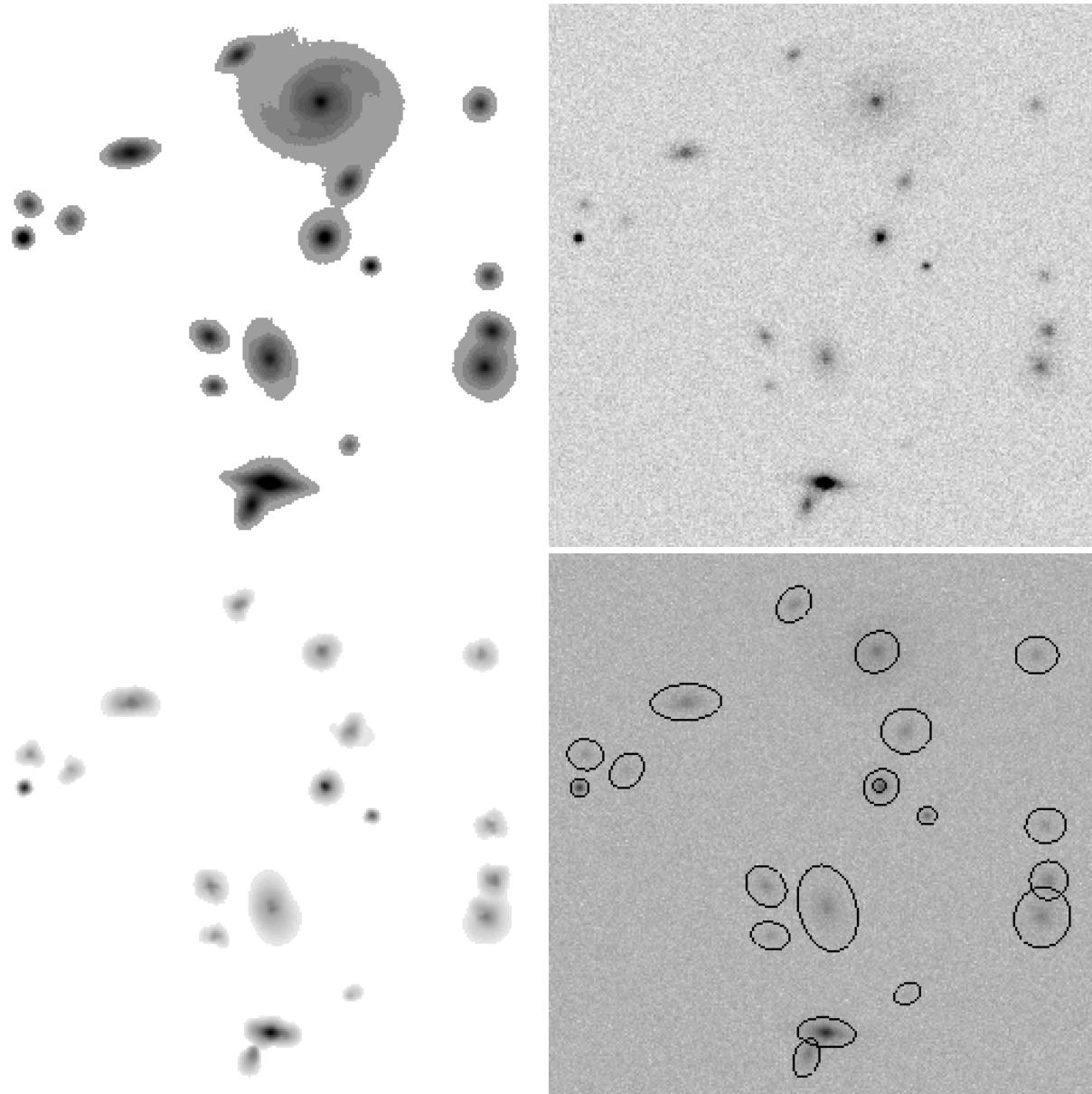


Figure 6.2: Top left, original simulated image (stars + galaxies). Top right, same image plus Gaussian noise. Bottom left, output image produced by *mr_detect*. Bottom right, noise image, and ellipses overplotted.

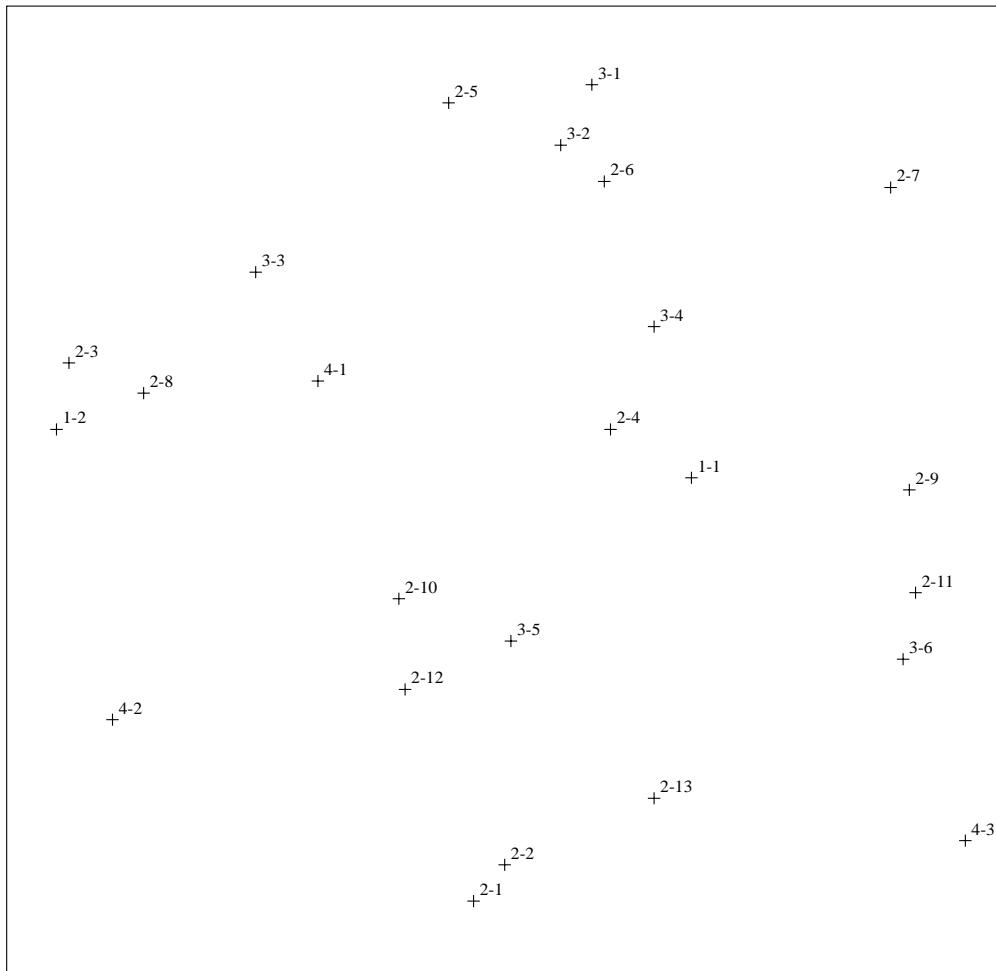


Figure 6.3: Position of each object in the image.

Chapter 7

MR/1 Geometric Registration

7.1 Introduction

Image registration is a procedure which determines the best spatial fit between two or more images that overlap the same scene, and were acquired at the same or at a different time, by identical or different sensors. Thus, registration is required for processing a new set of data in such a way that its image under an appropriate transform is in a proper geometrical relationship with the previous set of data.

Several digital techniques have been used for automatic registration of images such as cross-correlation, normal cross-correlation and minimum distance criteria. The advantage of the wavelet transform is that it produces both spatial and frequency domain information which allow the study of the image by frequency bands [56, 57, 55].

An automatic image registration procedure can be helpful for several applications:

1. Comparison between two images obtained at the same wavelength.
2. Comparison between two images obtained at different wavelengths.
3. Pixel field of view distortion estimation.

The geometrical correction is usually performed by three operations:

- The measure of a set of well-defined ground control points (GCPs), which are features well located both in the input image and in the reference image.
- The determination of the warping or deformation model, by specifying a mathematical deformation model defining the relation between the coordinates (x, y) and (X, Y) in the reference and input image respectively.
- The construction of the corrected image by output-to-input mapping.

The main difficulty lies in the automated localization of the corresponding GCPs, since the accuracy of their determination will affect the overall quality of the registration. In fact, there are always ambiguities in matching two sets of points, as a given point corresponds to a small region D , which takes into account the prior geometric uncertainty between the two images and many objects could be contained in this region.

One property of the wavelet transform is to have a sampling step proportional to the scale. When we compare the images in the wavelet transform space, we can choose a scale corresponding to the size of the region D , so that no more than one object can be detected in this area, and the matching is done automatically.

7.2 Deformation model

Geometric correction requires a spatial transformation to invert an unknown distortion function. A general model for characterizing misregistration between two sets of remotely sensed data is a pair of bivariate polynomials of the form:

$$x_i = \sum_{p=0}^N \sum_{q=0}^{N-p} a_{pq} X_i^p Y_i^q = Q(X_i, Y_i)$$

$$y_i = \sum_{p=0}^N \sum_{q=0}^{N-p} b_{pq} X_i^p Y_i^q = R(X_i, Y_i)$$

where (X_i, Y_i) are the coordinates of the i^{th} GCP in the reference image, (x_i, y_i) the corresponding GCP in the input image and N is the degree of the polynomial.

Usually, for images taken under the same *imaging direction*, polynomials of degree one or two are sufficient as they can model most of the usual deformations like shift, scale, skew, perspective and rotation (see Table 3.1).

We then compute the unknown parameters $((N + 1)(N + 2)/2$ for each polynomial) using the least mean square estimator.

Shift	$x = a_0 + X$ $y = b_0 + Y$
Scale	$x = a_1 X$ $y = b_2 Y$
Skew	$x = X + a_2 Y$ $y = Y$
Perspective	$x = a_3 X Y$ $y = Y$
Rotation	$x = \cos \theta X + \sin \theta Y$ $y = -\sin \theta X + \cos \theta Y$

Table 7.1: Some common deformations.

7.3 Image registration: mr_fusion

Program *mr_fusion* performs the geometrical registration of two images having the same size, and same resolution. Four deformation models are available (“-d” option). The program may fail to register the image when not enough control points are detected. In this case, the user can try another deformation model which may be more adapted to his data, or modify the noise model parameters. Here the noise model is only used for structure detection. If the image

contains strong features, the threshold parameter “-s” can be set at a greater value. Hence, the registration will be performed only from the strongest feature of the image. The number of scales is also very important. Indeed, the maximum distance between two pixels of the same point in the two images must be less than or equal to the size of the wavelet at the last scale. The number of scales can be fixed either using directly the “-n” option, or using the “-D” option. By choosing the latter, the user gives the maximum distance between two pixels of the same point, and the program calculates automatically the correct number of scales.

USAGE: `mr_fusion option image_ref image_in image_out`

Options are:

- **[-p]**
Poisson Noise. Default is no Poisson component (just Gaussian).
- **[-g SigmaNoise]**
SigmaNoise = Gaussian noise standard deviation. Default is automatically estimated.
- **[-c gain,sigma,mean]**
See section 3.3.2.
- **[-n number_of_scales]**
Number of scales used in the multiresolution transform. Default is 4.
- **[-s NSigma]**
Thresholding at NSigma * SigmaNoise. Default is 5.
- **[-r res_min]**
Minimum resolution for the reconstruction. The registration procedure is stopped at scale res_min and the resulting deformation model is used to register the input image. Default value is 1.
- **[-D dist_max]**
Maximum estimated distance between two identical points in both images. This value is used to estimate the number of scales for the wavelet transform.
- **[-i Interpolation type]**
Type of interpolation:
 - 0: Zero order interpolation – nearest neighbor.
 - 1: First order interpolation – bilinear.
 - 2: Second order interpolation – bicubic.
 Default is 2.

- **[-d DeforModel]**
Type of registration deformation model:
The type of polynomial model used for the geometrical registration. Three types are available:

- 0: Polynomial of the first order of type I:

$$x' = aX - bY + c_x \quad (7.1)$$

$$y' = bX + aY + c_y \quad (7.2)$$

- 1: Polynomial of the first order of type II:

$$x' = aX + bY + c \quad (7.3)$$

$$y' = dX + eY + f \quad (7.4)$$

- 2: Polynomial of the second order:

$$x' = aX^2 + bY^2 + cXY + dX + eY + f \quad (7.5)$$

$$y' = gX^2 + hY^2 + iXY + jX + kY + l \quad (7.6)$$

- 3: Polynomial of the third order.

Default is 1.

- **[-o]**

Manual Options specifications:

A few options are provided in order to have more control on the procedure. Using manual options, the following parameters can be fixed by the user for each scale:

- Matching distance:

The distance used for the matching procedure. The procedure looks for each control point candidate in the reference image which is the corresponding control point candidate in the input image within a radius of *Matching distance*.

- Threshold level:

The threshold level used for thresholding the wavelet transform.

- Type of registration deformation model (same as option -d).

- Type of interpolation (same as option -i).

The available manual options are the following:

- 0: Everything is taken care of by the program.

- 1: The matching distance is specified manually for each resolution.

- 2: The threshold level is specified manually for each resolution and for both the reference image and the input image.

- 3: The type of deformation model is specified manually for each resolution.

- 4: The matching distance, the Threshold level and the Type of deformation model are specified manually for each resolution.

- 5: The matching distance, the threshold level, the type of deformation model and the type of interpolation are specified manually for each resolution.

The default is none (0).

- **[-w]**

The following files are written to disk:

- deform_model.txt: contains the calculated coefficients of the deformation model, allowing us to calculate the coordinates in the second image of a given point from its coordinates in the reference image.

- scale_j_control_points.dat: contains the set of control points for each scale. The first line contains the number of control points, and all other lines the values Xr,Yr,Xi,Yi, where Xr and Yr are the coordinates of the control point in the reference image (origin is (0,0)), and Xi and Yi are the coordinates of the corresponding control point in the input image (second image).

- xx_grill_in: an artificial image which contains a “grilled” image.

- xx_grill_out: the resulting image after applying the deformation model to the artificial one.

The default is none.

A strong distortion was applied to the galaxy NGC2997 (see Figure 3.3). A synthetic image was made by shifting it by 5 and 10 pixels in each axis direction. Then this image was rotated by 10 degrees and Gaussian noise was added. Figure 7.1 shows the synthetic image (upper left panel), and also the difference between the original image and the synthetic one (upper right panel). Figure 7.1 (bottom left and right) shows the corrected image and the residual between the original image and the corrected image. The two images have been correctly – and automatically – registered.

Example:

- `mr_fusion -n 6 -s 10 -d 1 ngc2997.fits dec_ngc register_ima`

Register the image `dec_ngc` on `ngc2997`, with a 10 sigma detection, 6 scales, and using the first order deformation model of type 2. The result is presented in Figure 7.1.

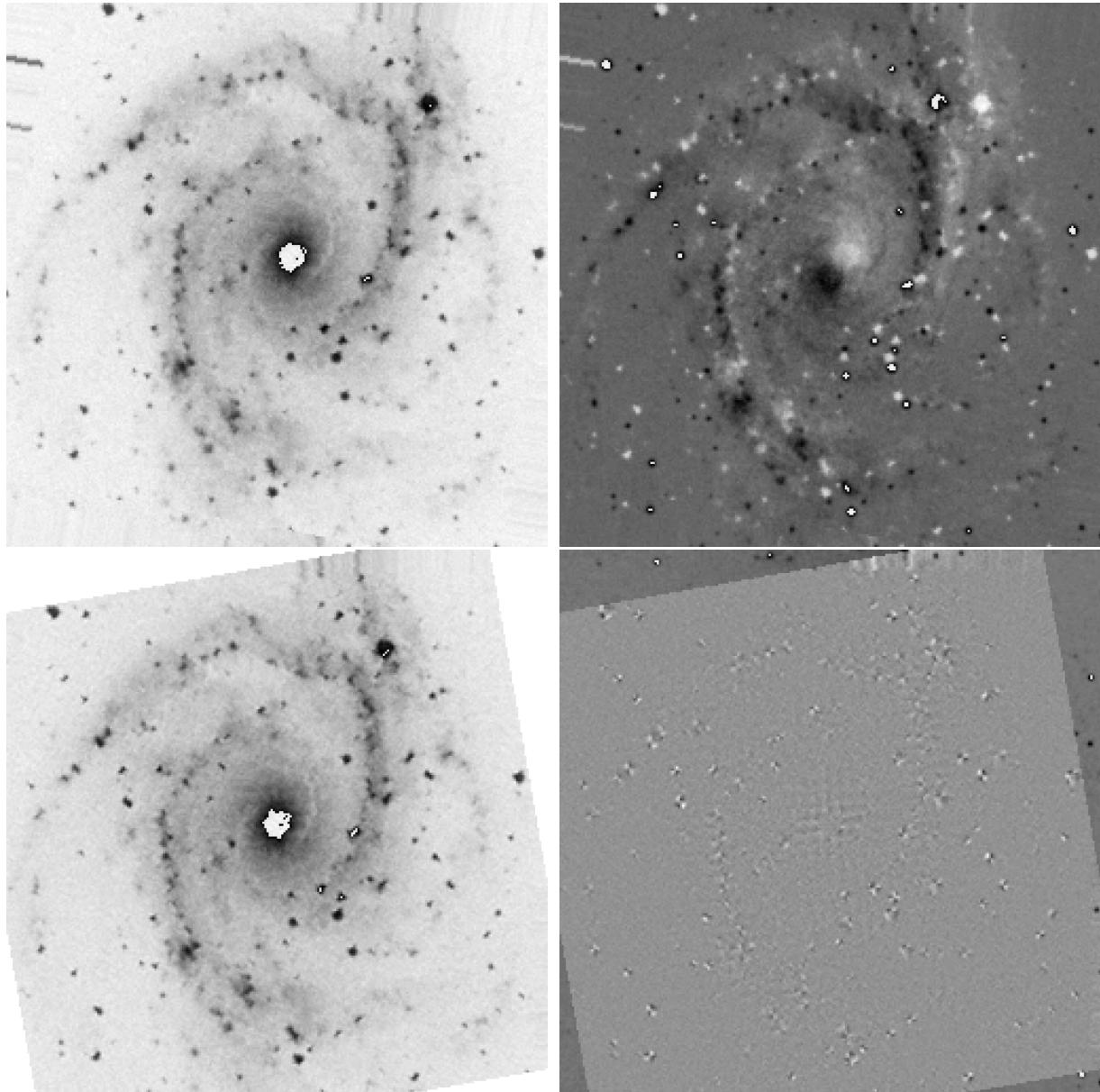


Figure 7.1: Synthetic image (upper left) and difference between the original and the synthetic image (upper right). Registered image (bottom left) and difference between NGC2997 and the registered image (bottom right).

If the deformation model is not sophisticated enough to resolve a specific problem, the result will certainly not be correct, but the user can still use the CGP points which do not depend on any model.

Chapter 8

MR/1 Edge Detection

8.1 Introduction

An edge is defined as a local variation of image intensity. Edges can be detected by the computation of a local derivative operator.

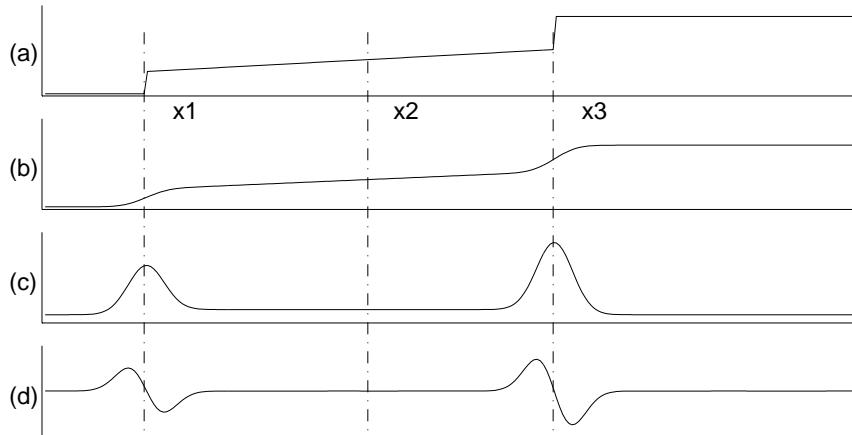


Figure 8.1: First and second derivative of $G_\sigma * f$. (a) Original signal, (b) signal convolved by a Gaussian, (c) first derivative of (b), (d) second derivative of (b).

Figure 8.1 shows how the inflection point of a signal can be found from its first and second derivative. Two methods can be used for generating first order derivative edge gradients.

8.2 First Order Derivative Edge Detection

8.2.1 Gradient

The gradient of an image f at location (x, y) , along the line normal to the edge slope, is the vector [154, 85, 96]:

$$\nabla f = \begin{bmatrix} f_x \\ f_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (8.1)$$

The spatial gradient amplitude is given by:

$$G(x, y) = \sqrt{f_x^2 + f_y^2} \quad (8.2)$$

and the gradient direction with respect to the row axis is

$$\Theta(x, y) = \arctan \frac{f_y}{f_x} \quad (8.3)$$

The first order derivative edge detection can be carried out either by using two orthogonal directions in an image or by using a set of directional derivatives.

8.2.2 Gradient mask operators

Gradient estimates can be obtained by using gradient operators of the form:

$$\begin{aligned} f_x &= f \otimes H_x \\ f_y &= f \otimes H_y \end{aligned} \quad (8.4)$$

where \otimes denotes the convolution product, and H_x and H_y are 3×3 row and column operators, called gradient masks. Table 8.1 shows the main gradient masks proposed in the literature. Pixel difference is the simplest one, which consists just of making the difference of pixels along rows and columns of the image:

$$\begin{aligned} f_x(x_m, y_n) &= f(x_m, y_n) - f(x_m - 1, y_n) \\ f_y(x_m, y_n) &= f(x_m, y_n) - f(x_m, y_n - 1) \end{aligned} \quad (8.5)$$

The Roberts gradient masks [159] are more sensitive to diagonal edges. Using these masks, the orientation must be calculated by

$$\Theta(x_m, y_n) = \frac{\pi}{4} + \arctan \left[\frac{f_y(x_m, y_n)}{f_x(x_m, y_n)} \right] \quad (8.6)$$

Prewitt [156], Sobel, and Frei-Chen [75] produce better results than the pixel difference, separated pixel difference and Roberts operator, because the mask is larger, and provides averaging of small luminance fluctuations. The Prewitt operator is more sensitive to horizontal and vertical edges than diagonal edges, and the reverse is true for the Sobel operator. The Frei-Chen edge detector has the same sensitivity for diagonal, vertical, and horizontal edges.

8.2.3 Compass operators

Compass operators measure gradients in a selected number of directions. The directions are $\Theta_k = k\frac{\pi}{4}$, $k = 0, \dots, 7$. The edge template gradient is defined as:

$$G(x_m, y_n) = \max_{k=0}^7 | f(x_m, y_n) \otimes H_k(x_m, y_n) | \quad (8.7)$$

Table 8.2 shows the principal template gradient operators.

Operator	H_x	H_y	Scale factor
Pixel difference	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	1
Separated pixel difference	$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	1
Roberts	$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	1
Prewitt	$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$	1
Sobel	$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$	$\frac{1}{4}$
Fei-Chen	$\begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & \sqrt{2} \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$	$\frac{1}{2+\sqrt{2}}$

Table 8.1: Gradient edge detector masks.

Gradient direction	Prewitt compass gradient	Kirsch	Robinson 3-level	Robinson 5-level
East	$\begin{bmatrix} 1 & 1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & -1 \end{bmatrix}$	$\begin{bmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$
Northeast	$\begin{bmatrix} 1 & -1 & -1 \\ 1 & -2 & -1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & -1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$
North	$\begin{bmatrix} -1 & -1 & -1 \\ 1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Northwest	$\begin{bmatrix} -1 & -1 & 1 \\ -1 & -2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$
West	$\begin{bmatrix} -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} -3 & 0 & 5 \\ -3 & -3 & 5 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
Southwest	$\begin{bmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix}$	$\begin{bmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$
South	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$
Southeast	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & -1 \\ 1 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix}$	$\begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}$
Scale factor	$\frac{1}{5}$	$\frac{1}{15}$	$\frac{1}{3}$	$\frac{1}{4}$

Table 8.2: Template gradients.

8.2.4 Derivative of Gaussian

The previous methods are relatively sensitive to the noise. A solution could be to extend the window size of the gradient mask operators. Another approach is to use the derivative of the convolution of the image by a Gaussian. The derivative of a Gaussian (DroG) operator is

$$\begin{aligned}\nabla(g \otimes f) &= \frac{\partial(g \otimes f)}{\partial x} + \frac{\partial(g \otimes f)}{\partial y} \\ &= f_x + f_y\end{aligned}\quad (8.8)$$

with $g = \exp -\frac{x^2+y^2}{2\sigma^2}$. Partial derivatives of the Gaussian function are

$$\begin{aligned}g_x(x, y) &= \frac{\partial g}{\partial x} = -\frac{x}{\sigma^2} \exp -\frac{x^2+y^2}{2\sigma^2} \\ g_y(x, y) &= \frac{\partial g}{\partial y} = -\frac{y}{\sigma^2} \exp -\frac{x^2+y^2}{2\sigma^2}\end{aligned}\quad (8.9)$$

The filters are separable so we have

$$\begin{aligned}g_x(x, y) &= g_x(x) * g(y) \\ g_y(x, y) &= g_y(y) * g(x)\end{aligned}\quad (8.10)$$

Then

$$\begin{aligned}f_x &= g_x(x) \otimes g(y) \otimes f \\ f_y &= g_y(y) \otimes g(x) \otimes f\end{aligned}\quad (8.11)$$

8.2.5 Thinning the contour

From the gradient map, we may want to consider only pixels which belong to the contour. This can be done by looking for each pixel in the direction of gradient. For each point P0 in the gradient map, we determine the two adjacent pixels P1,P2 in the direction orthogonal to the gradient. If P0 is not a maximum in this direction (i.e. $P0 < P1$, or $P0 < P2$), then we threshold P0 to zero.

8.3 Second Order Derivative Edge Detection

Second derivative operators allow us to accentuate the edges. The most frequently used operator is the Laplacian one, defined by

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}\quad (8.12)$$

Table 8.3 gives three discrete approximation of this operators.

Marr and Hildreth [127] have proposed the Laplacian of Gaussian (LoG) edge detector operator. It is defined as

$$L(x, y) = \frac{1}{\pi s^4} \left[1 - \frac{x^2 + y^2}{2s^2} \right] \exp \left(-\frac{x^2 + y^2}{2s^2} \right)\quad (8.13)$$

where σ controls the width of the Gaussian kernel.

Zero-crossings of a given image f convolved with L give its edge locations.

A simple algorithm for zero-crossings is:

Laplacian 1	Laplacian 2	Laplacian 3
$\frac{1}{4} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\frac{1}{8} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	$\frac{1}{8} \begin{bmatrix} -1 & -2 & -1 \\ -2 & 4 & -2 \\ -1 & -2 & -1 \end{bmatrix}$

Table 8.3: Laplacian operators.

1. For all pixels i,j do
2. $\text{ZeroCross}(i,j) = 0$
3. $P_0 = G(i,j); P_1 = G(i,j-1); P_2 = G(i-1,j); P_3 = G(i-1,j-1)$
4. If $(P_0 * P_1 < 0)$ or $(P_0 * P_2 < 0)$ or $(P_0 * P_3 < 0)$ then $\text{ZeroCross}(i,j) = 1$

8.4 Edge Detection Program: im_edge

The program *im_edge* detects the edges in an image by the methods previously described.

USAGE: `im_edge option file_name_in file_name_out`

where options are:

- **[-M edge_detection_method]**

1. Derivative of a Gaussian (DroG)
2. Pixel difference
3. Separated pixel difference
4. Sobel
5. Prewitt
6. Roberts
7. Frei Chen
8. Laplacian: filter 1
9. Laplacian: filter 2
10. Laplacian: filter 3
11. Marr and Hildreth: Laplacian of Gaussian (LoG)
12. Prewitt compass gradient
13. Kirsch
14. Robinson 3-level
15. Robinson 5-level

Default is Sobel method

- **[-k]**

Select local maxima in gradient direction for first derivative methods, and zero-crossings for second derivative methods.

Default is not to do these operations.

- **[-t ThresholdValue]**
Threshold all values in the edge map lower than *ThresholdValue*.
- **[-S Sigma]**
Scale parameter. Only used for DroG and LoG methods.
Default value is $\frac{1}{\sqrt{3}}$.

8.5 Wavelets and Edge Detection

The LoG operator, also called the Mexican hat, is a well-known wavelet function (even if wavelets did not exist when the LoG operator was proposed!). Furthermore, a scale parameter (σ) in this method leads to a multiscale approach when σ is varying. The advantage of using the wavelet framework is the existence of very fast WT, such as the à trous algorithm, which furnishes us with a way to get the edges directly at all dyadic scales, even for large scales.

8.5.1 Multiscale first derivative

Generalizing the concept of multiscale edge detection, Mallat [122, 123, 121] showed that the DroG operators can be easily associated with a wavelet transform. If $\phi(x)$ is a smoothing function (i.e. $\int \phi(x)dx = 1$), converges to zero at infinity, and is differentiable, we denote

$$\psi^1(x, y) = \frac{d\phi(x, y)}{dx} \quad \text{and} \quad \psi^2(x, y) = \frac{d\phi(x, y)}{dy} \quad (8.14)$$

By definition, ψ^x and ψ^y are wavelets (their integral is equal to zero). The local extrema of the wavelet coefficients using ψ^{x_1}, ψ^{y_1} correspond to the inflection points of $f * \phi_s$ (with $\phi_s = \frac{1}{s}\phi(\frac{x}{s})$).

Using the directional à trous algorithm, sometimes also called dyadic wavelet transform, we have at each scale j and at pixel location (x, y) two wavelet coefficients $w_{j,x}, w_{j,y}$. The modulus of the gradient is then defined by

$$G_j(x, y) = \sqrt{w_{j,x}^2 + w_{j,y}^2} \quad (8.15)$$

and the directional angle θ_j is

$$\theta_j(x, y) = \begin{cases} \arctan\left(\frac{w_{j,y}}{w_{j,x}}\right) & \text{if } w_{j,x} \geq 0 \\ \pi - \arctan\left(\frac{w_{j,y}}{w_{j,x}}\right) & \text{if } w_{j,x} < 0 \end{cases} \quad (8.16)$$

Multiscale edge points, also called modulus maxima, are points where the modulus is locally maximum with respect to its neighbors along the direction θ_j .

A similar approach can be developed for the LoG operator [122].

8.5.2 Image reconstruction from its multiscale edges

An image can be reconstructed (approximately) from its multiscale edges [121] using an iterative algorithm. It does not converge exactly towards the image, but in practice the error is very small. The algorithm consists of searching for an image h such that its wavelet transform has the same modulus maxima (i.e. same number of modulus maxima, same positions, and same amplitudes) as those of the original image f . Denoting as $w_j(X)$ the wavelet coefficients of an image X at a scale j , and $w_j^m(X)$ its modulus maxima, we require that $w_j^m(h) = w_j^m(f)$. The algorithm is the following:

1. Set $w_j^m(h)$ to zero.
2. Loop:
3. Calculate the difference $w_j^m(f) - w_j^m(h)$, and interpolate between the maxima: we get w_j^d .
4. Update $w_j(h)$: $w_j(h) = w_j(h) + w_j^d$.

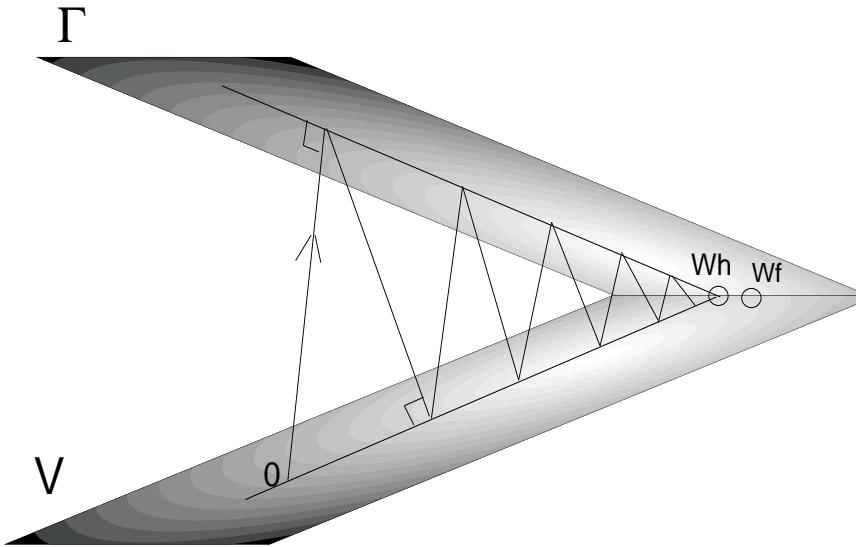


Figure 8.2: Approximation of the wavelet transform of f .

5. Assign in $w_j(h)$ the correct values at maxima positions.
6. Reconstruct h from $w_j(h)$
7. Do a wavelet transform of h : $w_j(h)$ is then updated.
8. Goto 2
9. Assign in $w_j(h)$ the correct values at maxima positions.
10. Reconstruct h from $w_j(h)$.

The algorithm converges quickly to the solution after a few iterations. The interpolation [122] can be viewed as a projection on an affine space Γ , and the inverse and forward wavelet transform as a projection on the space V of all possible wavelet transform signals. The two projections are visualized in Figure 8.5.2 (W_h and W_f are respectively the wavelet transform of h and f).

8.6 Multiscale Edge Detection Program

8.6.1 First derivative: mr_edge

The program *mr_edge* detects the edges in an image by the methods previously described. The wavelet transform used is the Mallat dyadic wavelet transform. The output file is a multiresolution file (“.mr”). For each scale, two bands are stored, one the maxima map and the second the gradient angle map.

USAGE: **mr_edge option file_name_in file_name_out**

where options are:

- **[-n number_of_scales]**
Number of scales used in the multiresolution transform. Default is 4.

Examples:

- `mr_edge image.d med.mr`
Build the multiscale edge detection file.
- `mr_extract -B -s 1 med.mr mod1`
Create an image which contains the detected edge map (maxima) of the first scale.
- `mr_extract -B -s 2 med.mr ang1`
Create an image which contains the gradient angle map of the previous edge map.
- `mr_extract -B -s 5 med.mr mod3`
- `mr_extract -B -s 6 med.mr ang3`
Ditto for scale 3.

8.6.2 Second derivative: mr_at_edge

The program *mr_at_edge* detects the edges in an image by the methods previously described. The wavelet transform used is the à trous algorithm. Only zero crossings of each scale are kept (i.e. not thresholded). The output file is a multiresolution file (“.mr”).

USAGE: mr_at_edge option file_name_in file_name_out

where options are:

- **[-n number_of_scales]**
Number of scales used in the multiresolution transform. Default is 4.

Examples:

- `mr_at_edge image.d med.mr`
Build the multiscale edge detection file.
- `mr_extract -s 1 med.mr zero1`
Create an image which contains the detected edge map of the first scale.
- `mr_extract -s 2 med.mr zero2`
Ditto for scale 2.

8.6.3 Image reconstruction: mr_rec_edge

The program *mr_rec_edge* reconstructs an image from its multiscale edges. The multiscale edge file must have been obtained from the *mr_edge* program.

USAGE: mr_rec_edge option file_name_in file_name_out

where options are:

- **[-i number_of_iterations]**
Number of iterations. Default is 10.

8.7 Contrast Enhancement

8.7.1 Introduction

Because some features are hardly detectable by eyes in a image, we often transform it before visualization. Histogram equalization is certainly one the most well known method for contrast enhancement. Such an approach is general useful for images with a poor intensity distribution. As edges play a fundamental role in image understanding, a way to enhance the contrast is to enhance the edges. For example, we can add to the original image its Laplacian ($I' = I + \gamma\Delta I$, where γ is a parameter). Only features at the finest scale are enhanced (linearly). For a high γ value, only the high frequencies are visible. Multiscale edge enhancement [208] can be seen as a generalization of this approach to all resolution levels. Images with a high dynamic range are also difficult to analyze. For example, astronomers generally visualize their images using a logarithmic transformation. We see in the next section that wavelet can also be used to compress the dynamic range at all scales, and therefore allows us to clearly see some very faint features.

8.7.2 Multiscale Edge Enhancement

Gray Images

Velde has proposed the following algorithm [208]:

1. The image L is mapped nonlinearly according to:

$$L(i) \rightarrow L(i)^{1-q} 100^q \quad (8.17)$$

2. The L image is decomposed into a multiscale gradient pyramid by using the dyadic wavelet transform (two directions per scale). The gradient at the scale j , the pixel position i is calculated by: $G_j(i) = \sqrt{(w_j^{(h)}(i))^2 + (w_j^{(v)}(i))^2}$ where $w_j^{(h)}$ and $w_j^{(v)}$ are the wavelet coefficients in both vertical and diagonal directions at pixel position i .
3. The two wavelet coefficients at scale j and at position i are multiplied by $y(G_j(i))$, where y is defined by:

$$\begin{aligned} y(x) &= (\frac{m}{c})^p \text{ if } |x| < c \\ y(x) &= (\frac{m}{|x|})^p \text{ if } c \leq |x| < m \\ y(x) &= 1 \text{ if } |x| \geq m \end{aligned} \quad (8.18)$$

4. The \tilde{L} image is reconstructed from the modified wavelet coefficients.
5. the \tilde{L} image is mapped nonlinearly according to:

$$\tilde{L}(i) \rightarrow \tilde{L}(i)^{\frac{1}{1-q}} 100^{-\frac{q}{1-q}} \quad (8.19)$$

Four parameters are needed p, q, m, c . p determines the degree of non-linearity in the nonlinear rescaling of the luminance, and must be in $]0, 1[$. q must be in $[-0.5, 0.5]$. When $q > 0$, then darker parts are less enhanced than the lighter parts. When $q < 0$, then the dark parts are more enhanced than lighter parts. Coefficients larger than m are not modified by the algorithm. The c parameter corresponds to the noise level.

Figure 8.3 shows the modified wavelet coefficients versus the original wavelet coefficients for a given set parameters ($m=30, c=3, p=0.5$, and $q=0$).

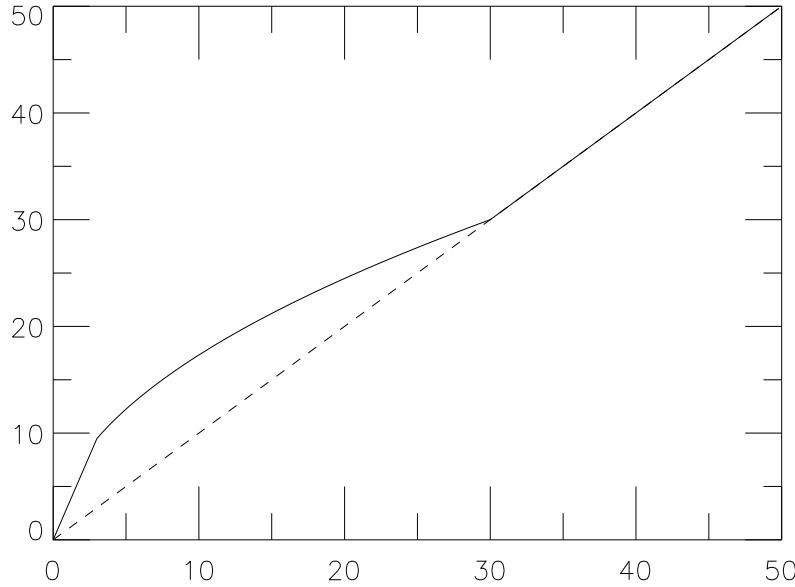


Figure 8.3: Enhanced coefficients versus original coefficients. Parameters are $m=30, c=3, p=0.5$, and $q=0$.

8.7.3 The LOG-Wavelet Representation of Gray Images

By using the à trous wavelet transform algorithm, an image I can be defined as the sum of its J wavelet scales and the last smooth array:

$$I(x, y) = c_J(x, y) + \sum_{j=1}^J w_j(x, y) \quad (8.20)$$

where the first term on the right is the last smoothed array, and w denotes a wavelet scale. See [191] more details about this algorithm. The wavelet-log representations consists in replacing $w_j(x, y)$ by $\log(|w_j(x, y)|)$:

$$I_w(x, y) = \log(c_J(x, y)) + \sum_{j=1}^J \operatorname{sgn}(w_j(x, y)) \log(|w_j(x, y)|) \quad (8.21)$$

Figure 8.4 left shows the logarithm of Hale-Bopp comet image and right its wavelet log representation. The jets clearly appears in the last representation.

8.7.4 Contrast Enhancement Program: mr_contrast

The program *mr_contrast* enhances the contrast of a gray image. Several methods are available.

USAGE: mr_contrast option in_image out_image

where options are:

- **[-m contrast_enhancement_method]**

1. Histogram Equalization

2. Wavelet Coefficients Enhancement
3. Wavelet-Log function: $f(w) = \log(|w| + L)$
4. Wavelet-Log function: $f(w) = \text{sgn}(w).\log(|w| + L)$
5. K-Sigma clipping.
6. Add to the image its Laplacian.

Default is Wavelet Coefficients Enhancement.

- **[-n number_of_scales]**

Number of scales used in the wavelet transform. Default is 4.

- **[-M M_parameter]**

M Parameter. Only used if “-e” option is set. Coefficients larger than M are not enhanced. Default is 100.

- **[-P P_parameter]**

P Parameter. Only used if “-e” option is set. P must be in the interval $]0, 1[$. Default is 0.5.

- **[-Q Q_parameter]**

Q Parameter. Only used if “-e” option is set. Q must be in the interval $[-0.5, 0.5]$. When $Q > 0$, darker part are less enhanced than lighter part. When $Q < 0$, darker part are more enhanced than lighter part. Default is 0.

- **[-C C_parameter]**

C Parameter. Only used if “-e” option is set. Default is 0.

- **[-K ClippingValue]**

Clipping value. Default is 3.

- **[-L Param]**

Parameter for for the Laplacian or the log method. Default is 0.1.

Examples:

- mr_contrast image.fits image_out.fits
Enhance the contrast by multiscale edge method.
- mr_contrast -n6 -m4 image.fits image_out.fits
Enhance the contrast by wavelet log representation using six resolution levels.

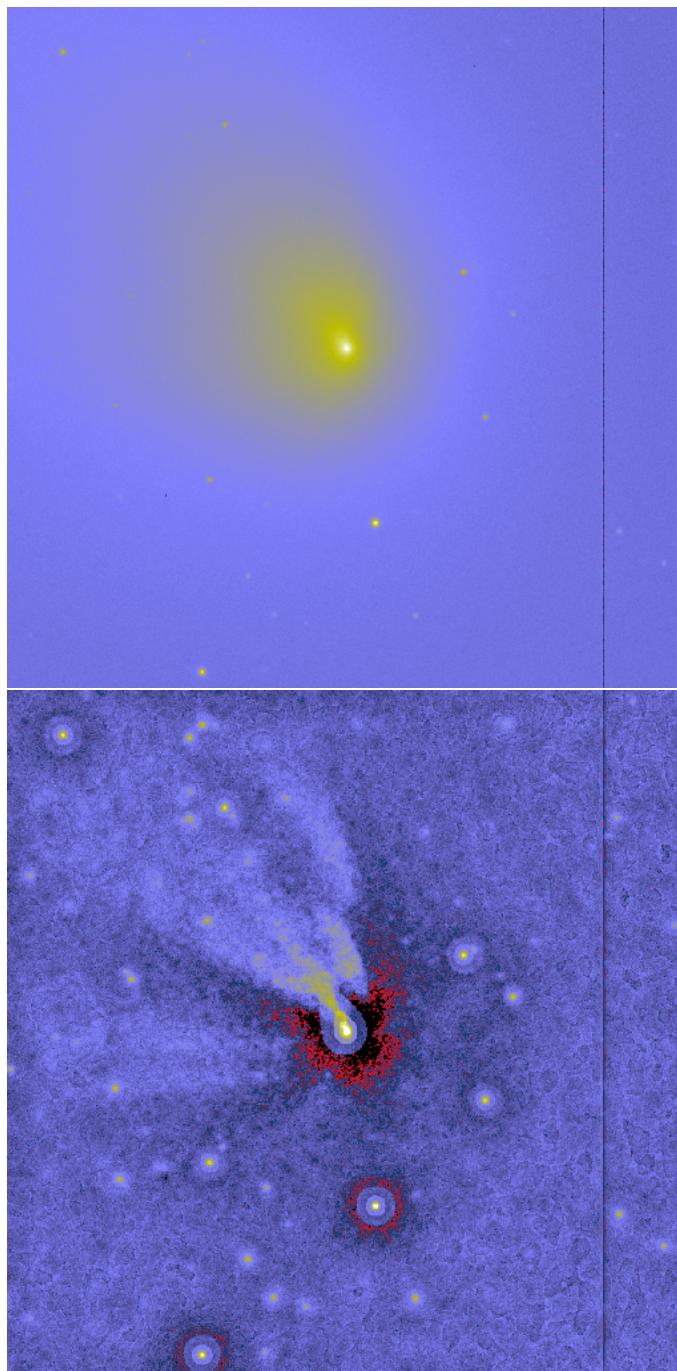


Figure 8.4: Hale-Bopp Comet image and its wavelet log representation.

Chapter 9

MR/1 1D Data

9.1 1D Tools

9.1.1 Conversion: im1d_convert

Program *im1d_convert* converts a 1D signal from one data format to another one. Currently supported signal formats are the ascii format, the FITS format, and Excel format. Suffixes for these formats are respectively “.dat”, “.fits”, and “.csv””. The “-s” option allows the user to suppress a given number of lines at the beginning of the file. This option has an effect only for ascii and Excel input formats. The ascii format consists of a series of numbers separated by a space, a tab, or a new line.

USAGE: `im1d_convert [-s NbrLines] file_name_in file_name_out`

Examples:

- `im1d_convert sig.dat image.fits`
Converts an ascii file to FITS format.
- `im1d_convert -s 2 sig.dat image.fits`
Ditto, but the first lines are not taken into account.

9.1.2 Statistical information: im1d_info

im1d_info gives information about a signal:

- the number of pixels
- the minimum, the maximum
- the arithmetic mean: $\bar{x} = \frac{1}{N} \sum_k x_k$
- the standard deviation: $\sigma = \sqrt{\frac{1}{N} \sum_k (x_k - \bar{x})^2}$
- the flux: $F = \sum_k x_k$
- the energy: $E = \sum_k x_k^2$
- the skewness: $S = \frac{1}{N\sigma^3} \sum_k (x_k - \bar{x})^3 = \frac{1}{\sigma^3} (\bar{x}^3 - 3\bar{x}\bar{x}^2 + 2\bar{x}^3)$

- the kurtosis: $C = \frac{1}{N\sigma^4} \sum_k (x_k - \bar{x})^4 - 3 = \frac{1}{\sigma^4} (\bar{x}^4 - 4\bar{x}\bar{x}^3 + 6\bar{x}^2\bar{x}^2 - 3\bar{x}^4) - 3$
- Measure of dependence: calculate the autoregressive model which fits the data, for all orders between 1 and M , and select the AR model which minimizes the following equation:

$$J(p) = \log \sigma_{A(p)}^2 + \mathcal{P}(A(p))$$

where σ_A is the prediction error, and \mathcal{P} is a penalty function, which increases with the AR order. Examples of penalty functions are:

- AIC: $AIC = \log \sigma_{A_j}^2 + \frac{2A_j}{N}$
- AICC: $AICC = \log \sigma_{A_j}^2 + \frac{N+A_j}{N-A_j-2}$
- SIC: $SIC = \log \sigma_{A_j}^2 + \frac{A_j \log N}{N}$

When the “-a” option is set, then the autocorrelation function $\rho(h)$ is also calculated:

$$\rho(h) = \frac{\gamma(h)}{\gamma(0)} \quad (9.1)$$

where $\gamma(h)$ is the autocovariance function defined by:

$$\gamma(h) = \frac{1}{N} \sum_k (x_{k+h} - \bar{x})(x_k - \bar{x}) \quad (9.2)$$

The command line is:

USAGE: im1d_info file_name.in

where options are:

- **[-a Nbr_of_Lags]**
Calculate the autocorrelation function (AF) with a given number of lags. The output AF is saved in a file of name “autocor”. Default number of lags is 10.
- **[-O Estimation_AR_Order_Method]**

1. AIC
2. AICC
3. BIC

Default is BIC method.

- **[-M MaxAROrder]**
Maximum AR model order. Default is 10.

Examples:

- **im1d_info data.dat**
Gives information about the data.
- **im1d_info -a 10 data.dat**
Ditto, but calculates also autocorrelation function with 10 lags.

9.1.3 Tendency estimation: im1d_tend

If the signal exhibits a tendency, it may be convenient to remove it before starting the analysis. A well-known fitting method is the polynomial one. A polynomial of order p is fitted around each pixel in a window of size T (p equals 2 in this program). In order to have smooth tendency estimation, it is recommended to weight the pixels with weights from 1 to zero for pixels in the middle to the border of the window.

USAGE: `im1d_tend option in_data out_tend out_signal_no_tend`

where options are:

- `[-T WindowSize_for_tendency_estimation]`
Default is 100.
- `[-f FirstPixels]` Default is the input signal size. If this option is set, the tendency is estimated only on the first *FirstPixels* pixels.

Examples:

- `im1d_tend sig.dat tend.dat sig_out`
Remove the tendency in the input data with all default options.
- `im1d_tend -T 200 sig.dat tend.dat sig_out`
Ditto, but increase the window size.

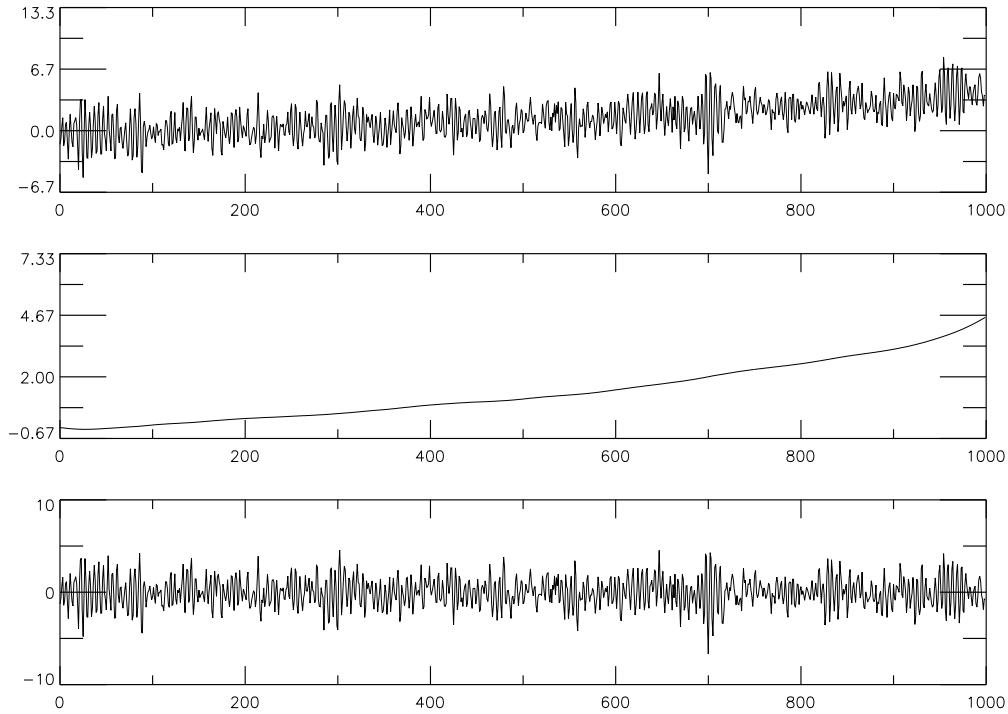


Figure 9.1: Top, input signal containing an AR(4) signal and a tendency. Middle, estimated tendency. Bottom, difference between the two previous signals.

Figure 9.1 shows the results when applying the `tend_est` program (with a window size equal to 400) to a time series containing a tendency and an AR(4).

9.2 Wavelet Transform

9.2.1 Multiresolution transform: mr1d_trans

Program *mr1d_trans* applies a one-dimensional multiresolution transform to a signal. The output file contains an image, in which each row is a band of the multiresolution transform. Morlet, Mexican hat, and French hat wavelet transforms are non-dyadic (the resolution is not decreased by a factor two between two scales), and 12 voices (fixed value) are calculated (instead of one for the dyadic case) when the resolution is divided by two. Then the number of rows in the output image will be equal to $12 * \text{number_of_scales}$. The Morlet wavelet is complex, so the wavelet transform is complex too. Using this transform, the first $12 * \text{number_of_scales}$ lines represent the real part of the transform, and the $12 * \text{number_of_scales}$ last lines the imaginary part. Four transforms are non-redundant: the bi-orthogonal, the lifting scheme, and the wavelet packet methods (transforms 16 and 17). In this case, the output is not an image but a signal (i.e. 1D rather than 2D), which has the same size as the original one. Position and length of a given band in the output signal can be found by reading the file created using the “-w” option. For the lifting scheme based method, the type of lifting can be changed using the “-l” option, and for the (bi-) orthogonal and packet one, the filter can be changed by the “-f” option.

19 transforms are available, which are grouped into 5 classes

- Class 1: no decimation (transforms 1 to 7 and 11 to 14).
- Class 2: pyramidal transform (transforms 8 to 10).
- Class 3: orthogonal transform (15 and 16).
- Class 4: Wavelet packets (17 and 18).
- Class 5: Wavelet packets via the à trous algorithm (19).

Depending on the class, the transform does not contain the same number of pixels, and the data representation differs. By default, the number of scales is calculated from the length of the signal.

USAGE: mr1d_trans option signal_in image_out

where options are:

- [-t **type_of_multiresolution_transform**]
 1. Linear wavelet transform: à trous algorithm
 2. B₁-spline wavelet transform: à trous algorithm
 3. B₃-spline wavelet transform: à trous algorithm
 4. Derivative of a B₃-spline: à trous algorithm
 5. Undecimated Haar wavelet transform: à trous algorithm
 6. Morphological median transform
 7. Undecimated (bi-) orthogonal wavelet transform
 8. Pyramidal linear wavelet transform
 9. Pyramidal B₃-spline wavelet transform
 10. Pyramidal median transform

11. Morlet's wavelet transform

Continuous wavelet transform with a complex wavelet which can be decomposed into two parts, one for the real part, and the other for the imaginary part:

$$\begin{aligned}\psi_r(x) &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \cos(2\pi\nu_0 \frac{x}{\sigma}) \\ \psi_i(x) &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \sin(2\pi\nu_0 \frac{x}{\sigma})\end{aligned}$$

First scale σ_0 is chosen equal to $2\nu_0$ and $\nu_0 = 0.8$, using 12 voices per octave.

12. Mexican hat wavelet transform

Continuous wavelet transform with the Mexican hat wavelet. This is the second derivative of a Gaussian

$$\psi(x) = (1 - \frac{x^2}{\sigma^2}) e^{-\frac{x^2}{2\sigma^2}} \quad (9.3)$$

First scale σ_0 is chosen equal to $\frac{1}{\sqrt{3}}$, using 12 voices per octave.

13. French hat wavelet transform

Continuous wavelet transform with the French hat wavelet

$$\psi(x) = \begin{cases} 0 & \text{if } |\frac{x}{\sigma}| > 1 \\ -1 & \text{if } |\frac{x}{\sigma}| \in [\frac{1}{3}, 1] \\ 2 & \text{if } |\frac{x}{\sigma}| < \frac{1}{3} \end{cases} \quad (9.4)$$

First scale σ_0 equal to 0.66, and 12 voices per octave.

14. Gaussian derivative wavelet transform

Continuous wavelet transform. The wavelet is the first derivative of a Gaussian

$$\psi(x) = -xe^{-\frac{1}{2}x^2} \quad (9.5)$$

First scale σ_0 equal to $\frac{1}{\sqrt{3}}$, and 12 voices per octave.

15. (bi-) orthogonal transform.

16. (bi-) orthogonal transform via lifting scheme.

17. Wavelet packets.

18. Wavelet packets via lifting scheme.

19. Wavelet packets using the à trous algorithm.

• [-n number_of_scales]

Number of scales used in the multiresolution transform.

• [-r]

Rebin all scales to the input signal size (for pyramidal transforms only).

• [-k]

Set to 0 pixels contaminated by the border problem.

• [-T type_of_filters]

1. Antonini 7/9 filters.
2. Daubechies filter 4.
3. Biorthogonal 2/6 Haar filters.
4. Biorthogonal 2/10 Haar filters.

5. Odegard 7/9 filters.
6. User's filters.

Default is Antonini 7/9 filters.

This option is only available if the chosen transform method is the (bi-) orthogonal transform (-t option in [7,15,17]).

- **[-L]**
Use an L_2 normalization. Default is L_1 .
- **[-l type_of_lifting_transform]**

1. Lifting scheme: CDF WT.
2. Lifting scheme: median prediction.
3. Lifting scheme: integer Haar WT.
4. Lifting scheme: integer CDF WT.
5. Lifting scheme: integer (4,2) interpolating transform.
6. Lifting scheme: Antonini 7/9 filters.
7. Lifting scheme: integer Antonini 7/9 filters.

Default is Lifting scheme: integer Haar WT.

This option is only available if the chosen transform method is the lifting scheme (-t 24).

- **[-w InfoFileName]**
Write in a file the size and the starting index of each band. This file contains a 2D float array (Array[2, NbrBand+3]).

```

info[0,0] = transform number
info[1,0] = number of scales
info[0,1] = transform class number (5 classes)
info[1,1] = number of bands
    it is not equal to the number of scales
    for wavelet packets transform.
info[0,2] = number of pixels
info[1,2] = lifting scheme type
info[0,3] = type of filter
info[1,3] = type of normalization
for i=4 to Number_of_bands + 3
info[0,i] = number of pixels in the band i
info[1,i] = position number of the pixel of the band

```

If a user filter file is given (i.e. -T 6,filename), with a filename of L characters, L lines are added to the array:

```

info[1,Number_of_bands + 4] = number of characters of the filter file name
for i=Number_of_bands+4 to Number_of_bands+4+L-1
info[0,i] = ascii number of the ith character.

```

Examples:

- mr1d_trans -n 7 -t 3 sig.fits image.fits
À trous algorithm with 7 scales.
- mr1d_trans -n 7 -T 5 -t 15 sig.fits image.fits
Bi-orthogonal wavelet transform (Odegard 7/9 filters).

- `mr1d_trans -n 7 -T 5 -t 15 -w info.fits sig.fits image.fits`
Ditto, but an information file is written. This information is necessary for reconstruction.
- `mr1d_trans -n 7 -T 5 -t 15 -w info.fits sig.fits image.fits`
Ditto, but an information file is written. This information is necessary for reconstruction.
- `mr1d_trans -n 7 -T 6,dau4 -t 15 -w info.fits sig.fits image.fits`
Ditto, but the filters defined in the file “`dau4.wvf`” are used instead of the Odegard filters.

9.2.2 Reconstruction: `mr1d_recons`

Program `mr1d_recons` reconstructs a signal from its transform. To be able to reconstruct it, the program needs the information file which has been created by the “`-w`” option during the transformation.

USAGE: `mr1d_recons TransformFileName_in InfoFileName_in signal_out`

Examples:

- `mr1d_trans -n 7 -t 3 -w info sig.fits image.fits`
`mr1d_recons image.fits info sig_rec.fits`
A trous algorithm with 7 scales, and reconstruction.
- `mr1d_trans -n 7 -T 4 -t 15 -w info sig.fits image.fits`
`mr1d_recons image.fits info sig_rec.fits`
Bi-orthogonal wavelet transform (Odegard 7/9 filters), and reconstruction.

Reconstruction is not implemented for transforms 10, 11, and 13. For pyramidal transforms, the reconstruction cannot be performed by `mr1d_recons` if the rebin option “`-r`” was set.

9.3 Filtering: `mr1d_filter`

Program `mr1d_filter` filters a signal using different methods.

USAGE: `mr1d_filter option signal_in signal_out`

where options are:

- `[-f type_of_filtering]`
 1. Multiresolution Hard K-Sigma Thresholding
 2. Multiresolution Soft K-Sigma Thresholding
 3. Iterative Multiresolution Thresholding
 4. Universal Hard Thesholding
 5. Universal Soft Thesholding
 6. SURE Hard Thesholding
 7. SURE Soft Thesholding
 8. MULTI-SURE Hard Thesholding
 9. MULTI-SURE Soft Thesholding
 10. Median Absolute Deviation (MAD) Hard Thesholding
 11. Median Absolute Deviation (MAD) Soft Thesholding
 12. Total Variation + Wavelet Constraint

Default is Multiresolution Hard K-Sigma Thresholding.

- [-t type_of_multiresolution_transform]

1. Linear wavelet transform: à trous algorithm
2. B_1 -spline wavelet transform: à trous algorithm
3. B_3 -spline wavelet transform: à trous algorithm
4. Derivative of a B_3 -spline: à trous algorithm
5. Undecimated Haar wavelet transform: à trous algorithm
6. morphological median transform
7. Undecimated (bi-) orthogonal wavelet transform
8. pyramidal linear wavelet transform
9. pyramidal B_3 -spline wavelet transform
10. pyramidal median transform

Default is 3.

- [-T type_of_filters]

1. Antonini 7/9 filters.
2. Daubechies filter 4.
3. Biorthogonal 2/6 Haar filters.
4. Biorthogonal 2/10 Haar filters.
5. Odegard 7/9 filters.
6. User's filters.

Default is Antonini 7/9 filters.

This option is only available if the chosen transform method is the (bi-) orthogonal transform (-t 7).

- [-m type_of_noise]

1. Gaussian Noise
2. Poisson Noise
3. Poisson Noise + Gaussian Noise
4. Multiplicative Noise
5. Non-stationary additive noise
6. Non-stationary multiplicative noise
7. Undefined stationary noise
8. Undefined noise
9. Stationary correlated noise
10. Poisson noise with few events

Default is Gaussian noise.

- [-g sigma]

- [-c gain,sigma,mean]

- [-E Epsilon]

Epsilon = precision for computing thresholds (only used in case of poisson noise with few events). Default is $1e - 03$.

- **[-n number_of_scales]**
- **[-s NSigma]**
- **[-i number_of_iterations]**
- **[-e epsilon]**
Convergence parameter. Default is $1e - 4$.
- **[-K]**
Suppress the last scale. Default is no.
- **[-p]**
Detect only positive structure. Default is no.
- **[-k]**
Suppress isolated pixels in the support. Default is no.
- **[-S SizeBlock]**
Size of the blocks used for local variance estimation. Default is 7.
- **[-N NiterSigmaClip]**
Iteration number used for local variance estimation. Default is 1.
- **[-F first_detection_scale]**
If this option is set, all wavelet coefficients detected at scales lower than *first_detection_scale* are considered as significant.
- **[-G RegulParam]**
Regularization parameter for filtering methods 11 and 12. default is 0.01.
- **[-v]**
Verbose. Default is no.

Examples:

- `mr1d_filter sig.fits filter.sig.fits`
Filtering using the à trous algorithm, and a Gaussian noise model.
- `mr1d_filter -m 2 sig.fits filter.sig.fits`
Ditto, but assuming Poisson noise.

9.4 Band Detection: mr1d_detect

Program *mr1d_detect* detects emission or absorption bands in a spectrum [198]. The output is an image, in which each row contains a detected band. By default, *mr1d_detect* detects both emission and absorption bands. The program gives better results if we detect only emission (“-e” option) bands, or only absorption bands (“-a” option). This is due to the fact that positivity and negativity constraints can be introduced in the iterative reconstruction for, respectively, emission and absorption band detection. If the user wants to detect both kinds of band at the same time, he/she should use the multiresolution median transform (“-M” option) which gives in this case better results than the wavelet transform.

USAGE: mr1d_detect option signal_in tab_band_out

where options are:

- **[-n number_of_scales]**
- **[-s NSigma]**
- **[-m type_of_noise]**

1. Gaussian Noise
2. Poisson Noise
3. Poisson Noise + Gaussian Noise
4. Multiplicative Noise
5. Non-stationary additive noise
6. Non-stationary multiplicative noise
7. Undefined stationary noise
8. Undefined noise

Description in section 4.2. Default is Gaussian noise.

- **[-g sigma]**
- **[-c gain,sigma,mean]**
- **[-a]**
Detection of Absorption lines only. Default is not to do this.
- **[-e]**
Detection of Emission lines only. Default is not to do this.
- **[-f FirstScale]**
First scale. Default is 1. All bands detected at the scale smaller than *FirstScale* are not reconstructed.
- **[-l LastScale]**
Last scale. Default is number_of_scales – 2. All bands detected at the scale higher than *LastScale* are not reconstructed.
- **[-i IterNumber]**
Number of iterations for the reconstruction. Default is 20.
- **[-M]**
Use the multiresolution median transform instead of the à trous algorithm. If this option is set, the reconstruction is not iterative, and the previous option (“-i”) is not active.
- **[-A]**
Detect only negative multiresolution coefficients. Default is no.
- **[-E]**
Detect only positive multiresolution coefficients. Default is no.
- **[-w]**
Write two other files:
 - tabadd.fits: sum of the reconstructed objects. All detected bands are co-added.
 - tabseg.fits: segmented wavelet transform.
- **[-v]**
Verbose. Default is no.

Examples:

- `mr1d_detect -a -A sig.fits band_sig.fits`
Detection using the à trous algorithm, and a Gaussian noise model. The detection is performed from the positive wavelet coefficients, and only emission bands are reconstructed.
- `mr1d_detect -m 2 -a -A -s 5 sig.fits band_sig.fits`
Ditto, but assuming Poisson noise and a five sigma detection.

A simulated spectrum was created containing a smooth continuum with superimposed, variable Gaussian noise and an emission band at $3.50\mu\text{m}$ with a maximum of five times the local noise standard deviation and a width of $\text{FWHM} = 0.01\mu\text{m}$ (see Figure 9.2 top). Figure 9.2 (middle) shows the comparison of the reconstructed emission band with the original Gaussian. Figure 9.2 (bottom) shows the difference between the simulated spectrum and the reconstructed band.

9.5 Modulus Maxima Representation

9.5.1 Modulus maxima detection: `mr1d_max`

Program `mr1d_max` detects the modulus maxima using the dyadic à trous algorithm with the wavelet function equal to the derivative of a B-spline (transform number 4 in `mr1d_trans`).

USAGE: `mr1d_max option signal_in output`

where options are:

- `[-n number_of_scales]`
- `[-v]`
Verbose. Default is no.

Example:

- `mr1d_max sig.fits band.sig.fits`
Modulus maxima detection.

9.5.2 Reconstruction: `mr1d_maxrecons`

Program `mr1d_maxrecons` reconstructs a signal from its modulus maxima.

USAGE: `mr1d_maxrecons option tab_band_in signal_out`

where options are:

- `[-i number_of_iterations]`
Maximum number of iterations for the reconstruction. Default is 10.
- `[-v]`
Verbose. Default is no.

Example:

- `mr1d_max sig.fits band.sig.fits`
`mr1d_maxrecons band.sig.fits sig_rec.fits`
Modulus maxima detection, and reconstruction from the maxima.

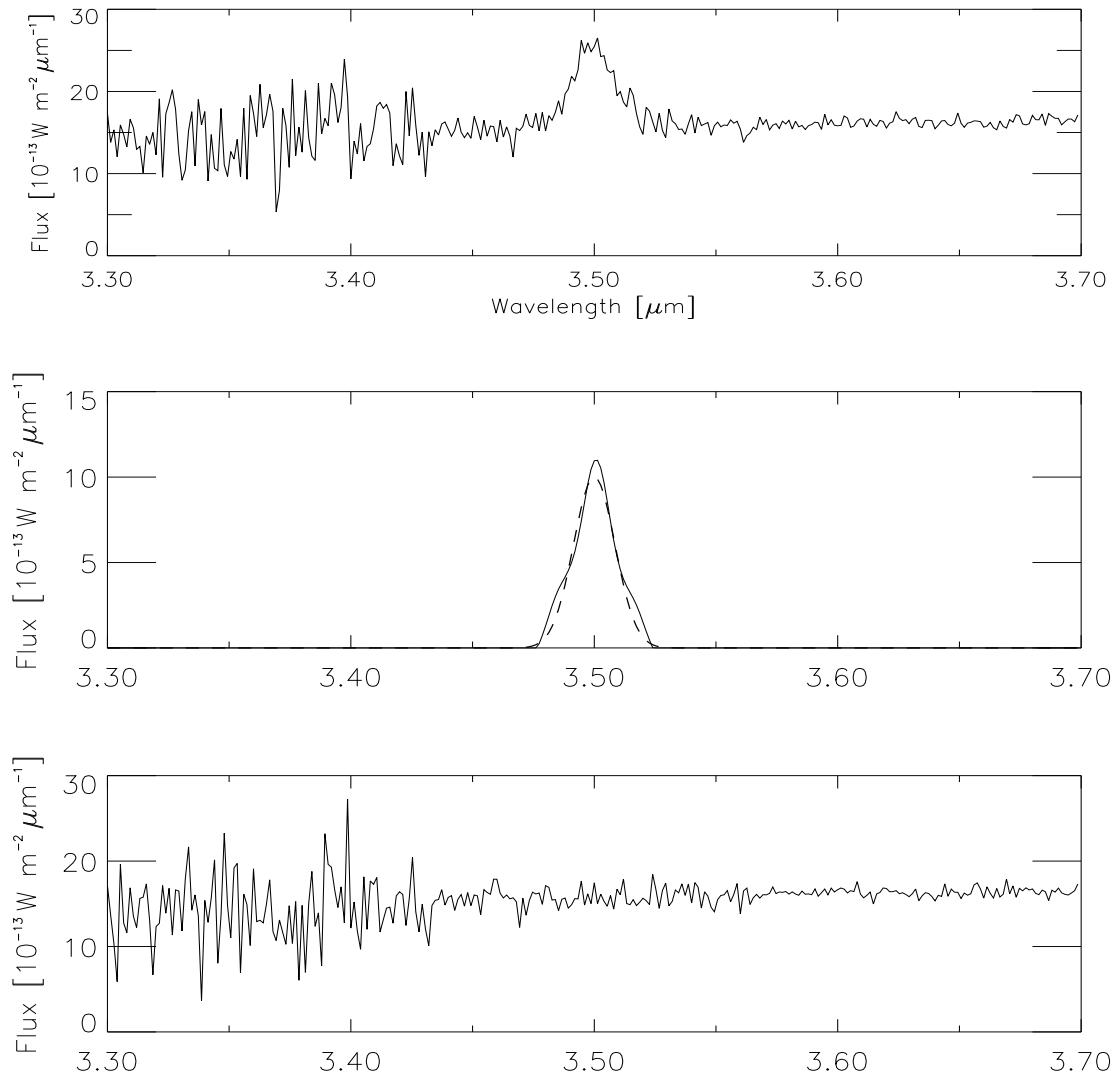


Figure 9.2: Simulation: top, simulated spectrum, middle, reconstructed simulated band (full line) and original band (dashed line). Bottom, simulated spectrum minus the reconstructed band.

9.6 Wavelet Analysis of Time Series

9.6.1 Introduction

Several approaches have been proposed for time series prediction by the WT, based on a neural network [217, 12], Kalman filtering [46, 93], or an AR (autoregression) model [176]. In [217, 176] the undecimated Haar transform was used. This choice of the Haar transform was motivated by the fact that the wavelet coefficients are calculated only from data obtained previously in time, and the choice of an undecimated wavelet transform avoids aliasing problems.

The à trous wavelet transform with a wavelet function related to a spline function, as described earlier, is not consistent with a directed (time-varying) data stream. We now keep the wavelet function, but alter the wavelet transform, to make of it a multiscale transform which is appropriate for a data stream. We consider a signal $s(1), s(2), \dots, s(n)$, where n is the present time-point.

1. For index k sufficiently large, carry out an à trous wavelet transform on $\{s(1), s(2), \dots, s(k)\}$.
2. Retain the detail coefficient values, and the continuum value, for the k th time-point only (cf. equation 8): $w_{1,k}, w_{2,k}, w_{J,k}, c_{J,k}$. Note that summing these values gives $s(k)$.
3. If k is less than n , set k to $k+1$ and return to Step 1.

This produces an additive decomposition of the signal, which is similar to the à trous wavelet transform decomposition with the B_3 spline on $\{s(1), s(2), \dots, s(k), \dots, s(n)\}$. The computational time is evidently greater, $O(n^2)$ as against $O(n)$.

We have not touched on an important matter in regard to equation 2: how to handle signal boundaries. Although other strategies could be envisaged, we use a mirror approach. This is tantamount, of course, to redefining the discrete filter associated with the scaling function in the signal boundary region; and to redefining the associated wavelet function in this region. This strategy is of particular relevance when we work on an ordered data stream. We hypothesize future data based on values in the immediate past. Not surprisingly there is discrepancy in fit in the succession of scales, which grows with scale as larger numbers of immediately past values are taken into account.

9.6.2 The redundant Haar wavelet transform

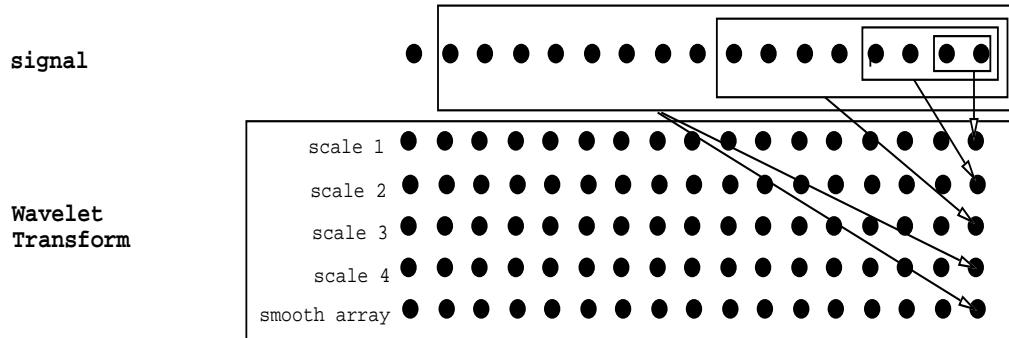


Figure 9.3: This figure shows which pixels of the input signal are used to calculate the last wavelet coefficient in the different scales.

The Haar wavelet transform was first described in the early years of this century and is described in almost every text on the wavelet transform. As already mentioned, the asymmetry of the wavelet function used makes it a good choice for edge detection, i.e. localized jumps. The usual Haar wavelet transform, however, is a decimated one. We now develop a non-decimated or redundant version of this transform. This will be an à trous algorithm, but with a different pair of scaling and wavelet functions compared to those used previously.

The non-decimated Haar algorithm is exactly the same as the à trous algorithm, except that the low-pass filter h , $(\frac{1}{16}, \frac{1}{4}, \frac{3}{8}, \frac{1}{4}, \frac{1}{16})$, is replaced by the simpler filter $(\frac{1}{2}, \frac{1}{2})$. There h is now non-symmetric. Consider the creation of the first wavelet resolution level. We have created it from by convolving the original signal with h . Then:

$$c_{j+1,l} = 0.5(c_{j,l-2^j} + c_{j,l}) \quad (9.6)$$

and

$$w_{j+1,l} = c_{j,l} - c_{j+1,l} \quad (9.7)$$

At any time point, l , we never use information after l in calculating the wavelet coefficient. Figure 9.3 shows which pixels of the input signal are used to calculate the last wavelet coefficient in the different scales. A wavelet coefficient at a position t is calculated from the signal samples at positions less than or equal to t , but never larger.

9.6.3 Autoregressive Multiscale Prediction

Stationary signal

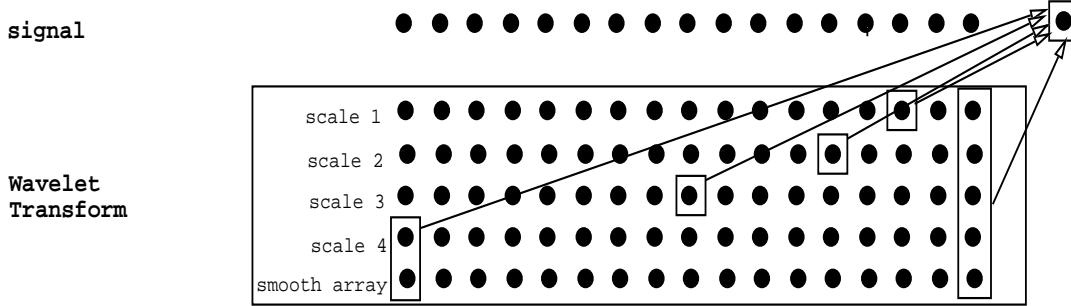


Figure 9.4: Wavelet coefficients which are used for the prediction of the next value.

Assuming a stationary signal $D = (d_1, \dots, d_N)$, the AR (autoregressive) multiscale prediction model is:

$$d_{t+1} = \sum_{j=1}^J \sum_{k=1}^{A_j} a_{j,k} w_{j,t-2^j(k-1)} + \sum_{k=1}^{A_{J+1}} a_{J+1,k} c_{J,t-2^J(k-1)} + \epsilon(t+1) \quad (9.8)$$

where $\mathcal{W} = w_1, \dots, w_J, c_J$ represents the Haar à trous wavelet transform of D ($D = \sum_{j=1}^J w_j + c_J$). For example, choosing $A_j = 1$ for all resolution levels j leads to the equation

$$d_{t+1} = \sum_{j=1}^J a_j w_{j,t} + a_{J+1} c_{J,t} + \epsilon(t+1) \quad (9.9)$$

Figure 9.4 shows which wavelet coefficients are used for the prediction using $A_j = 2$ for all resolution levels j , and a wavelet transform with five scales (four wavelet scales + the smoothed array). In this case, we can see that only ten coefficients are used, but taking into account low resolution information. This means that a long term prediction can easily be introduced, either by increasing the number of scales in the wavelet transform, or by increasing the AR order in the last scales, but with a very small additional number of parameters. To find the $Q = \sum_{j=1}^{J+1} A_j$

unknown parameters:

$$\{\{a_{1,1}, \dots, a_{1,A_1}\}, \dots, \{a_{j,1}, \dots, a_{j,A_j}\}, \dots, \{a_{J,1}, \dots, a_{J,A_J}\}, \{a_{J+1,1}, \dots, a_{J,A_{J+1}}\}\}$$

of our model, we need to resolve the following equation: $AX = S$, where A, X, W are defined by:

$$\begin{aligned} A^t &= (L_{N-1}, \dots, L_{N-P}) \\ L_i &= (w_{1,i}, \dots, w_{1,i-2A_1}, \dots, w_{2,i}, \dots, w_{2,i-2^2A_2}, \dots, w_{J,i}, \dots, w_{J,i-2^JA_J}, c_{J,i}, \dots, c_{J,i-2^JA_{J+1}}) \\ X^t &= (a_{1,1}, \dots, a_{1,A_1}, a_{2,1}, \dots, a_{2,A_2}, \dots, a_{J,1}, \dots, a_{J,A_j}, \dots, a_{J+1,1}, \dots, a_{J+1,A_{J+1}}) \\ S^t &= (d_N, \dots, d_{i+1}, \dots, d_{N-P+1}) \end{aligned}$$

We have P equations and Q unknowns, so A is a $Q \times P$ matrix (P rows L_i , each with Q elements), X and S are respectively Q - and P -sized vectors. When Q is larger than P , many minimization methods may be used to find X . In our experiments, we used the standard SVD method.

Non-stationary signal

When the signal is not stationary, the previous method will not correctly model our data. However, in many cases, the non-stationary part affects the low frequency components, while the high frequencies may still give rise to stationary behavior. Then we can separate our signal D into two parts, the low and the high frequencies L and H :

$$\begin{aligned} L &= c_J \\ H &= D - L = \sum_{j=1}^J w_j \\ d_{t+1} &= l_{t+1} + h_{t+1} \end{aligned}$$

The smoothed array of the wavelet transform is first subtracted from the data, and we consider now that the signal H is stationary. Our prediction will be the coaddition of two predicted values, one on the signal H by the AR Multiscale model, and the second on the low frequency component by some other method. The AR-Multiscale model gives:

$$h_{t+1} = \sum_{j=1}^J \sum_{k=1}^{A_j} a_{j,k} w_{j,t-2^j(k-1)} + \epsilon(t+1) \quad (9.10)$$

We have now $Q = \sum_{j=1}^J A_j$ unknown parameters:

$$\{\{a_{1,1}, \dots, a_{1,A_1}\}, \dots, \{a_{j,1}, \dots, a_{j,A_j}\}, \dots, \{a_{J,1}, \dots, a_{J,A_J}\}\}$$

and we need to resolve the following equation: $AX = S$, where A, X, W are defined by:

$$\begin{aligned} A^t &= (L_{N-1}, \dots, L_{N-P}) \\ L_i &= (w_{1,i}, \dots, w_{1,i-2A_1}, \dots, w_{2,i}, \dots, w_{2,i-2^2A_2}, \dots, w_{J,i}, \dots, w_{J,i-2^JA_J}) \\ X^t &= (a_{1,1}, \dots, a_{1,A_1}, a_{2,1}, \dots, a_{2,A_2}, \dots, a_{J,1}, \dots, a_{J,A_j}) \\ S^t &= (h_N, \dots, h_{i+1}, \dots, h_{N-P+1}) \end{aligned}$$

Many methods may be used for the prediction of l_{t+1} . The problem is simplified by the fact that L is very smooth. We used a polynomial fitting of degree 3 in our experiments.

AR order determination

The AR order at the different scales must now be defined. It can be a user parameter, but an automatic method is generally preferable. At each scale j , we need to know how many coefficients should be used. This value A_j may be determined by looking at how the wavelet coefficients at the scale j are correlated. Therefore each scale is first analyzed separately, and the best AR order A_j at scale j minimizes:

$$J(A_j) = \log \sigma_{A_j}^2 + \mathcal{P}(A_j)$$

where σ_{A_j} is the prediction error, and \mathcal{P} is a penalty function, which increases with the AR order. Examples of penalty functions are:

- AIC: $AIC = \log \sigma_{A_j}^2 + \frac{2A_j}{N}$
- AICC: $AICC = \log \sigma_{A_j}^2 + \frac{N+A_j}{N-A_j-2}$
- SIC: $SIC = \log \sigma_{A_j}^2 + \frac{A_j \log N}{N}$

9.6.4 Wavelets and autocorrelation function: mr1d_acor

Program *mr1d_acor* calculates the autocorrelation at each scale of the wavelet transform.

USAGE: mr1d_acor option signal_in autocor_out

where options are:

- [-n number_of_scales]
Number of scales used in the multiresolution transform.
- [-S Nbr_of_Lags]
Default is 10.

9.6.5 Transition detection: mr1d_nowcast

Program *mr1d_nowcast* detects the transitions in all scales at a given position. The wavelet transform used is the Haar transform, so a given wavelet coefficient at position x and at scale j ($j = 1..P$, P being the number of scales) is calculated from pixel values between positions $x - 2^j + 1$ and x . Only pixels in the signal which are on the left of a given position x (or before a given time for temporal signal) are used for the calculation of the wavelet coefficients at position x . This allows us to detect a new event in a temporal series irrespective of the time scale of the event. By default, the analysed position is the last one of the signal, but other positions can equally well be analyzed using the “-x” option. The program prints for each scale j the following information corresponding to the position x :

- **No detection**
if the wavelet coefficient $|w_j(x)| < k\sigma_j$
- **New upward detection**
if $w_j(x) > k\sigma_j$ and $|w_j(x-1)| < k\sigma_j$
- **New downward detection**
if $w_j(x) < -k\sigma_j$ and $|w_j(x-1)| < k\sigma_j$
- **Positive significant structure**
if $w_j(x) > k\sigma_j$ and $|w_j(x-1)| > k\sigma_j$
The first detected coefficient of the structure is also given.

- **Negative significant structure**

if $w_j(x) < -k\sigma_j$ and $|w_j(x-1)| > k\sigma_j$

The first detected coefficient of the structure is also given.

- **End of significant structure**

if $|w_j(x)| < k\sigma_j$ and $|w_j(x-1)| > k\sigma_j$

Furthermore the signal to noise ratio of the wavelet coefficient is given.

USAGE: `mr1d_nowcast option signal.in`

where options are:

- **[`-m type_of_noise`]**

1. Gaussian Noise
2. Poisson Noise
3. Poisson Noise + Gaussian Noise
4. Multiplicative Noise
5. Non-stationary additive noise
6. Non-stationary multiplicative noise
7. Undefined stationary noise
8. Undefined noise

Default is Gaussian noise.

- **[`-g sigma`]**

- **[`-c gain,sigma,mean`]**

- **[`-n number_of_scales`]**

- **[`-s NSigma`]**

- **[`-x Position`]**

Position to analyse. Default is the last point.

Examples:

- `mr1d_nowcast sig.dat`

Analyse the last point of the signal with all default option.

- `mr1d_nowcast -x 55 -s 10 sig.dat`

Analyse the point at position 55, and detect the transition with a signal to noise ratio equal to 10.

9.6.6 Prediction: `mr1d_fcast`

Program `mr1d_fcast` performs a forecasting by four different methods: the standard AR model (AR), an AR method per scale (and the prediction is the coaddition of the predicted wavelet coefficients), the multiresolution AR model (MAR), and a neural network. The program can be used in two modes: the evaluation and the prediction mode. In the evaluation mode, the first part of the time series is used to predict the second part, and the output file contains the same number of values as the input file. The initial values (corresponding to the initial part of the signal) are identical, and the other values are the predicted ones. The error prediction is calculated and printed on the standard output device (screen window). In the prediction mode, the output file contains more values than the input file. The last values correspond to the predicted values. For the AR and MAR models, the order of the model can either be fixed by

the user (“-a option”), or automatically calculated. In case of an automatic MAR model order estimation, the order can be different at each scale.

By default, the signal is assumed to be stationary. If the “-h” option is set, we assume a non-stationary signal. In that case, the last scale of the wavelet transform is analyzed differently (i.e. not with the AR model). Several methods can be selected by the “-B” option. The default is the polynomial extrapolation of order 2.

If the “-L” option is set, only the last pixels will be used in the analysis.

USAGE: mr1d_fcast option signal_in signal_out

where options are:

- **[-P predict_method]**

1. Autoregressive model.
2. Autoregressive model per scale.
3. Multiresolution Autoregressive model.
4. Neural network.

Default is Multiresolution Autoregressive Model.

- **[-n number_of_scales]**

Number of scales to be used in the Multiresolution AR model. Default is 5.

- **[-a AR_Order]**

AR order used for the prediction. Default is automatically estimated.

- **[-O Estimation_AR_Order_Method]**

1. AIC
2. AICC
3. BIC

Default is BIC method.

- **[-h]**

Non stationary signal. Default is stationary.

- **[-p Number_of_Predict]**

Number of prediction. Default is 0.

- **[-m MinAROrder]**

Minimum AR model order. Default is 1.

- **[-M MaxAROrder]**

Maximum AR model order. Default is 10.

- **[-w]**

Write to disk some information about the prediction error and the AR model order. The file contains a 1D table of size $N + 3$, where N is the number of scales ($N = 1$ when the MAR model is not used).

```
T[0] = prediction error
T[1] = Number_of_scale
T[2] = Percentage of prediction in the interval [Pred-SigmaNoise, Pred+SigmaNoise].
for j = 0 to Number_of_scale-1 do T[j] = AR order at scale j
```

- **[-L NPix]]**

Analyse the last NPix pixels. Default is all pixels.

- [-B extrapol_type]

1. Constant border ($c_J(N+i) = c_J(N)$, where c_J is the last scale, N the number of pixels).
2. Mirror border ($c_J(N+i) = c_J(N-i)$).
3. Double mirror border ($c_J(N+i) = 2c_J(N) - c_J(N-i)$).
4. Polynomial extrapolation (deg 1).
5. Polynomial extrapolation (deg 2).

Default is 5. Only used if the “-h” option is set.

- [-T Poly_Nbr_Pix]

Number of pixels used for the polynomial extrapolation. Default is 5. Only used if the “-h” option is set and if a polynomial extrapolation is used.

Examples:

- mr1d_fcast sig.dat eval.dat
Evaluate the prediction by MAR method.
- mr1d_fcast -p 1 sig.dat pred.dat
Make a prediction at position N+1.

9.7 Time-Frequencies Analysis

9.7.1 The Short Term Fourier Transform: im1d_stf

The Short-Term Fourier Transform of a 1D signal f is defined by:

$$STFT(t, \nu) = \int_{-\infty}^{+\infty} e^{-j2\pi\nu\tau} f(\tau)g(\tau-t)d\tau \quad (9.11)$$

If g is the Gaussian window, this corresponds to the Gabor transform. The energy density function, called the *spectrogram*, is given by:

$$SPEC(t, \nu) = | STFT(t, \nu) |^2 = \left| \int_{-\infty}^{+\infty} e^{-j2\pi\nu\tau} f(\tau)g(\tau-t)d\tau \right|^2 \quad (9.12)$$

Most used windows are:

- Truncated window function: $\hat{W}(\nu) = \begin{cases} 1 & \text{if } |\hat{P}(\nu)| \geq \sqrt{\epsilon} \\ 0 & \text{otherwise} \end{cases}$ where ϵ is the regularization parameter.
- Rectangular window: $\hat{W}(\nu) = \begin{cases} 1 & \text{if } |\nu| \leq \Omega \\ 0 & \text{otherwise} \end{cases}$ where Ω defines the band-width.
- Triangular window: $\hat{W}(\nu) = \begin{cases} 1 - \frac{\nu}{\Omega} & \text{if } |\nu| \leq \Omega \\ 0 & \text{otherwise} \end{cases}$
- Hamming Window: $\hat{W}(\nu) = \begin{cases} 0.54 + 0.46 \cos\left(\frac{2\pi\nu}{\Omega}\right) & \text{if } |\nu| \leq \Omega \\ 0 & \text{otherwise} \end{cases}$

- Hanning Window: $\hat{W}(\nu) = \begin{cases} \cos\left(\frac{\pi\nu}{\Omega}\right) & \text{if } |\nu| \leq \Omega \\ 0 & \text{otherwise} \end{cases}$
- Gaussian Window: $\hat{W}(\nu) = \begin{cases} \exp(-4.5\frac{\nu^2}{\Omega^2}) & \text{if } |\nu| \leq \Omega \\ 0 & \text{otherwise} \end{cases}$
- Blackman Window: $\hat{W}(\nu) = \begin{cases} 0.42 + 0.5 \cos\left(\frac{\pi\nu}{\Omega}\right) + 0.08 \cos\left(\frac{2\pi\nu}{\Omega}\right) & \text{if } |\nu| \leq \Omega \\ 0 & \text{otherwise} \end{cases}$

The inverse transform is obtained by:

$$f(t) = \int_{-\infty}^{+\infty} g(t - \tau) \int_{-\infty}^{+\infty} e^{j2\pi\nu\tau} STFT(\tau, \nu) d\nu d\tau \quad (9.13)$$

Program *im1d_stf* calculate the short term Fourier Transform of a signal. It outputs three files: the real part of the STF, the imaginary part of the STF, and the spectrogram. When the “-r” option is set, the signal is reconstructed from the STF real and imaginary parts (the spectrogram is not used for the reconstruction).

The command line is:

USAGE: im1d_stf option signal_in image_out

where options are:

- **[-t type_of_window]**
 1. Hamming window
 2. Hanning window
 3. Gaussian window
 4. Blackman window
 5. Rectangular window

Default is Hamming window.

- **[-w window_size]**
Window size. Default is 1024. If the signal has less than 1024 pixels, the default value is the number of pixels divided by four.
- **[-W window_param]**
Window parameter. Default is 0.5.
- **[-S Step]**
Step between two points. Default is WindowSize/2.
- **[-r]**
Reverse transform.
- **[-v]**
Verbose. Default is no.

Examples:

- `im1d_stf sig.fits stf`
Creates three files, “`stf_re.fits`”, “`stf_im.fits`”, “`stf_spec.fits`”, corresponding respectively the STF real part, the STF imaginary part and the spectrogram.
- `im1d_stf -r stf rec`
Reconstruct a signal from its STF (i.e. “`stf_re.fits`” and “`stf_im.fits`”).

9.7.2 Time-Frequency Distribution: im1d_tfreq

The Wigner-Ville distribution [215, 209] of a signal $s(t)$ is

$$W(t, \nu) = \frac{1}{2\pi} \int s^*(t - \frac{1}{2}\tau)s(t + \frac{1}{2}\tau)e^{-i\tau 2\pi\nu} d\tau \quad (9.14)$$

where s^* is the conjugate of s . The Wigner-Ville transform is always real (even for a complex signal). In practice, its use is limited by the existence of interference terms, even if they can be attenuated using specific averaging approaches. More details can be found in [42, 121].

Program `im1d_stf` calculates the short term Fourier Transform of a signal. It outputs three files: the real part of the STF, the imaginary part of the STF, and the spectrogram.

The command line is:

USAGE: im1d_stf option signal_in image_out

where options are:

- **[-T TimeFrequency_Distrib]**
 1. Short Term Fourier Transform Spectrogram
 2. Wigner-Ville Distribution

Default is 1.

- **[-t type_of_window]**
 1. Hamming window
 2. Hanning window
 3. Gaussian window
 4. Blackman window
 5. Rectangular window

default is Hamming window.

- **[-w window_size (time domain)]**
Window size. Default is 1024. If the signal has less than 1024 pixels, the default value is the number of pixels divided by four.
- **[-W window_param (time domain)]**
Window parameter. Default is 0.5.
- **[-S Step]**
Step between two points. Default is WindowSize / 2.
- **[-v]**
Verbose. Default is no.

Chapter 10

MR/1 Wavelet and Multifractal Analysis

10.1 Fractal

10.1.1 Introduction

The word “fractal” was introduced by Mandelbrot (1977) [126], and comes from the Latin word *fractus* which means “to break”. According to Mandelbrot, a fractal is an object which has a greater dimension than its topological dimension. A typical fractal is the Cantor set.

Example: the triadic Cantor set

The Cantor set is built in the following way: considering a segment of dimension L , we separate it into three equal parts, and suppress the middle part. There remain two segments of size $\frac{L}{3}$. Repeating the process on both segments, we get four segments of size $3^{-2}L$. After n iterations, we have 2^n segments of size $3^{-n}L$. The Cantor set is obtained when n tends to infinity. The set has the following properties:

1. It is self-similar.
2. It has a fine structure, i.e. detail on arbitrary small scales.
3. It is too irregular to be described in traditional geometrical language, both locally and globally.
4. It is obtained by successive iterations.
5. It has an infinite number of points but its length is null.

These properties define in fact a fractal object.

A real fractal does not exist in the nature, and we always need to indicate at which scales we are talking about fractality.

10.1.2 The Hausdorff and Minkowski measure

Measure

An object dimension describes how an object F fills the space. A simple manner of measuring the length of curves, the area of surfaces or the volume of objects is to divide the space into small boxes (segment in one dimension, surface in 2D, and cubes in 3D) of diameter δ as shown in Figure 10.1 [71]. These boxes are chosen so that their diameter is not greater than a given size δ , which corresponds to the measure resolution.

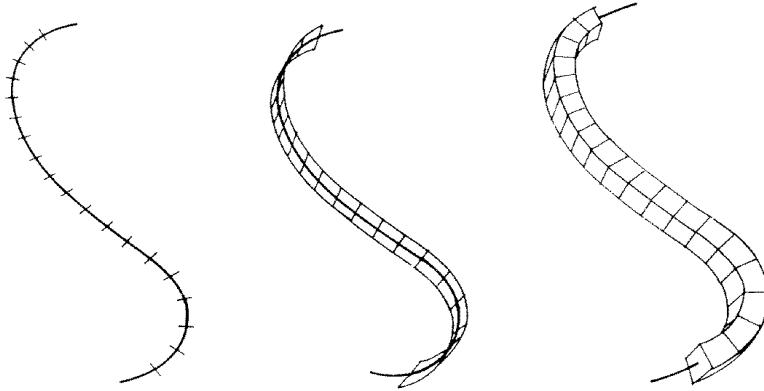


Figure 10.1: Measuring the “size” of curves (Feder 1988).

We consider now the quantity:

$$L_\delta^d(F) = \sum diam(B_i)^d \quad (10.1)$$

where d is a real number, $diam(B_i)$ is the diameter of the box i . $L_\delta^s(F)$ represents an estimation of the size of F at the resolution δ . Depending on the choice of the boxes, the measure is more or less correct. Generally, it is easier to manipulate the Minkowski-Bouligand measure which fixes all the boxes to the same size δ . Using this measure, the size of F is given by:

$$M^s(F) = \lim_{\delta \rightarrow 0} \sum diam(B_i)^s = \delta^s N_B(\delta) \quad (10.2)$$

where N_B is the number of boxes needed to cover the object F . Then the curves of length L_* in Figure 10.1 can be measured by finding the number $N_B(\delta)$ of line segments (respectively squares and cubes for the second and third object) of length δ needed to cover the object. The three sizes are:

$$\begin{aligned} L &= M^1(F) = N_B(\delta) \delta^1 \xrightarrow{\delta \rightarrow 0} L_* \delta^0 \\ A &= M^2(F) = N_B(\delta) \delta^2 \xrightarrow{\delta \rightarrow 0} L_* \delta^1 \\ V &= M^3(F) = N_B(\delta) \delta^3 \xrightarrow{\delta \rightarrow 0} L_* \delta^2 \end{aligned} \quad (10.3)$$

10.1.3 The Hausdorff and Minkowski dimension

The Hausdorff dimension d_H of the set F is the *critical dimension* for which the measure $H^d(F)$ jumps from infinity to zero:

$$H^d(F) = \begin{cases} 0, & d > d_H, \\ \infty, & d < d_H. \end{cases} \quad (10.4)$$

But $H^{d_H}(F)$ can be finite or infinite. For a simple set (segment, square, cube), Hausdorff dimension is equal to the topological dimension (i.e. 1, 2, or 3). This is not true for a more complex set, such as the Cantor set. Minkowski dimension d_M (and $d_H(F) \leq d_M(F)$) is defined in a similar way using Minkowski measure.

By definition, we have:

$$M^{d_M} = \lim_{\delta \rightarrow 0} \delta^{d_M} N_B(\delta) \quad (10.5)$$

When $\delta \rightarrow 0$, we have $d_M \ln M = d_M \ln \delta + \ln N_N(\delta)$. If M is finite, the Minkowski dimension, also called box-counting, can be defined by

$$d_M = \lim_{\delta \rightarrow 0} \frac{\ln N_B(\delta)}{-\ln \delta} \quad (10.6)$$

In the case of the Cantor set, at iteration n , we have 2^n segments of size 3^{-n} ($\delta = 3^{-n}$). When $n \rightarrow \infty$, we have

$$d_M(\text{Cantor}) = \frac{\ln 2^n}{-\ln 3^{-n}} = \frac{\ln 2}{\ln 3} \quad (10.7)$$

10.2 Multifractality

The multifractal picture is a refinement and generalization of the fractal properties that arise naturally in the case of self-similar distributions. The singularity spectrum $f(\alpha)$ can be introduced as a quantity which characterizes the degree of regularity and homogeneity of a fractal measure.

10.2.1 Definition

Hölder exponent

A multifractal measure describes a non-homogeneous set A . Such a measure is called multifractal if it is everywhere self-similar, i.e. if the measure varies locally as a power law, at any point of A . Denoting μ a measure, we call the Hölder exponent or singularity exponent at x_0 the limit

$$\alpha(x_0) = \lim_{\delta \rightarrow 0} \frac{\ln \mu(B_{x_0}(\delta))}{\ln \delta} \quad (10.8)$$

where B_{r_0} is a box centered at r_0 of size δ . We have:

$$\mu(B_{x_0}(\delta)) \propto \delta^{\alpha(x_0)} \quad (10.9)$$

The smaller the value $\alpha(x_0)$, the less the measure is regular around x_0 . For example, if μ corresponds to a Dirac distribution centered at 0, then $\alpha(0) = 0$, and if μ corresponds to a Gaussian distribution, then $\alpha(0) = -1$.

Singularity spectrum

The singularity spectrum, associated with a measure μ , is the function which associates with α the fractal dimension of any point x_0 such that $\alpha(x_0) = \alpha$:

$$f(\alpha) = d_F(\{x_0 \in A \mid \alpha(x_0) = \alpha\}) \quad (10.10)$$

The function $f(\alpha)$ is usually ([151]) a single-humped function with the maximum at $\max_\alpha f(\alpha) = D$, where D is the dimension of the support. In the case of a single fractal, the function $f(\alpha)$ is reduced to a single point: $f(\alpha) = \alpha = D$.

The singularity spectrum describes statistically the α exponent distribution on the measure support. For example, if we split the support into boxes of size δ , then the number of boxes with a measure varying as δ^α for a given α is

$$N_\alpha(\delta) \propto \delta^{-f(\alpha)} \quad (10.11)$$

$f(\alpha)$ describes the histogram of $N_\alpha(\delta)$ when δ is small. A measure is homogeneous if its singularity spectrum is concentrated in a single point. If $f(\alpha)$ is large, the measure is multifractal.

Multifractal quantities

From a practical point of view one does not determine directly the spectrum of exponents $[f(\alpha), \alpha]$; it is more convenient to compute its Legendre transformation $[\tau(q), q]$ given by

$$\begin{cases} f(\alpha) = q \cdot \alpha - \tau(q) \\ \alpha = \frac{d\tau(q)}{dq} \end{cases} \quad (10.12)$$

In the case of a simple fractal one has $\alpha = f(\alpha) = D$. In terms of the Legendre transformation this corresponds to

$$\tau(q) = D(q - 1) \quad (10.13)$$

i.e. the behavior of $\tau(q)$ versus q is a straight line with coefficient given by the fractal dimension.

10.2.2 Generalized fractal dimension

Definition

The generalized fractal dimension, also called Reyni dimension of order q , is given by:

$$D_q = \frac{\tau(q)}{q - 1} \quad (10.14)$$

D_0 is also called capacity dimension, and coincides with the Hausdorff dimension. Dimensions D_1 , and D_2 are respectively called information and correlation dimension.

Partition function

The partition function Z is defined by:

$$Z(q, \delta) = \sum_{i=1}^{N(\delta)} \mu_i^q(\delta) \quad (10.15)$$

where we denote $\mu_i(\delta) = \mu(B_i(\delta))$. If the measure μ is multifractal, Z follows a power law in the limit $\delta \rightarrow 0$.

$$Z(q, \delta) \propto \delta^{\tau(q)} \quad (10.16)$$

The box-counting method consists of calculating the partition function, to derive from Z $\tau(q)$, and to obtain, the multifractal spectrum by a Legendre transform.

10.3 Wavelets and Multifractality

10.3.1 Singularity analysis

Let $f(x)$ be the input signal, x_0 the singularity location, $\alpha(x_0)$ the Hölder exponent at the singularity point x_0 and n the degree of Taylor development such that $n \leq \alpha(x_0) < n + 1$. We have

$$\begin{aligned} f(x) &= f(x_0) + (x - x_0) f^{(1)}(x_0) + \dots + \\ &\quad \frac{(x - x_0)^n}{n!} f^{(n)}(x_0) + C |x - x_0|^{\alpha(x_0)} \end{aligned} \quad (10.17)$$

Letting ψ be the wavelet with $n_\psi > n$ vanishing moments, then we have for the wavelet transform of $f(x)$ at x_0 when the scale goes to 0 (ψ is orthogonal to polynomials up to order n):

$$\lim_{\text{scales} \rightarrow 0^+} T_\psi[f](x_0, s) \sim a^{\alpha(x_0)} \quad (10.18)$$

One can prove that if f is C^∞ , then we have

$$\lim_{\text{scales} \rightarrow 0^+} T_\psi[f](x_0, s) \sim a^{n_\psi} \quad (10.19)$$

Thus, we have

$$\begin{cases} T_\psi[f] \sim s^{n_\psi} & \text{where the signal } f \text{ is regular} \\ T_\psi[f] \sim s^\alpha (>> s^{n_\psi}) & \text{around a singular zone} \end{cases} \quad (10.20)$$

For a fixed scale s , $T_\psi[f](., s)$ will be greater when the signal is singular. This local maximum is organized in maxima lines (function of s) which converges, when s goes to 0, to a singularity of the signal. Mallat and Hwang (1992) demonstrate that to recover the Hölder exponent $\alpha(x_0)$ at x_0 , one need only study the wavelet transform along these lines of maxima which converge (when the scale goes to 0) towards the singularity point x_0 .

Along this maxima line 1 we have

$$T_\psi[f](b, s) \sim a^{\alpha(x_0)}, \quad (b, a) \in l, s \rightarrow 0^+ \quad (10.21)$$

Figure 10.2 displays the function $f(x) = K(x - x_0)^{0.4}$ with a singular point at x_0 . The Hölder exponent at x_0 is equal to 0.4. In Figure 10.3, we display the wavelet transform of $f(x)$ with a wavelet ψ which is the first derivative of a Gaussian. In Figure 10.4, we display $\log_2|T_\psi[f](x, s)|$ as a function of $\log_2(s)$.

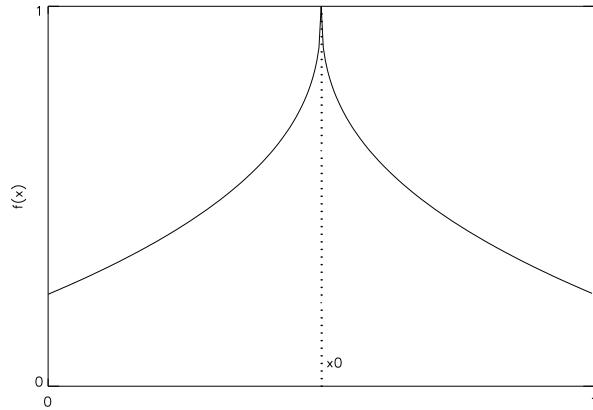


Figure 10.2: Function $f(x) = K(x - x_0)^{0.4}$ [8].

When we compute the slope of the curve $\log_2|T_\psi[f](x, s)|$ versus $\log_2(s)$ along a maxima line which converges at x_0 , we obtain an estimation of the Hölder exponent (in this case $\alpha(x_0) \approx 0.4$ which corresponds to the theoretical value).

10.3.2 Wavelet transform of multifractal signals

The estimation of the Hölder exponent by this method becomes inaccurate in the case of multifractal signals [8]. We need to use a more global method. One can define the wavelet-based partition function by

$$Z(q, s) = \sum_{b_i} |T_\psi[\mu](b_i, s)|^q \quad (10.22)$$

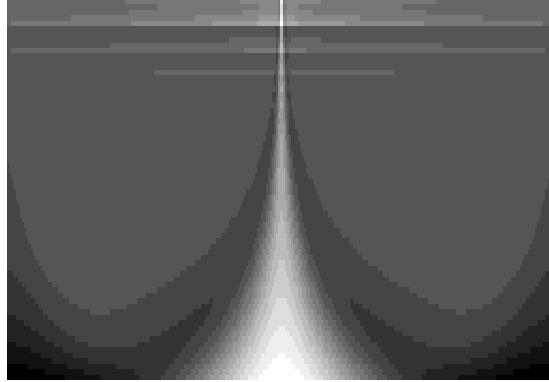


Figure 10.3: Wavelet transform of $f(x)$ with a wavelet ψ which is the first derivative of a Gaussian. The small scales are at the top. The maxima line converges to the singularity point at x_0 .

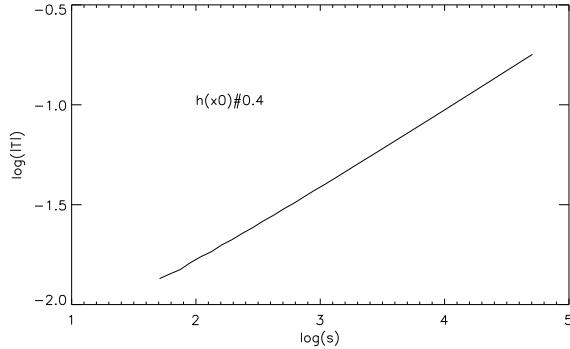


Figure 10.4: Estimation of the Hölder exponent $\alpha(x_0)$ at x_0 by computing the slope of $\log_2 |T_\psi[f](x, s)|$ versus $\log_2(s)$ along a maxima line which converges to x_0 .

where $(b_i, s)_i$ are all local maxima at scale s .

Let $\tau(q)$ be the scaling exponent. We can prove that we have

$$Z(q, s) \sim a^{\tau(q)} \quad (10.23)$$

We can then calculate the singularity spectrum $D(\alpha)$ by its Legendre transformation

$$D(\alpha) = \min_q (q\alpha - \tau(q)) \quad (10.24)$$

This method is called the *Wavelet Transform Modulus Maxima* (WTMM).

10.3.3 Numerical applications of WWTM method

The calculation of the singularity spectrum of signal f proceeds as follows:

- compute the wavelet transform and the modulus maxima $T_\psi[f]$ for all (s, q) . We chain all maxima across scale lines of maxima
- compute the partition function $Z(q, s) = \sum_{b_i} |T_\psi[f](b_i, s)|^q$
- compute $\tau(q)$ with $\log_2 Z(q, s) \approx \tau(q) \log_2(s) + C(q)$
- compute $D(\alpha) = \min_q (q\alpha - \tau(q))$

The triadic Cantor set

Definition : The measure associated with the triadic Cantor set is defined by $f(x) = \int_0^x d\mu$ where μ is the uniform measure lying on the triadic Cantor set described in section 10.1.1. To compute $f(x)$, we used the next recursive function called the Devil's Staircase function (which looks like a staircase whose steps are uncountable and infinitely small, see Figure 10.5).

$$f(x) = \begin{cases} p_1 f(3x) & \text{if } x \in [0, \frac{1}{3}] \\ p_1 & \text{if } x \in [\frac{1}{3}, \frac{2}{3}] \\ p_1 + p_2 f(3x - 2) & \text{if } x \in [\frac{2}{3}, 1] \end{cases} \quad (10.25)$$

This is a continuous function that increases from 0 to 1 on $[0,1]$. The recursive construction of $f(x)$ implies that $f(x)$ is self-similar.

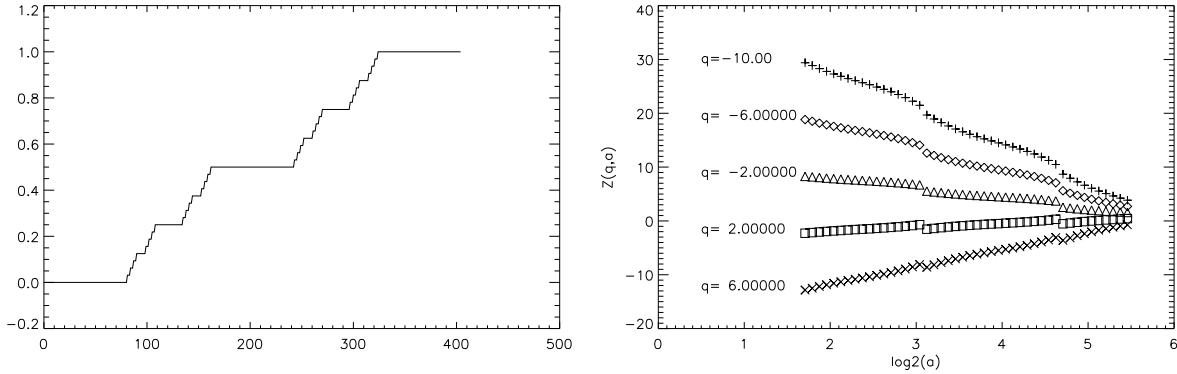


Figure 10.5: Classical Devil's Staircase function (associated with triadic Cantor set) with $p_1 = 0.5$ and $p_2 = 0.5$ (left) and partition function $Z(q, s)$ for several values of q (right). The wavelet transform is calculated with ψ equal to the first derivative of a Gaussian.

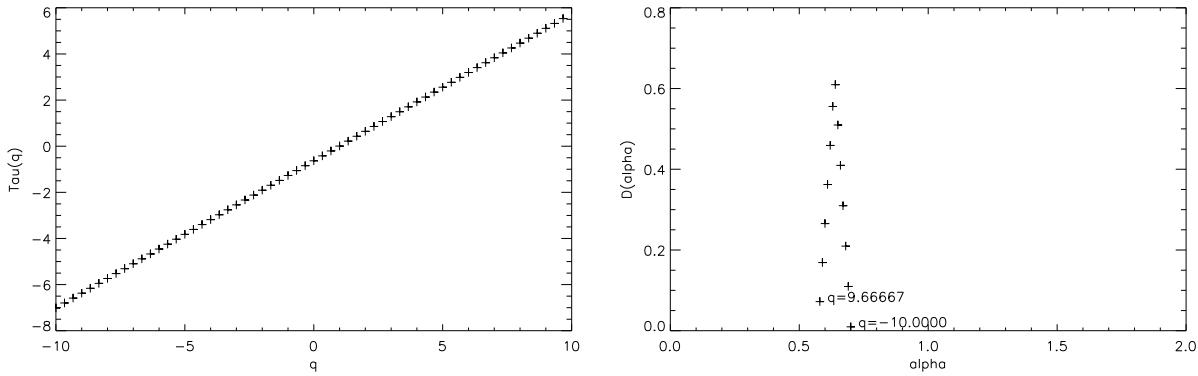


Figure 10.6: Scaling Exponent estimation $\tau(q)$ (left) and Singularity Spectrum $D(\alpha)$ (right). The Scaling Exponent curve corresponds to the theoretical curve $\tau(q) = (q - 1) \log_2(2) / \log_2(3)$. Points of the Singularity Spectrum where $q \neq \infty(\max q)$ and $q \neq -\infty(\min q)$ are reduced to a single point ($\alpha = \log_2(2) / \log_2(3)$, $D(\alpha) = \log_2(2) / \log_2(3)$). This point corresponds to the Hausdorff dimension of the triadic Cantor Set.

The generalized Devil's Staircase with $p_1 = 0.4$ and $p_2 = 0.6$

One can prove [8] that the theoretical singularity spectrum $D(\alpha)$ of the generalized Devil's Staircase function $f(x) = \int_0^x d\mu$ verifies the following:

- The singular spectrum is a convex curve with a maximum value α_{max} which corresponds to the fractal dimension of the support of the measure (μ).
- The theoretical support of $D(\alpha)$ is reduced at the interval $[\alpha_{min}, \alpha_{max}]$:

$$\begin{cases} \alpha_{min} = \min\left(\frac{\ln p_1}{\ln(1/3)}, \frac{\ln p_2}{\ln(1/3)}\right) \\ \alpha_{min} = \min\left(\frac{\ln p_1}{\ln(1/3)}, \frac{\ln p_2}{\ln(1/3)}\right) \end{cases} \quad (10.26)$$

Figure 10.7 displays the generalized Devil's Staircase and its partition function $Z(q, s)$. In Figure 10.8, we can see the Scaling exponent and the Singularity spectrum. This one is in perfect "accord" with the theoretical values: bell curve, $D(\alpha)_{max} = \log_2(2)/\log_2(3)$, $\alpha_{min} \approx 0.47$ and $\alpha_{max} \approx 0.83$

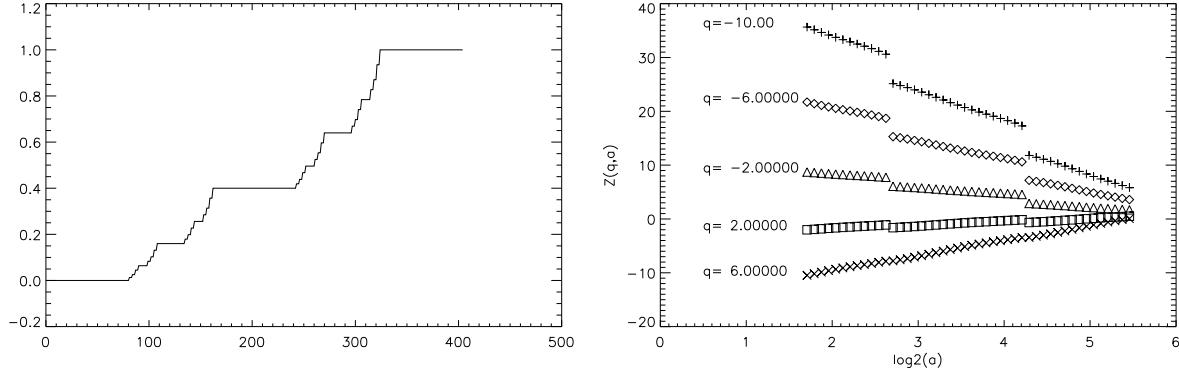


Figure 10.7: Devil's Staircase function with $p_1 = 0.4$ and $p_2 = 0.6$ (left) and partition function $Z(q, s)$ for several values of q (right). The wavelet transform is calculated with ψ equal to the first derivative of a Gaussian.

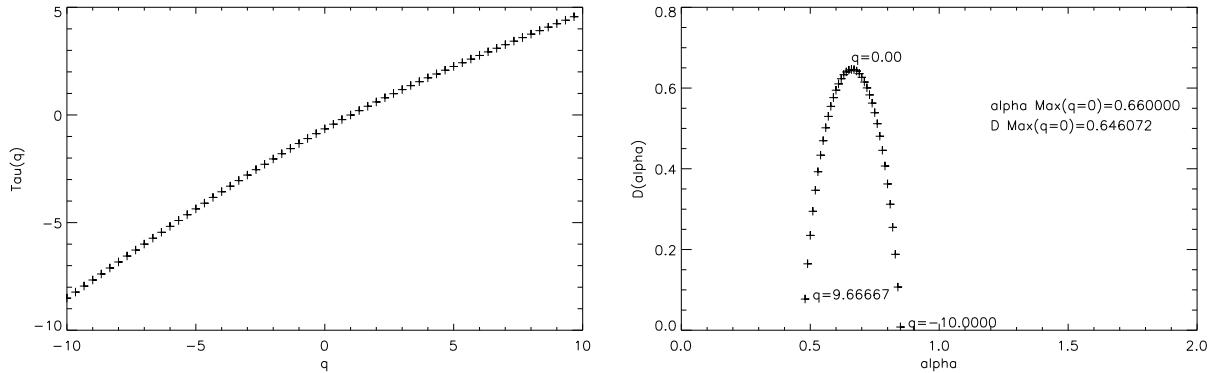


Figure 10.8: Scaling Exponent estimation $\tau(q)$ (left) and Singularity Spectrum $D(\alpha)$ (right). The theoretical maximum value of $D(\alpha)$ is obtained for $\alpha_{maxD} = \log_2(2)/\log_2(3)$ and $D(\alpha_{maxD}) = \log_2(2)/\log_2(3)$.

10.4 Program

10.4.1 Devil's Staircase: `mf1d_create_ds`

Program `mf1d_create_ds` creates a Devil's Staircase function lying on the Cantor sets measure. If the “-b” option is set, a border is added, which is equal to 0 on the left side, and to 1 on the right side. The size of the border is $\frac{N}{3}$ where N is the number of points. Hence the output signal size becomes $N + \frac{2N}{3} = \frac{5N}{3}$. It is recommended to choose a number of points which can be divided by 3.

USAGE: `mf1d_create_ds options image out`

where options are

- **`[-n number_of_points]`**
Number of points of Devil's Staircase. Default is 243.
- **`[-p Prob1]`**
Coefficient p_1 of the first interval of the Devil's Staircase. p_2 is equal to $1 - p_1$. Default value is 0.5.
- **`[-b]`**
Add a border. Default is no.
- **`[-v]`**
Verbose. Default is False.

Examples:

- `mf1d_create_ds NameOut`
Create a Devil's Staircase with all default values (243 points, probabilities equal to 0.5, ...).
- `mf1d_create_ds -p 0.4 NameOut`
Ditto, but probabilities p_1 and p_2 are respectively equal to 0.4 and 0.6.

10.4.2 Chain maxima wavelet coefficients: `mf1d_chain`

Program `mf1d_chain` calculates the wavelet transform of a signal, finds its maxima wavelet coefficients, and chains them. Finally, chains which do not respect some criteria are deleted. Two files are created: the first contains the maxima, and the second the skeleton.

USAGE: `mf1d_chain options signal.in`

where options are

- **`[-t type_of_transform]`**
0: Gaussian derivative wavelet transform
1: Mexican hat wavelet transform
Default is 0.
- **`[-r min_length_chain]`**
Remove chains with length < `min_length`. Default is 0.
- **`[-s dynamique(%)]`**
Remove points of the skeleton map if level < `dynamique * max(current scale)`. Default is 0.
- **`[-v]`**
Verbose

Wavelet transform file name is **WTMM_signal.in.fits**.

Skeleton file name is **WTMMSkel_signal.in.fits**.

Example:

- `mf1d_chain Data01.fits`
Create the files **WTMM_Data01.fits** and **WTMMSkel_Data01.fits**.

10.4.3 Partition function: `mf1d_repart`

Program *mf1d_repart* computes the partition function. It needs the two output files of the *mf1d_chain* program (i.e. `WTMMxxx.fits` and `WTMMSkelxxx.fits`), and creates three files:

- **q_FileNameOut.fits**: which contains the one-dimensional array Q of the q values for which the partition function is calculated.
- **s_FileNameOut.fits**: which contains the one-dimensional array S of the $\log_2 s$ values for which the partition function is calculated.
- **Z_FileNameOut.fits**: which contains the partition functions. It is a two dimensional array. $Z(i, j)$ is the partition function value for $q = Q(i)$ and $s = S(j)$ (i indexing the x-axis, and j the y-axis).

USAGE: `mf1d_repart options MaxFileName MaxSupFileName FileNameOut`

where options are

- **[-a q_minq]**
Value min of vector q . Default is -10.00.
- **[-b q_max]**
Value max of vector q . Default is +10.00.
- **[-c number_of_q]**
Number of values of vector q . Default is 20.

$Z(q, s)$ file name is **Z_FileNameOut.fits**

q file name is **q_FileNameOut.fits**

s file name is **s_FileNameOut.fits**.

Example:

- `mf1d_chain Data01.fits`
`mf1d_repart WTMM_Data01.fits WTMMSkel_Data01.fits Repart.fits`
Create the files **Z_Data01.fits**, **q_Data01.fits**, and **s_Data01.fits**.

10.4.4 Multifractal analysis: `mf_analyse`

Program *mf_analyse* realizes the multifractal analysis. It computes the scaling exponent estimation $\tau(q)$ and the singularity spectrum $\alpha(q)$. It needs the three files created by *mf1d_repart*. Four files are created:

- **X_Tau.fits**: which contains the x-coordinates of the $\tau(q)$ curve.
- **Y_Tau.fits**: which contains the y-coordinates of the $\tau(q)$ curve.
- **X_FracSpectrum.fits**: which contains the x-coordinates of the singularity spectrum curve (i.e. α values).
- **Y_FracSpectrum.fits**: which contains y-coordinates of the singularity spectrum curve (i.e. $D(\alpha)$ values).

USAGE: `mf_analyse options FileNameIn`

where options are

- **`[-d α_{min}]`**
Value min of vector α . Default is 0.00.
- **`[-e α_{max}]`**
Value max of vector α . Default is 1.00.
- **`[-f number_of_α]`**
Number of values of vector α . Default is 20.
- **`[-m ind_scale_min]`**
ind scale min for computing τ . Default is 30.
- **`[-M ind_scale_max]`**
ind scale max for computing τ . Default is scale_max-2.

$Z(q, s)$ are read in **Z_FileNameIn.fits** file

q are read in **q_FileNameIn.fits** file

s are read in **s_FileNameIn.fits** file .

Example:

- `mf1d_chain Data01.fits`
`mf1d_repart WTMM_Data01.fits WTMMskel_Data01.fits Repart`
`mf_analyse Repart`
Create the files **X_Tau.fits**, **Y_Tau.fits**, **X_FracSpectrum.fits**, **Y_FracSpectrum.fits**, and **Y_TauTheo.fits**.

10.4.5 Examples

Fractal analysis of Devil's Staircase function

Create Devil's Staircase function with $p_1 = .6$ and $p_2 = .4$.

```
mf1d_create_ds -p .4 -b DSC3
```

Compute the wavelet transform **WTMM_DSC3.fits**, the skeleton **WTMMskel_DSC3.fits** (default transform: ψ is the first derivative of a Gaussian).

```
mf1d_chain DSC3
```

Compute the partition function $Z(q, s)$ for $q \in [-10., 10.]$ with 60 values, write the $Z(q, s)$ to file **Z_DSC3.fits**, the s (scale) file to **s_DSC3.fits** and the q file to **q_DSC3.fits**.

```
mf1d_repart -c 60 WTMM_DSC3 WTMMskel_DSC3 DSC3
```

Compute the scaling exponent $\tau(q)$, write it in **X_Tau.fits** and **Y_Tau.fits**, compute the singularity spectrum $\alpha(q)$, write it in **X_FracSpectrum.fits** and **Y_FracSpectrum.fits** (100 values are used for α).

```
mf_analyse -f 100 DSC3
```

Fractal analysis

Compute the wavelet transform WTMM_DSC3.fits, the skeleton WTMMskel_DSC3.fits (use the Mexican hat for ψ)

```
mf1d_chain -t 1 signal
```

Compute the partition function $Z(q, s)$ for $q \in [-5., 5.]$ with 100 values, write the $Z(q, s)$ in file Z_signal.fits, the s (scale) file in s_signal and the q file in q_signal.

```
mf1d_repart -a -5. -b 5. -c 100 WTMM_signal WTMMskel_signal sigout
```

Compute the scaling exponent $\tau(q)$, write it in X_Tau.fits and Y_Tau.fits, compute the singularity spectrum $\alpha(q)$, write it in X_FracSpectrum.fits and Y_FracSpectrum.fits ($\alpha \in [0, 2]$, 100 values are used).

```
mf_analyse -d 0. -e 2. -f 100 sigout
```

Chapter 11

IDL Routines

11.1 Introduction

A set of routines has been developed in IDL. Starting IDL using the script program *mre* allows the user to get the multiresolution environment, and all routines described in the following can be called. An online help facility is also available by invoking the *mrh* program under IDL.

11.2 IDL Tools

11.2.1 del_pattern

Suppress a pattern in an image. A pattern is considered as a peak in Fourier space. So it can be iteratively eliminated.

USAGE: `output = del_pattern(Image, n_iter=n_iter, pattern=pattern, disp=disp, NSigma=NSigma, NbrScale=NbrScale)`

where

- *Image* is an IDL 2D array.
- *n_iter* is the number of iterations. Default is 3.
- if *disp* set, then results are displayed during the iterations.
- *output* is an IDL 2D array. It contains the input image, without the pattern.
- *pattern* is an IDL 2D array. It is equal to the difference between *Image* and *output*.
- *NSigma* is a real value (default is 3.0).
- *NbrScale* is the number of scales used in the wavelet transform.

11.2.2 block_trans

This routine separates the image into blocks of size *BlockSize* \times *BlockSize* and computes inside each block the mean and the variance. The variance versus mean plot gives information about the type of noise in the data.

USAGE: `block_trans, Image, TabVar=TabVar, TabMean=TabMean, Plot=Plot, BlockSize=BlockSize`

where

- *Image* is an IDL 2D array.
- *TabVar* is an output keyword and is the array of variances.
- *TabMean* is an output keyword and is the array of means.
- *Plot* is an input keyword. If set, the curve variance versus mean is plotted.
- *BlockSize* is the size of the blocks. Default is 8.

11.2.3 delete

Suppress a file in working directory.

USAGE: **delete, filename**

11.2.4 im_smooth

Smooth an image, taking into account the border.

USAGE: **im_smooth, Im_in, Im_out, method=method, Step_Trou=Step_Trou, Border=Border, WinSize=WinSize**

where

- *Im_in* is the input 2D IDL array.
- *Im_out* is the output 2D IDL smoothed array.
- *method*: keyword specifying the method for smoothing. Available methods are “linear”, “bspline”, or “median”. Default is “median”.
- *Step_Trou*: 2^{Step_Trou} is the distance between two adjacent pixels (default is 0).
- *Border* defines the way the borders of the image must be managed. Available methods are: “cont” ($Im_{in}(-i, -j) = Im_{in}(0, 0)$), “mirror” ($Im_{in}(-i, -j) = Im_{in}(i, j)$), “zero” ($Im_{in}(-i, -j) = 0$), “nobord” (the borders of the image are not smoothed).
- *WinSize* is the window size (only used if method = “median”). Default is 3.

11.2.5 Tikhonov

Deconvolve an image using Tikhonov regularization. This program does not shift the maximum of the PSF to the center. The minimized functional is:

$$J(O) = \| I - P * O \| + \alpha \| H * O \| \quad (11.1)$$

where *O* is the unknown object, *P* the point spread function (PSF), *I* the data and *H* a high pass filter (the Laplacian).

USAGE: **Result = TIKHONOV(Img, Psf, residu=residu, niter=niter, CvgParam=CvgParam, RegulParam=RegulParam, NoRegul=NoRegul)**

where

- *Img* is an input 2D IDL array, i.e. the image to deconvolve.
- *Psf* is an input 2D IDL array, i.e. point spread function.
- *niter* is the input number of iterations. Default is 10.

- *CvgParam* is the convergence parameter. Default is 0.1.
- *RegulParam* is the regularization parameter (i.e. α).
- *NoRegul* if set, no regularization is performed.
- *Residu* is an output 2D IDL array containing the residual: residual = Imag - Psf*Result.

If the keyword NoRegul is set, then no regularization is done, and the algorithm becomes a simple one step gradient method.

Examples:

- Result = TIKHONOV(Imag, Psf)
Deconvolve an image with all default options.
- Result = TIKHONOV(Imag, Psf)
Same example, but impose the number of iterations to be 30.
- Result = TIKHONOV (Imag, Psf, niter=30)
Deconvolution by the one step gradient method, without any regularization, with 30 iterations.
- Result = TIKHONOV (Imag, Psf, niter=30, /NoRegul)

11.3 Noise Related Routines

11.3.1 poisson_image

Create an image with Poisson-distributed values around the mean values IMAGE, using Knuth's "Algorithm Q" (D.E. Knuth, The Art of Computer Programming, Volume 2, "Seminumerical Algorithms", Addison-Wesley, 1969, p. 117). This routine has been written by James Hamill (Siemens Medical Systems, 2501 N. Barrington Rd., Hoffman Estates, IL 60195-7372), and has been inserted in the MR/1 package.

USAGE: `output = poisson_image (image[, seed])`

where *image* is a numeric array (byte, integer, long, or float) of arbitrary dimensionality. This is the array of values around which values in the result will be Poisson-distributed. *seed* is a longword seed for the random number generator. If this is not supplied, the value -123456789L is used for generating the first random value.

11.3.2 poisson_to_gauss

Transforms the data with Poisson noise into a new set of data with Gaussian noise using the generalized Anscombe transform.

$$T(x) = \frac{2}{gain} \sqrt{gain * x + \frac{3}{8} * gain^2 + \sigma^2 - gain * mean} \quad (11.2)$$

where

- *gain* = gain of the CCD
- σ = standard deviation of the readout noise
- *mean* = mean of the read out noise

For pure Poisson noise, *gain*=1, *mean*=0, and σ =0. If the *inv* keyword is set, then the inverse transform is applied:

$$T(x) = \frac{x^2}{4} * gain - \frac{\frac{3}{8} * gain^2 + \sigma^2 - gain * mean}{gain} \quad (11.3)$$

USAGE: `output = poisson_to_gauss (Data, poisson=poisson, inv=inv)`

- *Data*: IDL array: data
- *poisson*: float array (*poisson*(0) = gain, *poisson*(1)= σ , and *poisson*(2) = mean). By default, *poisson*=[1.,0.,0.].
- *inv*: If set, the inverse transform is applied.

11.3.3 sigma_clip

Returns the noise standard deviation obtained by k-sigma.

USAGE: `output = sigma_clip(Data, sigma_clip=sigma_clip, mean=mean)`

where

- *output*: standard deviation estimated by k-sigma clipping.
- *Data*: IDL array (input data).
- *sigma_clip*: number of iterations. Default value is 3.
- *mean*: If mean is set, the mean of the data (taking into account outliers) is returned.

11.3.4 get_noise

Find the standard deviation of white Gaussian noise in the data.

USAGE: `SigmaNoise = get_noise(Data, Niter=Niter)`

Data is an IDL array (1D, 2D, or 3D), and (*Niter*) is a keyword which fixes the number of iterations used by the sigma clipping. Default value is 3.

11.4 Display

11.4.1 tvlut

Display an image with a zoom factor, and the LUT is displayed. The zoom factor is calculated automatically in order to visualize the image in a window of size 320×320 (for an image 32×32 , the zoom factor is $320/32 = 10$). An offset is automatically calculated (or is set by keywords) in order to center the image in the IDL window.

USAGE: `tvlut, Data, Depx=Depx, Depy=Depy`

where

- *Data*: 2D IDL array (input data) to visualize.
- *Depx*: offset in x (default is 50).
- *Depy*: offset in y (default is 80).

11.4.2 xdisp

XDISP is a widget program for image analysis. Several operations can be carried out by using the mouse and pressing buttons:

- QUIT: quit the application
- LOAD: load a FITS image.
- LUT: modify the LUT.
- PROFILE: examine rows or columns.
- CURSOR: examine pixel values. If the image format is FITS and if the header contains astrometric position, then the pixel position in the sky is given (right ascension and declination).
- HISTO: plot the histogram.
- CONTOURS: Contours of the image (isophotes).
- 3D VISU: three-dimensional representation of the image.
- INFO: print the min, max, mean, sigma of the image.
- FFT: compute the Fourier transform and display either the power spectrum, the phase, the real part or the imaginary part.
- PAN: make a zoom. Zoom factors are 2,4,8,1/2,1/4,1/8

USAGE: **xdisp, Data [, FitsHeader]**

Data is a two-dimensional IDL array, and *FitsHeader* (string array) is an optional parameter for astrometric information (in FITS format).

11.4.3 x3d

X3D is a widget program for cube analysis. Several operations can be carried out by using the mouse and buttons:

- QUIT: quit the application
- Temporal Cut: A temporal cut is displayed.
- Horizontal Cut: A horizontal cut is displayed.
- Vertical Cut: A vertical cut is displayed.
- Frame Number: When the user enters a value V followed by carriage return the image Cube(*,*,V) is displayed.
- Window Size: Define the number of elements plotted.
- Slider Frame Number: When the user moves the slider, the corresponding frame is displayed.

USAGE: **x3d, Data, from=from, to=to**

Data is an IDL 3D array. The keywords *from* and *to* allow the user to define a subcube, which can be analysed separately with x3d.

Example:

```
IDL> HELP,MY_CUBE
      MY_CUBE           INT      = Array(32, 32, 100)
IDL> X3D, my_cube,from=[0,21,50],to=[20,49,99]
```

11.4.4 xdump

Save the selected window in a Postscript file. The file name is “idl.ps”.

USAGE: **xdump**, **FILENAME=filename**, **WIN=win**, **LANDSCAPE=landscape**, **SCALE=scale**, **TITLE=title**, **GIF=gif**

where keywords are

- *filename*: filename of output Postscript file (default is “idl.ps”)
- *win*: identifier of IDL window (default is active window)
- *landscape*: set landscape orientation
- *title*: add the title at the upper left corner of window
- *scale*: specify a scale to be applied to the entire graph
- *gif*: store the data in GIF format instead of PS format. Default title in this case “idl.gif”.

11.5 Multiresolution Routines (1D)**11.5.1 mr1d_atrou**

Computes a multiresolution transform of a 1D Signal by the à trous algorithm.

USAGE: **mr1d_atrou**, **Signal**, **WaveTrans**, **Nscale**, **mirror=mirror**, **linear=linear**

where

- *Signal*: input one-dimensional IDL array.
- *WaveTrans*: output two-dimensional array (wavelet transform).
- *Nscale*: number of scales (input parameter).
- *mirror*: if set, then mirroring is used at the border.
- *linear*: if set, then a linear wavelet function is used. If not set, then a B-spline wavelet function is used.

11.5.2 mr1d_pavemed

Computes a multiresolution transform of a 1D Signal by the multiresolution median algorithm.

USAGE: **mr1d_pavemed**, **Signal**, **MedTrans**, **Nscale**, **cont=cont**

where

- *Signal*: input one-dimensional IDL array.
- *MedTrans*: output two-dimensional array (multiresolution transform).
- *Nscale*: number of scales (input parameter).
- *cont*: if set, consider the data as constant beyond the borders.

11.5.3 mr1d_minmax

Computes a multiresolution transform of a 1D Signal by the min-max algorithm.

USAGE: `mr1d_minmax, Signal, MinMaxTrans, Nscale`

where

- *Signal*: input one-dimensional IDL array.
- *MinMaxTrans*: output two-dimensional array (multiresolution transform).
- *Nscale*: number of scales (input parameter).

11.5.4 mr1d_pyrmed

Computes a multiresolution transform of a 1D Signal by the pyramidal median algorithm.

USAGE: `mr1d_pyrmed, Signal, MedTrans, Nscale, TabNp=TabNp, interp=interp, cont=cont`

where

- *Signal*: input one-dimensional IDL array.
- *MedTrans*: output two-dimensional array (multiresolution transform).
- *Nscale*: number of scales (input parameter).
- *TabNp*: Number of pixels at each scale of the multiresolution transform (*MedTrans*(0:*TabNp*(*j*)-1) are the multiresolution coefficients of the scale *j*).
- *interp*: if set, each scale is interpolated to the size of the input signal.
- *cont*: if set, consider the data as constant outside the borders.

11.5.5 mr1d_paverec

Reconstruct a signal from its multiresolution transform (à trous algorithm or multiresolution median transform).

USAGE: `mr1d_paverec, MR_Trans, Rec_Signal, Adjoint=Adjoint, mirror=mirror, nosmooth=nosmooth`

where

- *MR_Trans*: input two-dimensional IDL array (multiresolution transform).
- *Rec_Signal*: output one-dimensional IDL array. This is the reconstructed signal.
- *Adjoint*: if set, the adjoint operator is used for the reconstruction.
- *nosmooth*: if set, the last scale is not used.
- *interp*: if set, and if adjoint is set, then mirroring is used beyond the borders.

11.5.6 mr1d_pyrrec

Reconstruct a signal from its pyramidal multiresolution transform.

USAGE: `mr1d_pyrrec, MR_Trans, Rec_Signal`

where

- *MR_Trans*: input two-dimensional IDL array (multiresolution transform).
- *Rec_Signal*: output one-dimensional IDL array. This is the reconstructed signal.

11.5.7 mr1d_pyrinterp

Interpolate each scale of a 1D pyramidal multiresolution transform to the size of the original signal.

USAGE: `mr1d_pyrinterp, MR_Trans`

where *MR_Trans* is a pyramidal multiresolution transform.

11.5.8 mr1d_tabcoef

Return the pre-computed table for noise behavior in the multiresolution transform.

USAGE: `output = mr1d_tabcoef(TypeTransform)`

where *TypeTransform* is the type of multiresolution transform. Available types are “atrou”, “pyrmed” or “pavmed”. The output is a float array which gives the standard deviation of the noise at each scale, when we apply a multiresolution transform to a signal following a Gaussian distribution with a standard deviation equal to 1.

11.5.9 mr1d_trans

One-dimensional wavelet transform. 19 transforms are available (see 9.2.1), which are grouped in 5 classes:

- Class 1: no decimation (transform 1 to 7 and 11 to 14).
- Class 2: pyramidal transform (transform 8 to 10).
- Class 3: orthogonal transform (15 and 16).
- Class 4: Wavelet packets (17 and 18).
- Class 5: Wavelet packets from the à trous algorithm (19).

Depending on the class, the transform does not contain the same number of pixels, and the data representation differs.

USAGE: `MR1D_Trans, Signal, Result, OPT=OPT, BAND=BAND, NODEL=NODEL`

where

- *Signal*: input one-dimensional IDL array.
- *Result*: output IDL structure.
- *Opt*: string which contains the different options (see the *mr1d_trans* C++ program).
- *Band*: if set, a tag per band is created in the output structure.
- *NodeL*: if set, the two created file are not deleted:
 xx_result.fits: wavelet coefficients file
 xx_info.fits: information about the transform

Result is IDL structure which contains the wavelet transform. The structure contains the following tags:

- *N_BAND*: float; number of bands in the transfrom
- *INFO*: 2D float array (Array[2, NbrBand+3])

```

info[0,0] = transform number
info[1,0] = number of scales
info[0,1] = transform class number (5 classes)
info[1,1] = number of bands
    it is not equal to the number of scales
    for wavelet packets transform.
info[0,2] = number of pixels
info[1,2] = lifting scheme type
info[0,3] = type of filter
info[1,3] = type of normalization
for i=4 to Number_of_bands + 3
    info[0,i] = number of pixels in the band i
    info[1,i] = position number of the pixel of the band
If a user filter file is given (i.e. -T 6,filename),
    with a filename of $L$ characters, $L$ lines are added
    to the array:
    info[1,Number_of_bands + 4] = number of characters of
        the filter file name
    for i=Number_of_bands+4 to Number_of_bands+4+L-1
        info[0,i] = ascii number of the ith character.

```

If a user filter file is given (i.e. -T 6,filename), with a filename of L characters, L lines are added to the array:

```

info[1,Number_of_bands + 4] = number of characters of the filter file name
for i=Number_of_bands+4 to Number_of_bands+4+L-1
info[0,i] = ascii number of the ith character.

```

- FROM: 1D array (NbrBand); position of the first pixel
- TO: 1D array (NbrBand); position of the last pixel
- COEF: 1D or 2D FLOAT array; wavelet coefficients for non-redundant transform (15,16,17), it is a 1D array
for other transform, it is 2D array
class 1 and 5: coeff[*,:i] = band i (i in [0..NbrBand-1])
class 2: coeff[0:to[i],:] = band i
class 3 and 4: coeff[from[i]:to[i]] = band i
- Bandi: if BAND keyword is set, the array coef is also split into bands:

```

BAND1   : band 1
BAND2   : band 2
...
BANDi   : band i

```

11.5.10 mr1d_recons

Reconstruct a one-dimensional signal from its wavelet transform.

USAGE: MR1D_RECONS, WT_Struct, result

where

- *WT_Struct*: input IDL structure (obtained using IDL MR1D_TRANS program)..
- *Result*: output one-dimensional IDL array.

11.5.11 mr1d_filter

Filter a 1D signal:

USAGE: mr1d_filter, Signal, Result, Opt=Opt

where

- *Signal*: input one-dimensional IDL array.
- *Result*: output one-dimensional IDL array (filtered signal).
- *Opt*: string which contains the different options (see the *mr1d_filter* C++ program).

11.6 Spectral Analysis

11.6.1 mr1d_continuum

Estimate the continuum of a 1D signal.

USAGE: output = mr1d_continuum(Signal,
Nscale,Sigma=Sigma,median=median,mirror=mirror)

where

- *Signal*: 1D IDL array. Input spectrum.
- *Output*: 1D IDL array. Estimated continuum.
- *Nscale*: number of scales.
- *median*: if set use multiresolution median transform
- *mirror*: if set, use mirroring at borders.
- *niter*: number of iterations. Default is 5.
- *Sigma*: noise estimation (output keyword).

11.6.2 mr1d_optical_depth

Estimate the optical depth of a spectrum.

USAGE: output = mr1d_optical_depth(Signal, Nscale, Sigma=Sigma)

where

- *Signal*: 1D IDL array. Input spectrum.
- *Output*: 1D IDL array. Estimated optical depth.
- *Nscale*: number of scales.
- *Sigma*: noise estimation (output keyword).

11.6.3 mr1d_detect

USAGE: `mr1d_detect, Signal, result, OPT=OPT, print=print, tabobj=tabobj, NbrObj=NbrObj, nodel=nodel, tabw=tabw`

where

- *Signal*: input signal.
- *result*: output signal (sum of all detected objects).
- *OPT*: string which contains the different options (see the *mr1d_detect* C++ program).
- *print*: if set, information about each detected object is printed.
- *tabobj*: IDL structure which contains the information about the objects. For each object, we have
 - NumObj: Object number
 - Pos: Position in the signal (in wavelength units).
 - Sigma: standard deviation of the band (in wavelength units).
 - Fwhm: Full-width at half-maximum (in wavelength units).
 - PosPix: Position in the signal (in pixel units).
 - SigmaPix: standard deviation of the band (in pixel units).
 - Flux: integrated flux of the band.
- *NbrObj*: number of detected objects
- *nodel*: if set, the created files (by the *mr1d_detect* C++ program) are not deleted.
- *tabw*: wavelength array

11.7 Multiresolution Routines (2D)

11.7.1 mr_transform

Computes a multiresolution transform of an image. If the keyword *MR_File_Name* is set, a file is created which contains the multiresolution transform. A multiresolution file has a “.mr” extension, and if the parameter file name does not specify this, then the extension is added to the file name. Result is stored in the *DataTransf*. Depending on the options, *DataTransf* can be a cube, an image, or an IDL structure. This routine calls the C++ executable *mr_transform*. The keyword “*OPT*” allows to pass to the executable all options described in section 3.2.1.

USAGE: `mr_transform, Data, DataTransf, MR_File_Name=MR_File_Name, OPT=Opt`

Examples:

- `mr_transform, I, Output, MR_File_Name='result.mr'`
Compute the multiresolution of the image *I* with default options (i.e. à trous algorithm with 4 scales). The result is stored in the file “*result.mr*”.
- `mr_transform, I, Output, MR_File_Name='result_pyr_med', OPT='t 10 -n 5'`
Compute the multiresolution of *I* by using the pyramidal median algorithm with 5 scales. The result is stored in the file “*result_pyr_med.mr*”.

11.7.2 xmr_transform

Computes a multiresolution transform of an image. Parameters are chosen through a widget interface. Creates a file which contains the multiresolution transform of the image. The transform is made by calling the routine `mr_transform`. If there is an image parameter the user doesn't need to load an image with the interface.

USAGE: `xmr_transform, TransfData, input=input`

TransfData is the output transform, and *input* is the input image.

11.7.3 mr_info

Calculate statistical information about the wavelet transform of an image. This routine calls the C++ executable `mr_info`. The keyword “OPT” allows to pass to the executable all options described in section 3.2.1. The output is the 2D IDL array $T(*, 0 : 4)$ with the following syntax:

- $T(j,0)$ = standard deviation of the j th band
- $T(j,1)$ = skewness of the j th band
- $T(j,2)$ = kurtosis of the j th band
- $T(j,3)$ = minimum of the j th band
- $T(j,4)$ = maximum of the j th band

USAGE: `mr_info, Data, TabStat, OPT=Opt, nodel=nodel, NameRes=NameRes`

Example:

- `mr_info, I, Stat`

Compute the multiresolution of the image *I* with default options (i.e. à trous algorithm with 4 scales) and calculate statistical information about each band.

11.7.4 mr_extract

Extract a scale from a multiresolution transform. This routine calls the C++ executable `mr_extract`. The keyword “OPT” allows all options described in section 3.2.3 to be passed to the executable.

USAGE: `mr_extract, Multiresolution_File_Name, ScaleImage, OPT=Opt`

where *Multiresolution_File_Name* is a string which contain the name of multiresolution file (“.mr”), and *ScaleImage* is the output image (IDL 2D array).

11.7.5 mr_read

Read a multiresolution file (extension “.mr”). If the multiresolution transform is a cube, the output is a three-dimensional array. If it is an image the output is a two-dimensional IDL array. If it is a pyramidal transform or a half-pyramidal transform, we have 3 different outputs:

1. an image containing several subimages (flag raw set)
2. a cube of interpolated or rebinned images (flag interpol set)
3. a structure containing several subimages (default)

USAGE: `output = mr_read(filename, interpol=interpol, raw=raw, debug=debug)`

where

- *filename*: string which contains the file name of the multiresolution transform (extension “.mr”).
- *interp*: integer (for pyramidal transform only)
 - 0: the output will not be interpolated
 - 1: the output will be rebinned
 - 2: the output will be interpolated
- *raw*: if set the output overwrites the input FITS file (for pyramidal transform only).
- *debug*: if set, the routine is verbose

11.7.6 mr_compare

Comparison between a reference image and an image or a sequence of images. The comparison is carried out on the multiresolution scales. If *Ima_Or_Cube* is a cube, the processing is repeated on each image of the cube (*,*,i). For each image *ima_i* of the cube *Ima_Or_Cube*

- we compute the wavelet transform *WaveRef* of *ImaRef*
- we compute the wavelet transform *WaveIma* of *ima_i*
- we calculate the correlation at each scale s:
if *TabCorrel*[s,i] = 1 then *WaveRef*[*,*,s] = *WaveIma*[*,*,s] are identical
else *TabCorrel*[s,i] is less than 1, and some differences exist.
- we calculate the RMS at each scale:
TabRMS[s,i] = sigma(*WaveRef*[*,*,s] - *WaveIma*[*,*,s])
- we calculate the normalized SNR at each scale (in dB):
TabSNR[s,i] = 10 alog10 (mean(*WaveRef*[*,*,s]²) / *TabRMS*[s,i]²)

This routine calls the C++ executable *mr_transform* described in section 3.2.1.

USAGE: **mr_compare, ImaRef, Ima_Or_Cube, TabCorrel, TabRMS, TabSNR,**
Nscale=Nscale, plot=plot, title=title

where

- *ImaRef*: IDL 2D array. Reference image.
- *Ima_Or_Cube*: IDL 2D or 3D array. Image or sequence of images to be compared.
- *Nscale*: number of scales for the comparison
- *plot*: if set, then plot the results
- *title*: is set, add the title to the plots
- *TabCorrel*: 1D or 2D IDL array: output correlation table
- *TabRMS*: 1D or 2D IDL array: output RMS table
- *TabSNR*: 1D or 2D IDL array: output SNR table (in dB)

Example

```
mr_compare, imaref, ima1, tc, tr, ts, /plot, title='Comparison ImaRef-Ima1'
Comparison of the images imaref and ima1.
```

11.7.7 mr_background

Estimate the background of an image.

USAGE: `output=mr_background(Image, nscale=nscale, border=border)`

If border is set, the background is estimated from the border of the image. If not, the pyramidal median transform is used with *nscale* scales (default is 3) by calling the C++ executable `mr_background` described in section 3.3.6.

11.7.8 mr_filter

Filter an image by using a multiresolution transform. This routine is calling the C++ executable `mr_filter`. The keyword “OPT” allows all options described in section 4.2 to be passed to the executable.

USAGE: `mr_filter, Data, FilterData, opt=opt`

Data is an image (2D IDL array), and *FilterData* is the result of the filtering.

11.7.9 im_deconv

Deconvolve an image by standard methods. This routine calls the C++ executable `im_deconv`. The keyword “OPT” allows all options described in section 4.6 to be passed to the executable.

USAGE: `im_deconv, Data, PSF, DeconvData, opt=opt`

Examples:

- `im_deconv, Imag, Psf, Result`
deconvolve an image with all default options (gradient method).
- `im_deconv, Imag, Psf, Result, OPT='i 30 -e 0'`
same example, but impose the number of iterations to be 30.
- `im_deconv, Imag, Psf, Result, OPT='d 4 -i 30'`
deconvolution by the one step gradient method, without any regularization, with 30 iterations.

11.7.10 mr_deconv

Deconvolve an image by using the multiresolution support. This routine calls the C++ executable `mr_deconv`. The keyword “OPT” allows all options described in section 4.6 to be passed to the executable.

USAGE: `mr_deconv, Data, PSF, DeconvData, opt=opt`

Examples:

- `mr_deconv, Imag, Psf, Result`
deconvolve an image with all default options (Richardson-Lucy method + regularization in wavelet space by using the à trous algorithm, etc.).
- `mr_deconv, Imag, Psf, Result, OPT='i 30 -e 0'`
same example, but impose the number of iterations to be 30.
- `mr_deconv, Imag, Psf, Result, OPT='d 2 -i 30'`
deconvolution by the one step gradient method, without any regularization, with 30 iterations.

11.7.11 mr_detect

Detect the sources in an image by using a multiresolution transform. This routine calls the C++ executable `mr_detect`. The keyword “OPT” allows all options described in section 6.1 to be passed to the executable.

USAGE: `mr_detect, imag, result, OPT=OPT, print=print, tabobj=tabobj,`
`NbrObj=NbrObj, nodel=nodel, RMS=RMS`

where

- *imag*: input image.
- *result*: output image (sum of all detected objects).
- *OPT*: option keyword.
- *print*: if set, information about each detected object is printed.
- *tabobj*: IDL structure which contains the information about the objects. For each object, we have
 - *ScaleObj*: scale where the object is detected.
 - *NumObj*: object number
 - *PosX*: X coordinate in the image.
 - *PosY*: Y coordinate in the image.
 - *SigmaX*: standard deviation on first main axis of the object.
 - *SigmaY*: standard deviation on second main axis of the object.
 - *Angle*: angle between the main axis and x-axis.
 - *ValPixMax*: value of the maximum of the object.
 - *Flux*: integrated flux of the object.
 - *Magnitude*: magnitude of the object.
 - *ErrorFlux*: flux error.
 - *SNR_ValMaxCoef*: signal to noise ratio of the maximum of the wavelet coefficient.
 - *SNR_Obj*: signal to noise ratio of the object.
 - *PosCoefMaxX*: X coordinate of the maximum wavelet coefficient
 - *PosCoefMaxY*: Y coordinate of the maximum wavelet coefficient
- *NbrObj*: number of detected objects
- *nodel*: if set, the files created (by the program `mr_detect`) are not deleted.
- *RMS*: RMS image related to the input image.

Examples:

```
mr_detect, imag, result, tabobj=tabobj, NbrObj=NbrObj
detects all sources with default options in the image imag.
print, NbrObj
print the number of objects.
print, tabobj(0).PosX, tabobj(0).PosY
print the coordinates of the first object.
```

11.7.12 xlive

XLIVE is a widget program for large image analysis. The large image has to be compressed by mr_comp or mr_lcomp before being analyzed. Then the XLIVE data format is the MRC format. When an MRC file is read, an image at very low resolution is displayed and the user can improve the resolution of the image, or of a part the image, using the RESOLUP button. When RESOLUP is called, XLIVE reads from the MRC file the wavelet coefficient needed for improving the resolution. If the large image has been compressed by block (-C option), only the blocks needed are decompressed, and the image in memory has always a size compatible with the window size. The Dat parameter (if given) contains the last displayed image. If it is not given, the user can however get it using the global variable XIMA.

Several operations can be carried out by using the mouse and pressing buttons:

- QUIT: quit the application.
- LOAD: load a FITS image.
- LUT: modify the LUT.
- PROFILE: examine rows or columns.
- CURSOR: examine pixel values. If the image format is FITS and if the header contains astrometric position, then the pixel position in the sky is given (right ascension and declination).
- HISTO: plot the histogram.
- CONTOURS: contours of the image (isophotes).
- 3D VISU: three-dimensional representation of the image.
- INFO: print the min, max, mean, sigma of the image.
- FFT: compute the Fourier transform and display either the power spectrum, the phase, the real part or the imaginary part.
- PAN: make a zoom. Zoom factors are 2,4,8,1/2,1/4,1/8
- RESOLUP: Improve the image resolution. If the new image size is greater than the window size, the user must click in the area he/she wishes to see, and only this part of the image will be decompressed.
- RESOLDOWN: decrease the resolution.
- RESOL: goto a given resolution.

USAGE: `xlive, [Dat,], FILEName=FileName, WindowSize=WindowSize`

Dat is an output two-dimensional IDL array, and *FileName* (string array) is an optional parameter containing the MRC filename to be read.

Appendix A: The Lifting Scheme

Introduction

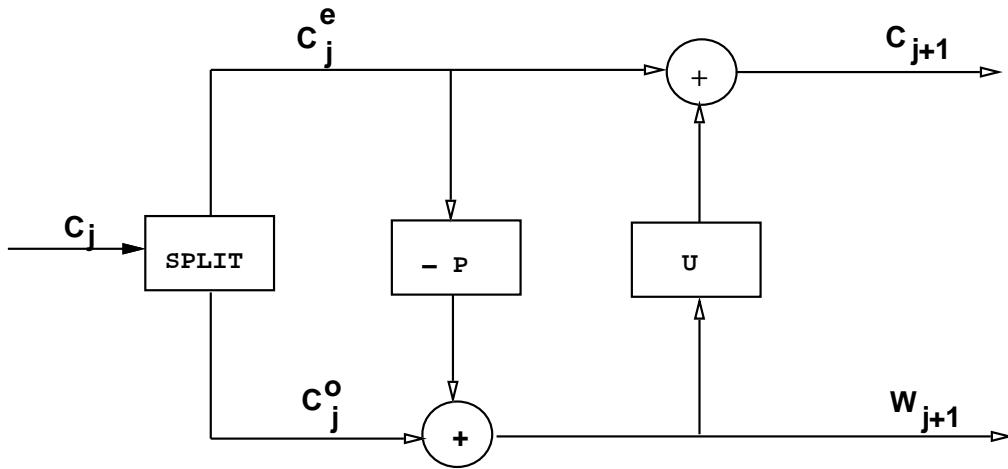


Figure 11.1: The lifting scheme – forward direction.

The lifting scheme [201] is a flexible technique that has been used in several different settings, for easy construction and implementation of traditional wavelets [201], and of second generation wavelets [200] such as spherical wavelets [164].

Its principle is to compute the difference between a true coefficient and its prediction:

$$w_{j+1,l} = c_{j,2l+1} - \mathcal{P}(c_{j,2l-2L}, \dots, c_{j,2l-2}, c_{j,2l}, c_{j,2l+2}, \dots, c_{j,2l+2L}) \quad (11.4)$$

A pixel at an odd location $2l+1$ is then predicted using pixels at even locations.

The transformation is done in three steps:

1. Split the signal into even and odd number samples:

$$\begin{aligned} c_{j+1,l} &= c_{j,2l} \\ w_{j+1,l} &= c_{j,2l+1} \end{aligned}$$

2. Set

$$w_{j+1,l} = w_{j+1,l} - \mathcal{P}(c_{j+1,l})$$

3. Set

$$c_{j+1,l} = c_{j+1,l} + \mathcal{U}(w_{j+1,l})$$

where \mathcal{U} is the update operator.

The reconstruction is obtained by:

$$\begin{aligned} c_{j,2l} &= c_{j+1,l} - \mathcal{U}(w_{j+1,l}) \\ c_{j,2l+1} &= w_{j+1,l} + \mathcal{P}(c_{j+1,l}) \end{aligned}$$

Example of transforms

Haar wavelet via lifting

The Haar transform can be performed via the lifting scheme by taking the predict operator equal to the identity, and an update operator which halves the difference. The transform becomes:

$$\begin{aligned} w_{j+1,l} &= w_{j+1,l} - c_{j+1,l} \\ c_{j+1,l} &= c_{j+1,l} + \frac{w_{j+1,l}}{2} \end{aligned}$$

All computation can be done in place. Every wavelet transform can be written via lifting.

Linear wavelets via lifting

The identity predictor used before is correct when the signal is constant. In the same way, we can use a linear predictor which is correct when the signal is linear. The predictor and update operators are now:

$$\begin{aligned} \mathcal{P}(c_{j-1,l}) &= \frac{1}{2}(c_{j-1,l} + c_{j-1,l+1}) \\ \mathcal{U}(w_{j-1,l}) &= \frac{1}{4}(w_{j-1,l-1} + w_{j-1,l}) \end{aligned}$$

It is easy to verify that:

$$c_{j-1,l} = -\frac{1}{8}c_{j,2l-2} + \frac{1}{4}c_{j,2l-1} + \frac{3}{4}c_{j,2l} + \frac{1}{4}c_{j,2l+1} - \frac{1}{8}c_{j,2l+2}$$

which is the bi-orthogonal Cohen-Daubechies-Feauveau [41] wavelet transform.

Integer wavelet transform

When the input data are integer values, the wavelet transform no longer consists of integers. For lossless coding, it is useful to have a wavelet transform which produces integer values. We can build an integer version of every wavelet transform [29]. For instance, denoting $\lfloor x \rfloor$ as the largest integer not exceeding x , the integer Haar transform (also called “S” transform) can be calculated by:

$$\begin{aligned} w_{j+1,l} &= c_{j,2l+1} - c_{j,2l} \\ c_{j+1,l} &= c_{j,2l} + \lfloor \frac{w_{j+1,l}}{2} \rfloor = c_{j+1,l} + \lfloor \frac{w_{j+1,l}}{2} \rfloor \end{aligned} \tag{11.5}$$

while the reconstruction is

$$\begin{aligned} c_{j,2l} &= c_{j+1,l} - \lfloor \frac{w_{j+1,l}}{2} \rfloor \\ c_{j,2l+1} &= w_{j+1,l} + c_{j,2l} \end{aligned} \tag{11.6}$$

More generally, the lifting operators for an integer version of the wavelet transform are:

$$\begin{aligned}\mathcal{P}(c_{j+1,l}) &= \lfloor \sum_k p_k c_{j+1,l-k} + \frac{1}{2} \rfloor \\ \mathcal{U}(w_{j+1,l}) &= \lfloor \sum_k u_k w_{j+1,l-k} + \frac{1}{2} \rfloor\end{aligned}$$

The linear integer wavelet transform is

$$\begin{aligned}w_{j+1,l} &= w_{j+1,l} - \lfloor \frac{1}{2}(c_{j+1,l} + c_{j+1,l+1}) + \frac{1}{2} \rfloor \\ c_{j+1,l} &= c_{j+1,l} + \lfloor \frac{1}{4}(w_{j+1,l-1} + w_{j+1,l}) + \frac{1}{2} \rfloor\end{aligned}$$

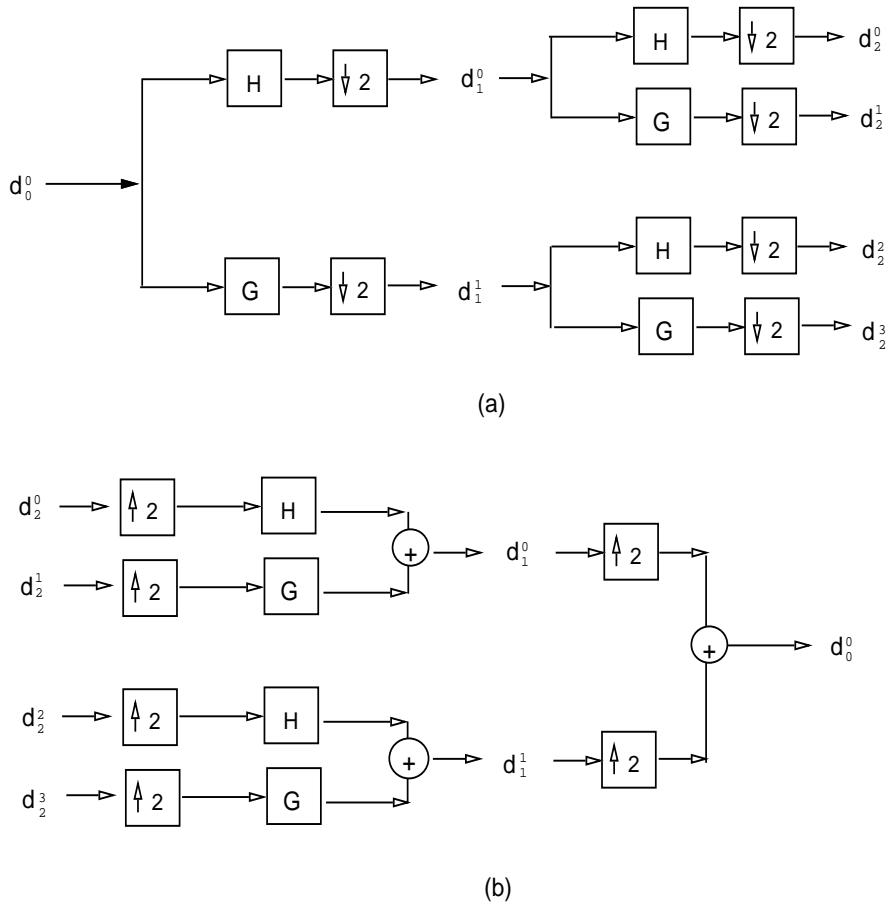
Even if there is no filter that consistently performs better than all the other filters on all images, the following one performs generally better than others [29]:

$$\begin{aligned}w_{j+1,l} &= w_{j+1,l} - \lfloor \frac{1}{2}(c_{j+1,l} + c_{j+1,l+1}) + \frac{1}{2} \rfloor \\ c_{j+1,l} &= c_{j+1,l} + \lfloor \frac{1}{4}(w_{j+1,l-1} + w_{j+1,l}) + \frac{1}{2} \rfloor\end{aligned}$$

More filters can be found in [29].

Appendix B: Wavelet Packets

Wavelet packets were introduced by Coifman, Meyer and Wickerhauser [44]. Instead of dividing only the approximation space, as in the standard orthogonal wavelet transform, detail spaces are also divided.



$\downarrow 2$	Keep one sample out of two	\times	Convolution with the filter X
$\uparrow 2$	Put one zero between each sample		

Figure 11.2: Wavelet packet scheme: (a) forward direction, (b) reconstruction.

Figure 11.2 shows the wavelet packet scheme.

Appendix C: Nonlinear Pyramidal Wavelet Transform (WT-PMT)

One of the advantages of the pyramidal wavelet transform over the pyramidal median transform (PMT) is the ability to have robust noise estimation in the different scales, while the advantage of the PMT is a better separation of the structures in the scales. Using the PMT, a strong structure (like a bright star, cosmic rays, bad pixels, etc.) will not be spread over all scales as when using a wavelet transform. In fact, when there is no signal in a given region, a wavelet transform would be better, and if a strong signal appears, it is the PMT that we would like to use. So the idea arises to try to merge both transforms, and to adapt the analysis at each position and at each scale, depending on the amplitude of the coefficient we measure.

A possible algorithm to perform this on an image I is the following:

1. Let $c_j = I$, and $j = 0$.
2. Filter the the image c_j by the median: we get m_j
3. Set $c_j^* = c_j$ and $d_j = c_j - m_j$
4. For all pixels k do
if $|d_j(k)| > k\sigma_j$ then $c_j^*(k) = m_j(k)$
5. Smooth c_j^* by a B₃-spline: we get b_j
6. Set $w_j = c_j - b_j$
7. $c_{j+1} = dec(b_j)$ (where the decimation operation, dec, entails 1 pixel replacing each 2×2 subimage). Let $j = j + 1$ and return to step 2 if $j < N_s$ (N_s being the number of scales).

In this algorithm, the linear filtering relative to the wavelet transform is not applied to the strong features contained in the image. Indeed, significant pixel values are detected at step 4, and are replaced by the median. Regions containing no bright object are treated as if the pyramidal wavelet transform is used. The parameter k used at step 4 must be large enough in order to be sure that noise is not filtered by the median ($k = 5$ seems high enough). As this transform merges the wavelet transform and the PMT, we call it the PMWT transform. PMWT takes more time than the PMT, but this algorithm has the advantages of the PMT without the drawbacks. The reconstruction is the same as for the PMT.

Appendix D: Half Pyramidal Wavelet Transform

At each iteration of the pyramidal transform, there is a smoothing and a decimation of the image. But the smoothing is not strong enough to reduce the cut-off frequency by a factor two. The decimation then violates Shannon's theorem. To avoid this problem, which may create artifacts, Bijaoui [20] proposed not to decimate the image at the first iteration of the algorithm. This means that the two first scales have the same size as the original image (instead of just one in the standard pyramidal transform). Then the two first iterations are identical as in the à trous algorithm. For the following iterations, there is a decimation and the filtering is done as for the second iteration (i.e. by taking pixels interspersed by one pixel for the filtering).

Part II

MR/2 : Multiscale Entropy and Applications

Chapter 12

Multiscale Entropy Theory

12.1 Entropy and Image Restoration

12.1.1 Introduction

The term “entropy” is due to Clausius (1865), and the concept of entropy was introduced by Boltzmann into statistical mechanics in order to measure the number of microscopic ways that a given macroscopic state can be realized. Shannon (1948) [166] founded the mathematical theory of communication when he suggested that the information gained in a measurement depends on the number of possible outcomes out of which one is realized. Shannon also suggested that the entropy can be used for maximization of the bit transfer rate under a quality constraint. Jaynes (1957) [99] proposed to use the entropy measure for radio interferometric image deconvolution, in order to select between a set of possible solutions that which contains the minimum of information or, following his entropy definition, that which has maximum entropy. In principle, the solution verifying such a condition should be the most reliable. A great deal of work has been carried out in the last 30 years on the use of the entropy for the general problem of data filtering and deconvolution [1, 24, 27, 76, 88, 144, 152, 171, 213, 132, 133].

Traditionally information and entropy are determined from events and the probability of their occurrence. Signal and noise are basic building-blocks of signal and data analysis in the physical sciences. Instead of the probability of an event, we are led to consider the probabilities of our data being either signal or noise.

Observed data Y in the physical sciences are generally corrupted by noise, which is often additive and which follows in many cases a Gaussian distribution, a Poisson distribution, or a combination of both. Other noise models may also be considered. Using Bayes’ theorem to evaluate the probability distribution of the realization of the original signal X , knowing the data Y , we have

$$p(X|Y) = \frac{p(Y|X).p(X)}{p(Y)} \quad (12.1)$$

$p(Y|X)$ is the conditional probability distribution of getting the data Y given an original signal X , i.e. it represents the distribution of the noise. It is given, in the case of uncorrelated Gaussian noise with variance σ^2 , by:

$$p(Y|X) = \exp \left\{ - \sum_{pixels} \frac{(Y - X)^2}{2\sigma^2} \right\} \quad (12.2)$$

The denominator in equation 12.1 is independent of X and is considered as a constant (stationary noise). $p(X)$ is the a priori distribution of the solution X . In the absence of any information on the solution X except its positivity, a possible course of action is to derive the probability of X from its entropy, which is defined from information theory.

The main idea of information theory [166] is to establish a relation between the received information and the probability of the observed event [13]. If we note $\mathcal{I}(E)$ the information related to the event E , and p the probability of this event happening, then we consider that

$$\mathcal{I}(E) = f(p) \quad (12.3)$$

Then we assume the two following principles:

- The information is a decreasing function of the probability. This implies that the more information we have, the less will be the probability associated with one event.
- Additivity of the information. If we have two independent events E_1 and E_2 , the information $\mathcal{I}(E)$ associated with the happening of both is equal to the addition of the information of each of them.

$$\mathcal{I}(E) = \mathcal{I}(E_1) + \mathcal{I}(E_2) \quad (12.4)$$

Since E_1 (of probability p_1) and E_2 (of probability p_2) are independent, then the probability of both happening is equal to the product of p_1 and p_2 . Hence

$$f(p_1 p_2) = f(p_1) + f(p_2) \quad (12.5)$$

Then we can say that the information measure is

$$\mathcal{I}(E) = k \ln(p) \quad (12.6)$$

where k is a constant. Information must be positive, and k is generally fixed at -1 .

Another interesting measure is the mean information which is denoted

$$H = - \sum_i p_i \ln(p_i) \quad (12.7)$$

This quantity is called the entropy of the system and was established by Shannon in 1948 [166].

This measure has several properties:

- It is maximal when all events have the same probability $p_i = 1/N_e$ (N_e being the number of events), and is equal to $\ln(N_e)$. It is in this configuration that the system is the most undefined.
- It is minimal when one event is sure. In this case, the system is perfectly known, and no information can be added.
- The entropy is a positive, continuous, and symmetric function.

If we know the entropy H of the solution (the next section describes different ways to calculate it), we derive its probability by

$$p(X) = \exp(-\alpha H(X)) \quad (12.8)$$

Given the data, the most probable image is obtained by maximizing $p(X|Y)$. Taking the logarithm of equation 12.1, we thus need to maximize

$$\ln(p(X|Y)) = -\alpha H(X) + \ln(p(Y|X)) - \ln(p(Y)) \quad (12.9)$$

The last term is a constant and can be omitted. Then, in the case of Gaussian noise, the solution is found by minimizing

$$J(X) = \sum_{pixels} \frac{(Y - X)^2}{2\sigma^2} + \alpha H(X) = \frac{\chi^2}{2} + \alpha H(X) \quad (12.10)$$

which is a linear combination of two terms: the entropy of the signal, and a quantity corresponding to χ^2 in statistics measuring the discrepancy between the data and the predictions of the model. α is a parameter that can be viewed alternatively as a Lagrangian parameter or a value fixing the relative weight between the goodness-of-fit and the entropy H .

For the deconvolution problem, the object-data relation is given by the convolution

$$Y = P * X \quad (12.11)$$

where P is the point spread function, and the solution is found (in the case of Gaussian noise) by minimizing

$$J(X) = \sum_{pixels} \frac{(Y - P * X)^2}{2\sigma^2} + \alpha H(X) \quad (12.12)$$

The way the entropy is defined is fundamental, because from its definition will depend the solution. The next section discusses the different approaches which have been proposed in the past.

12.1.2 The concept of entropy

We wish to estimate an unknown probability density $p(X)$ of the data. Shannon [166], in the framework of the information theory, has defined the entropy of an image X by

$$H_s(X) = - \sum_{k=1}^{N_b} p_k \log p_k \quad (12.13)$$

where $X = \{X_1, \dots, X_N\}$ is an image containing integer values, N_b is number of possible values which can take a given pixel X_k (256 for a 8 bits image), and p_k values are derived the histogram of X :

$$p_k = \frac{\#X_j = k}{N} \quad (12.14)$$

$\#X_j = k$ giving the number of pixels $X_j = k$.

If the image contains floating values, it is possible to build up the histogram L of values L_i , using a suitable interval Δ , counting up how many times m_k each interval $(L_k, L_k + \Delta)$ occurs among the N occurrences. Then the probability that a data value belongs to an interval k is $p_k = \frac{m_k}{N}$, and each data value has a probability p_k .

The entropy is minimum and equal to zero when the signal is flat, and increases when we have some fluctuations. Using this entropy in equation 12.10 leads to minimize:

$$J(X) = \frac{\chi^2}{2} + \alpha H_s(X) \quad (12.15)$$

It is a minimum entropy restoration method.

The trouble with this approach is that, because the number of occurrences is finite, the estimate p_k will be in error by an amount proportional to $m_k^{-\frac{1}{2}}$ [77]. The error becomes significant when m_k is small. Furthermore this kind of entropy definition is not easy to use for signal restoration, because the gradient of equation 12.15 is not easy to compute. For these reasons, other entropy functions are generally used. The main ones are:

- Burg [27]:

$$H_b(X) = - \sum_{k=1}^N \ln(X_k) \quad (12.16)$$

- Frieden [76]:

$$H_f(X) = - \sum_{k=1}^N X_k \ln(X_k) \quad (12.17)$$

- Gull and Skilling [88]:

$$H_g(X) = \sum_{k=1}^N X_k - M_k - X_k \ln\left(\frac{X_k}{M_k}\right) \quad (12.18)$$

where M is a given model, usually taken as a flat image

where N is the number of pixels, and k represents an index pixel.

Each of these entropies can be used, and they correspond to different probability distributions that one can associate with an image [144]. (See [76, 171] for descriptions). The last definition of the entropy has the advantage of having a zero maximum when X equals the model M . All

of these entropy measures are negative (if $X_k > 1$), and maximum when the image is flat. They are negative because an offset term is omitted which has no importance for the minimization of the functional. The fact that we consider that a signal has maximum information value when it is flat is evidently a curious way to measure information. A consequence is that we must now maximize the entropy if we want a smooth solution, and the probability of X must be redefined by:

$$p(X) = \exp(\alpha H(X)) \quad (12.19)$$

The sign has been inverted (see equation 12.8), which is natural if we want the best solution to be the smoothest. These three entropies, above, lead to the Maximum Entropy Method method (MEM), for which the solution is found by minimizing (for Gaussian noise)

$$J(X) = \sum_{k=1}^N \frac{(Y_k - X_k)^2}{2\sigma^2} - \alpha H(X) \quad (12.20)$$

These different entropy functions which have been proposed for image restoration have the property of being maximal when the image is flat, and of decreasing when we introduce some information. So minimizing the information is equivalent to maximizing the entropy, and this has led to the well known Maximum Entropy Method (MEM). For the Shannon entropy (which is obtained from the histogram of the data), this is the opposite. The entropy is null for a flat image, and increases when the data contains some information. So, if the Shannon entropy were used for restoration, this would lead to a Minimum Entropy Method.

In 1986, Narayan and Nityanda [144] compared several entropy functions, and finally concluded by saying that all were comparable if they have good properties, i.e. they enforce positivity, and they have a negative second derivative which discourages ripple. They showed also that results varied strongly with the background level, and that these entropy functions produced poor results for negative structures, i.e. structures under the background level (absorption area in an image, absorption band in a spectrum, etc.), and compact structures in the signal. The Gull and Skilling entropy gives rise to the difficulty of estimating a model. Furthermore it has been shown [24] that the solution is dependent on this choice.

The determination of the α parameter is also not an easy task and in fact it is a very serious problem facing the maximum entropy method. In the historic MAXENT algorithm of Skilling and Gull, the choice of α is such that it must satisfy the ad hoc constraint $\chi^2 = N$ when the deconvolution is achieved, N being the number of degrees of freedom of the system i.e. the number of pixels in image deconvolution problems. But this choice systematically leads to an under-fitting of the data [204] which is clearly apparent for imaging problems with little blurring. In reality, the χ^2 statistic is expected to vary in the range $N \pm \sqrt{2N}$ from one data realization to another. In the Quantified Maximum Entropy point of view [171], the optimum value of α is determined by including its probability $P(\alpha)$ in Bayes' equation and then by maximizing the marginal probability of having α , knowing the data and the model m . In practice, a value of α which is too large gives a resulting image which is too regularized with a large loss of resolution. A value which is too small leads to a poorly regularized solution showing unacceptable artifacts. Taking a flat model of the prior image softens the discontinuities which may appear unacceptable for astronomical images often containing stars and other point-like objects. Therefore the basic maximum entropy method appears to be not very appropriate for this kind of image which contains high and low spatial frequencies at the same time. Another point to be noted is a ringing effect of the maximum entropy method algorithm, producing artifacts around bright sources.

To solve these problems while still using the maximum entropy concept, some enhancements of the maximum entropy method have been proposed. Noticing that neighboring pixels of reconstructed images with MAXENT could have values differing a lot in expected flat regions [37], Gull and Skilling introduced the concepts of hidden image S and intrinsic correlation function C (Gaussian or cubic spline-like) in the Preblur MAXENT algorithm.

The ICF describes a minimum scale length of correlation in the desired image O which is achieved by assuming that

$$O = C * S \quad (12.21)$$

This corresponds to imposing a minimum resolution on the solution O . Since the hidden space image S is not spatially correlated, this can be regularized by the entropy

$$H_g(h) = \sum_{k=1}^N S_k - M_k - S_k \ln\left(\frac{S_k}{M_k}\right) \quad (12.22)$$

Since in astronomical images many scale lengths are present, the *Multi-channel Maximum Entropy Method*, developed by Weir [212, 213], uses a set of ICFs having different scale lengths, each defining a channel. The visible-space image is now formed by a weighted sum of the visible-space image channels O_j :

$$O = \sum_{j=1}^{N_c} p_j O_j \quad (12.23)$$

where N_c is the number of channels. Like in Preblur MAXENT, each solution O_j is supposed to be the result of the convolution between a hidden image S_j with a low-pass filter (ICF) C_j :

$$O_j = C_j * S_j \quad (12.24)$$

But such a method has several drawbacks:

1. The solution depends on the width of the ICFs [24].
2. There is no rigorous way to fix the weights p_j [24].
3. The computation time increases linearly with the number of pixels.
4. The solution obtained depends on the choice of the models M_j ($j = 1 \dots N_c$) which were chosen independently of the channel.

In 1993, Bontekoe et al. [24] used a special application of this method which they called Pyramid Maximum Entropy on infrared image data. The pyramidal approach allows the user to have constant ICF width, and the computation time is reduced. It is demonstrated [24] that all weights can be fixed ($p_j = 1$ for each channel).

This method eliminates the first three drawbacks, and gives better reconstruction of the sharp and smooth structures. But in addition to the two last drawbacks, a new one is added: as the images O_j have different sizes (due to the pyramidal approach), the solution O is built by duplicating the pixels of the subimages O_j of each channel. This procedure is known to produce artifacts due to the appearance of high frequencies which are incompatible with the real spectrum of the true image \hat{O} .

However this problem can be easily overcome by duplicating the pixels before convolving with the ICF, or expanding the channels using linear interpolation. Thus the introduction of the “pyramid of resolution” has solved some problems and brought lots of improvements to the classic maximum entropy method, but has also raised other questions. In order to derive the model from a physical value, Pantin and Starck [152] introduced the wavelet transform, and defined entropy as follows:

$$H(O) = \frac{1}{\sigma_I^2} \sum_{j=1}^l \sum_{k=1}^{N_j} \sigma_j (w_{j,k} - M_{j,k} - |w_{j,k}| \ln \frac{|w_{j,k}|}{M_{j,k}}) \quad (12.25)$$

where σ_I is the noise standard deviation in the data, l is the number of scales, and N_j is the number of samples in the band j ($N_j = N$ for the à trous algorithm). The multiscale entropy is the sum of the entropy at each scale.

The coefficients $w_{j,k}$ are wavelet coefficients, and we take the absolute value of $w_{j,k}$ in this definition because the values of $w_{j,k}$ can be positive or negative, and a negative signal contains also some information in the wavelet transform.

The advantage of such a definition of entropy is the fact we can use previous work concerning the wavelet transform and image restoration [139, 187, 181]. The noise behavior has already been studied in the wavelet transform and we can estimate the standard deviation of the noise σ_j at scale j . These estimates can be naturally introduced in our models m_j

$$M_{j,k} = k_m \sigma_j \quad (12.26)$$

The model M_j at scale j represents the value taken by a wavelet coefficient in the absence of any relevant signal and, in practice, it must be a small value compared to any significant signal value. Following the Gull and Skilling procedure, we take M_j as a fraction of the noise because the value of σ_j can be considered as a sort of physical limit under which a signal cannot be distinguished from the noise ($k_m = \frac{1}{100}$).

12.1.3 Conclusion

As described above, many studies have been carried out in order to improve the functional to be minimized. But the question which should be raised is: what is a good entropy for signal restoration?

Trying to answer, this corresponds to asking what is the information in the signal. We first assume that a signal X can be decomposed in several components:

$$X = S + B + N \quad (12.27)$$

where S is the signal of interest, B is the background, and N the noise.

The entropy should verify the following criteria [194]:

1. **The information in a flat signal is zero** ($S = 0, N = 0$ et $B = \text{Cst}$).
2. **The amount of information in a signal is independent of the background** ($H(X)$ is independent of B).
3. **The amount of information is dependent on the noise** ($H(X)$ is dependent of N). A given signal X doesn't furnish the same information if the noise N is high or small.
4. **The entropy must work in the same way for a pixel which has a value $B + \epsilon$, and for a pixel which has a value $B - \epsilon$.** $H(X)$ must be a function of the absolute value of S instead of S .
5. **The amount of information is dependent on the correlation in the signal.** If the signal S presents large features above the noise, it contains a lot of information. By generating a new set of data from S , by randomly taking the pixel values in S , the large features will evidently disappear, and this new signal will contain less information. But the pixel values will be the same as in S .

Fig. 12.1 illustrates the last point perfectly. The second image is obtained by distributing randomly the Saturn image pixel values, and the standard entropy definitions produce the same information measurement for both images. The concept of information becomes really subjective, or at least it depends on the application domain. Indeed, for someone who is not involved in image processing, the second image contains *less* information than the first one. For someone working on image transmission, it is clear that the second image will require more bits for lossless transmission, and from this point of view, he/she will consider that the second image contains *more* information. Finally, for data restoration, all fluctuations due to noise are not of interest, and do not contain relevant information. From this physical point of view, the standard definition of entropy seems badly adapted to information measurement in signal restoration.

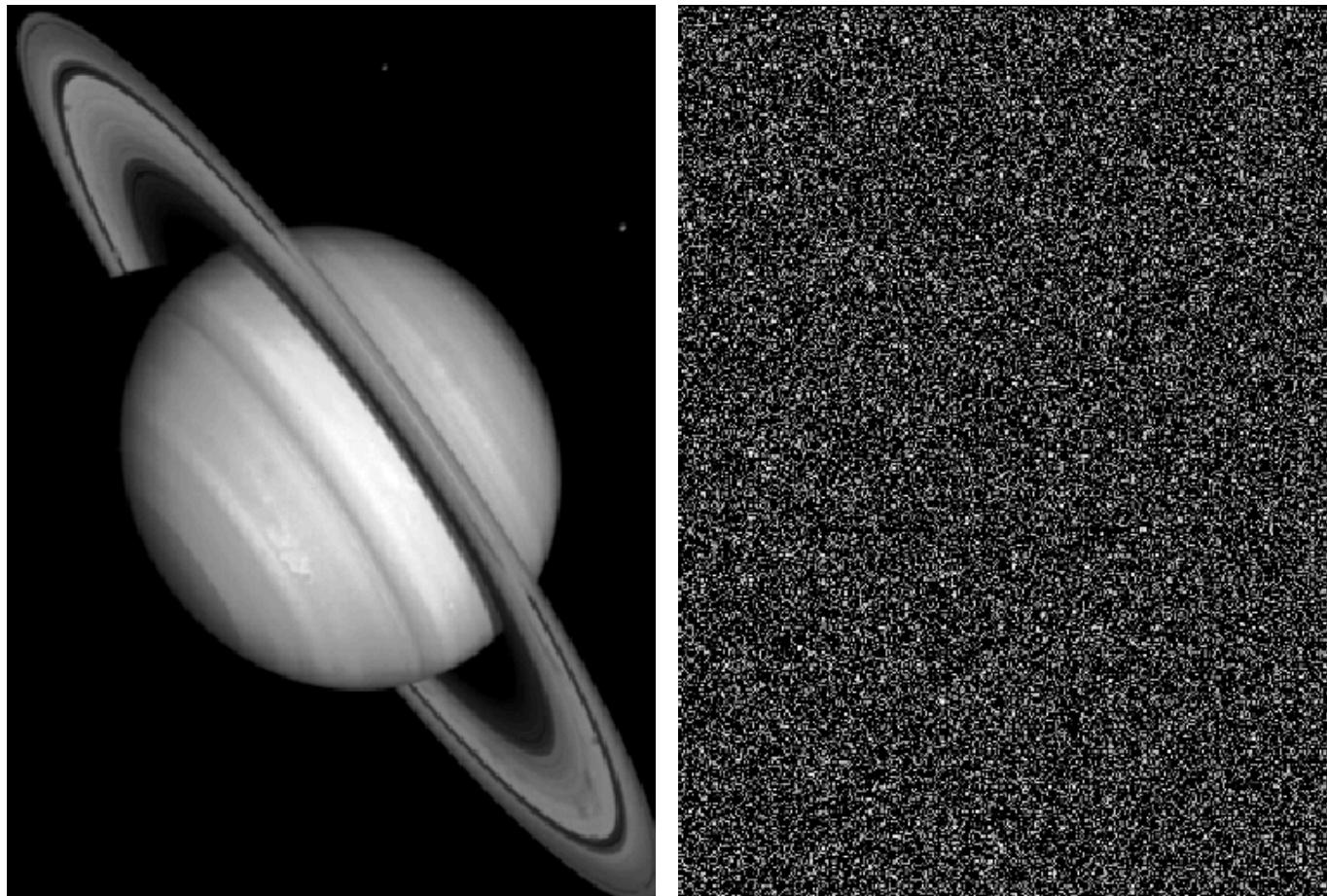


Figure 12.1: Saturn image (left) and the same data distributed differently (right). These two images have the same entropy, using any of the standard entropy definitions.

12.2 Entropy from Noise Modeling

12.2.1 Information and wavelet coefficient

In the case of signal restoration, the noise is the main problem. This means that we should not consider the probability of appearance of a pixel value in an image, but rather its probability of being due to the signal (or to the noise). If we consider a variable x which follows a probability distribution $p(x)$, we can define the information in x by $-\ln(p(x))$, and a signal S can be considered as a set of individual variables x_k (pixels), each of which follows the same probability distribution. Then the information contained in the data can be measured by $-\sum_{k=1}^N \ln(p(x_k))$. If X follows a Gaussian distribution with zero mean, we have

$$H(X) = \sum_{k=1}^N \frac{x_k^2}{2\sigma^2} + \text{Cst} \quad (12.28)$$

The energy gives a good measurement of information. But many of the required criteria are not fulfilled by using such an entropy (correlation between pixels, background-independence, etc.). It seems difficult to derive a good probability distribution from the pixel values which fulfill the entropy requirements.

This is not so for transformed data, especially when using the wavelet transform. This has already been done, in fact, for finding threshold levels in filtering methods by means of wavelet coefficient thresholding [184, 135, 65, 108, 38, 3]. Thus we must introduce the concept of multiresolution into our entropy. We will now consider that the information contained in some dataset is the sum of the information at different resolution levels j . The wavelet transform W of a signal by a fast algorithm contains a set of coefficients $w_{j,k}$ (j being the scale index), and a set of coefficients c_k representing the signal at a very low resolution (see [121, 191] for more information about the wavelet transform). If the number of scales is enough large, we can assume that the coefficients c_k furnish information only about the background, and not on the signal of interest. The entropy of X must be measured only from the wavelet coefficients $w_{j,k}$.

Due to the properties of the wavelet transform, the set w_j for all j has a zero mean. From noise modeling, we can derive the probability distribution in the wavelet space of a wavelet coefficient, assuming it is due to the noise. The entropy becomes

$$H(X) = \sum_{j=1}^l \sum_{k=1}^{N_j} h(w_{j,k}) \quad (12.29)$$

with $h(w_{j,k}) = -\ln p(w_{j,k})$. We will note in the following h for the entropy (or information) relative to a wavelet coefficient, and H for the multiscale entropy (MSE) of a signal or an image. For Gaussian noise, we get

$$H(X) = \sum_{j=1}^l \sum_{k=1}^{N_j} \frac{w_{j,k}^2}{2\sigma_j^2} + \text{Cst} \quad (12.30)$$

where σ_j is the noise at scale j . We see that the information is proportional to the energy of the wavelet coefficients. The higher a wavelet coefficient, then the lower will be the probability, and the higher will be the information furnished by this wavelet coefficient. As the constant has no effect in the solution calculation in restoration problems, we take the liberty to remove it in the following.

We can see easily that this entropy fulfills all the requirements of section 12.1.2. If we consider two signals S_1, S_2 , derived from a third one S_0 by adding noise:

$$\begin{aligned} S_1 &= S_0 + N_1(\sigma_1) \\ S_2 &= S_0 + N_2(\sigma_2) \end{aligned} \quad (12.31)$$

then we have:

$$\text{if } \sigma_1 < \sigma_2 \text{ then } H(S_1) > H(S_2) \quad (12.32)$$

and a flat image has zero entropy.

Our entropy definition is completely dependent on the noise modeling. If we consider a signal S , and we assume that the noise is Gaussian, with a standard deviation equal to σ , we won't measure the same information compared to the case when we consider that the noise has another standard deviation value, or if the noise follows another distribution. As for the Shannon entropy, the information increases with the entropy, and using such an entropy leads to a Minimum Entropy Method.

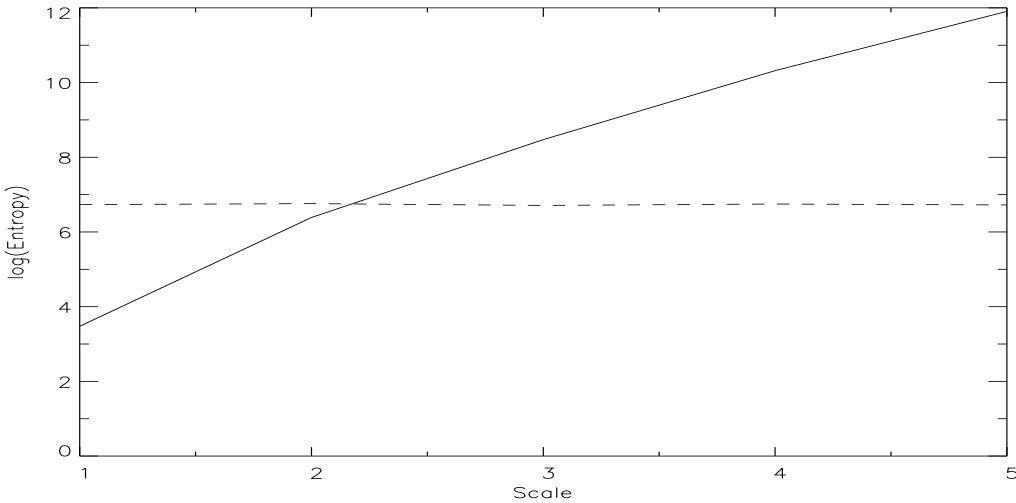


Figure 12.2: Multiscale entropy of the saturn image (continuous curve), and multiscale entropy of the scrambled image (dashed curve).

Figure 12.2 shows the information measure at each scale for both the saturn image and its scrambled version. The global information is the addition of the information at each scale. We see that for the scrambled image (dashed curve), the information-versus-scale curve is flat, while for the unscrambled saturn image, it increases with the scale.

12.2.2 Signal information and noise information

In the previous section, we have seen how it was possible to measure the information H related to a wavelet coefficient. Assuming the signal X is still composed of the three elements S, B, N ($X = S + F + B$, signal of interest, background, and noise), H is independent of B . But some of this information is spurious and undesirable and has been introduced via the noise N . To get the useful information, we must subtract out this spurious portion. Trying to decompose our information measure into two components, one (H_S) corresponding to the non-corrupted part, and another (H_N) to the corrupted part, we have

$$H(X) = H_S(X) + H_N(X) \quad (12.33)$$

We will define in the following H_S as the signal information, and H_N as the noise information. It must be clear that noise does not contain any information, and what we call noise information is a quantity which is measured as information by the multiscale entropy, and which is probably not informative to us.

If a wavelet coefficient is small, its value can be due to noise, and the information h relative to this single wavelet coefficient should be assigned to H_N . If the wavelet coefficient is high,

compared to the noise standard deviation, its value cannot be due to the noise, and h should be assigned to H_S . h can be distributed as H_N or H_S based on the probability $P_n(w_{j,k})$ that the wavelet coefficient is due to noise, or the probability $P_s(w_{j,k})$ that it is due to signal. We have $P_s(w_{j,k}) = 1 - P_n(w_{j,k})$. For the Gaussian noise case, we estimate $P_n(w_{j,k})$ that a wavelet coefficient is due to the noise by

$$\begin{aligned} P_n(w_{j,k}) = \text{Prob}(W > |w_{j,k}|) &= \frac{2}{\sqrt{2\pi}\sigma_j} \int_{|w_{j,k}|}^{+\infty} \exp(-W^2/2\sigma_j^2) dW \\ &= \text{erfc}\left(\frac{|w_{j,k}|}{\sqrt{2}\sigma_j}\right) \end{aligned}$$

For each wavelet coefficient $w_{j,k}$, we have to estimate now the fractions h_n and h_s of h which should be assigned to H_n and H_s . Hence signal information and noise information are defined by

$$\begin{aligned} H_s(X) &= \sum_{j=1}^l \sum_{k=1}^{N_j} h_s(w_{j,k}) \\ H_n(X) &= \sum_{j=1}^l \sum_{k=1}^{N_j} h_n(w_{j,k}) \end{aligned} \quad (12.34)$$

Note that $H_s(X) + H_n(X)$ is always equal to $H(X)$.

N1-MSE

A first approach for deriving h_s and h_n from P_s and P_n is to just consider P_s and P_n as weights on the information h . Then we have:

$$\begin{aligned} h_s(w_{j,k}) &= P_s(w_{j,k})h(w_{j,k}) \\ h_n(w_{j,k}) &= P_n(w_{j,k})h(w_{j,k}) \end{aligned} \quad (12.35)$$

and the noise and signal information in a signal are

$$\begin{aligned} H_s(X) &= \sum_{j=1}^l \sum_{k=1}^{N_j} h_s(w_{j,k}) \\ H_n(X) &= \sum_{j=1}^l \sum_{k=1}^{N_j} h_n(w_{j,k}) \end{aligned} \quad (12.36)$$

which leads for a Gaussian noise to:

$$\begin{aligned} H_s(X) &= \sum_{j=1}^l \sum_{k=1}^{N_j} \frac{w_{j,k}^2}{2\sigma_j^2} \text{erf}\left(\frac{|w_{j,k}|}{\sqrt{2}\sigma_j}\right) \\ H_n(X) &= \sum_{j=1}^l \sum_{k=1}^{N_j} \frac{w_{j,k}^2}{2\sigma_j^2} \text{erfc}\left(\frac{|w_{j,k}|}{\sqrt{2}\sigma_j}\right) \end{aligned} \quad (12.37)$$

We will refer to these functions by the name N1-MSE in the following.

N2-MSE

By the previous entropy measure, information relative to high wavelet coefficients is completely assigned to the signal. For a restoration, this allows us also to exclude wavelet coefficients with high signal-to-noise ratio (SNR) from the regularization. It leads to perfect fit of the solution with the data at scales and positions with high SNR. If we want to consider the information due to noise, even for significant wavelet coefficients, the noise information relative to a wavelet coefficient must be estimated differently. The idea for deriving h_s and h_n is the following: we imagine that the information h relative to a wavelet coefficient is a sum of small information components dh , each of them having a probability of being noise information, or signal information [194]. For example, for two coefficients u and w ($w > u$), with a Gaussian noise ($\sigma = 1$), the information relative to w is $h(w) = w^2$, and by varying u from 0 to w with a step du , the information $h(u)$ increases until it is equal to $h(w)$. When u becomes closer to w , the difference $w - u$ can be due to the noise, and the added information $dh(u) = h(u + du) - h(u)$ is contaminated by the noise. The idea is to weight $dh(u)$ with the probability that $w - u$ is due to the noise. Hence, h_n and h_s are calculated by:

$$h_n(w_{j,k}) = \int_0^{|w_{j,k}|} P_n(|w_{j,k}| - u) \left(\frac{\partial h(x)}{\partial x} \right)_{x=u} du \quad (12.38)$$

is the noise information relative to a single wavelet coefficient, and

$$h_s(w_{j,k}) = \int_0^{|w_{j,k}|} P_s(|w_{j,k}| - u) \left(\frac{\partial h(x)}{\partial x} \right)_{x=u} du \quad (12.39)$$

is the signal information relative to a single wavelet coefficient. For Gaussian noise, we have

$$\begin{aligned} h_n(w_{j,k}) &= \frac{1}{\sigma_j^2} \int_0^{|w_{j,k}|} u \operatorname{erfc}\left(\frac{|w_{j,k}| - u}{\sqrt{2}\sigma_j}\right) du \\ h_s(w_{j,k}) &= \frac{1}{\sigma_j^2} \int_0^{|w_{j,k}|} u \operatorname{erf}\left(\frac{|w_{j,k}| - u}{\sqrt{2}\sigma_j}\right) du \end{aligned} \quad (12.40)$$

and the noise and signal information in a signal are

$$\begin{aligned} H_s(X) &= \sum_{j=1}^l \sum_{k=1}^{N_j} h_s(w_{j,k}) \\ H_n(X) &= \sum_{j=1}^l \sum_{k=1}^{N_j} h_n(w_{j,k}) \end{aligned} \quad (12.41)$$

We will refer to these functions by the name N2-MSE in the following.

Equations 12.35 and 12.34 lead to two different ways to regularize a signal. The first requires that we use all the information which is furnished in high wavelet coefficients, and leads to an exact preservation of the flux in a structure. If the signal presents high discontinuities, artifacts can appear in the solution due to the fact that the wavelet coefficients located at the discontinuities are not noisy, but have been modified like noise. The second equation doesn't have this drawback, but a part of the flux of a structure (compatible with noise amplitude) can be lost in the restoration process. It is however not as effective as in the standard maximum entropy methods.

LOG-MSE

The multiscale entropy function used in [152] (we call it LOG-MSE in the following) can be considered in our framework if h is defined by:

$$h(w_{j,k}) = \frac{\sigma_j}{\sigma_X^2} [w_{j,k} - M_j - |w_{j,k}| \log\left(\frac{|w_{j,k}|}{K_m \sigma_j}\right)] \quad (12.42)$$

where σ_X is the noise standard deviation in the data. And h_n is defined by:

$$h_n(w_{j,k}) = A(p_n(w_{j,k}))h(w_{j,k}) \quad (12.43)$$

where A is a function which takes the values 0 or 1 depending on $p_n(w_{j,k})$:

$$A(p_n(w_{j,k})) = \begin{cases} 1 & \text{if } p_n(w_{j,k}) > \epsilon \\ 0 & \text{if } p_n(w_{j,k}) \leq \epsilon \end{cases} \quad (12.44)$$

Wavelet coefficients which are significant will impose $A(p_n(w_{j,k}))$ to be equal to 0 (because their probabilities of being due to noise is very small), and do not contribute to H_n . This means that using H_n in a regularization process will have an effect only on scales and positions where no significant wavelet coefficient is detected.

In practice we prefer N1-MSE and N2-MSE for several reasons. First the way the model is used in equation 12.42 is a bit artificial, and there is an undetermination when the wavelet coefficient is equal to 0. Furthermore, LOG-MSE seems difficult to generalize to other classes of noise, which is not the case for N1-MSE and N2-MSE. N2-MSE has the advantage of estimating the corrupted part in the measured information h , even for large wavelet coefficients.

12.2.3 Conclusion

In practice we prefer N1-MSE and N2-MSE for several reasons. First the way the model is used in equation 12.42 is a bit artificial, and there is an undetermination when the wavelet coefficient is equal to 0. Furthermore, LOG-MSE seems difficult to generalize to other classes of noise, which is not the case for N1-MSE and N2-MSE. N2-MSE has the advantage of estimating the corrupted part in the measured information h , even for large wavelet coefficients. Concerning the five points, cited in section 12.1.3:

1. *The information in a flat signal is zero:*

It is true for N1-MSE and N2-MSE. IN the case of LOG-MSE, it is true if we take $\sigma = 0$.

2. *The amount of information in a signal is independent of the background:*

It is always true because the last scale of the wavelet transform is never taken into account in the entropy calculation.

3. *The amount of information is dependent on the noise:*

Whatever the method, it is normalized coefficients which are used, so it is always true.

4. *The entropy must work in the same way for a pixel which has a value $B + \epsilon$, and for a pixel which has a value $B - \epsilon$:*

For N1-MSE and N2-MSE, it is true because the entropy is calculated from the absolute values or the square of the wavelet coefficients. For LOG-MSE, this point is not verified because a $w_{j,k}$ appears. But it has no effect on the solution, because it is the derived of the entropy which is used in the calculations, and this term becomes constant.

5. *The amount of information is dependent on the correlation in the signal. If the signal S presents large features above the noise, it contains a lot of information:*

As the entropy is calculated from the wavelet coefficients, this point is always verified.

Chapter 13

Multiscale Entropy Applied to Filtering

13.1 Introduction

The wavelet transform (WT) has been widely used in recent times and furnishes a new approach for describing and modeling data. Using wavelets, a signal can be decomposed into components of different scales. There are many 2D WT algorithms [191]. The most well-known are perhaps the orthogonal wavelet transform proposed by Mallat [124], and its biorthogonal version [41]. These methods are based on the principle of reducing the redundancy of the information in the transformed data. Other WT algorithms exist, such as the Feauveau algorithm [72] (which is an orthogonal transform using an isotropic wavelet), or the à trous algorithm which is non-orthogonal and furnishes a very redundant dataset [92]. All these methods have advantages and drawbacks. Following the content of the data, and the nature of the noise, each can be considered as optimal.

Once the vision model is chosen, the second fundamental point is to estimate the noise behavior in the transformed data. Linear transforms have in this case the advantage of allowing robust estimation of noise variance. But again, different strategies can be employed, which include soft or hard thresholding [65, 63], and in these latter cases threshold level estimation [181, 135, 65, 108, 38, 3, 145, 146, 146].

We review in the second section the algorithms which can be used for a multiresolution decomposition (we call these vision models in the sequel), and which strategies can be used for treating the noise, once the data have been transformed. Then we introduce in section 3 the Multiscale Entropy Filtering method (MEF), and present a large number of examples. Results of a set of simulations are presented and discussed in section 4 in order to compare the MEF method to other standard wavelet-based methods. Finally, we discuss why a single vision model is often not sufficient to describe the data, and how the combination of several vision models can improve the result. This leads to the concept of a multiple vision model.

13.2 Multiresolution and Filtering

This section reviews different strategies available for wavelet coefficient filtering. A range of important and widely-used transform and filtering approaches are used.

13.2.1 The choice of the multiresolution transform

- The (bi-) orthogonal wavelet transform.

This wavelet transform [124], often referred to as the Fast Wavelet Transform (FWT), is certainly the most widely used among available discrete wavelet transform algorithms.

It is a non-redundant representation of the information. An introduction to this type of transform can be found in [199, 52].

An example is the Haar wavelet transform which consists of using a wavelet defined by

$$\begin{aligned}\psi(x) &= 1 && \text{if } 0 \leq x < \frac{1}{2} \\ \psi(x) &= -1 && \text{if } \frac{1}{2} \leq x < 1 \\ \psi(x) &= 0 && \text{otherwise}\end{aligned}$$

A large class of orthogonal wavelet functions are available.

- The Feauveau wavelet transform.

Feauveau [72] introduced quincunx analysis based on Adelson's work [2]. This analysis is not dyadic and allows an image decomposition with a resolution factor equal to $\sqrt{2}$. By this method, we have only one wavelet image at each scale, and not three as in the previous method.

- The à trous algorithm [92].

The wavelet transform of an image by this algorithm produces, at each scale j , a set $\{w_j\}$. This has the same number of pixels as the image. Furthermore, using a wavelet defined as the difference between the scaling functions of two successive scales ($\frac{1}{2}\psi(\frac{x}{2}) = \phi(x) - \phi(\frac{x}{2})$), the original image c_0 can be expressed as the sum of all the wavelet scales and the smoothed array c_l :

$$c_0 = c_l + \sum_{j=1}^l w_j \quad (13.1)$$

and a pixel at position x, y can be expressed also as the sum of all the wavelet coefficients at this position, plus the smoothed array:

$$c_{0,k} = c_{l,k} + \sum_{j=1}^l w_{j,k} \quad (13.2)$$

- The multiresolution median transform. The median transform is nonlinear, and offers advantages for robust smoothing (i.e. the effects of outlier pixel values are mitigated). The multiresolution median transform [195] (which is not a wavelet transform) consists of a series (c_1, \dots, c_p) of smoothings of the input image, with successively broader kernels. Each resolution scale w_j is constructed from differencing two successive smoothed images ($w_j = c_{j-1} - c_j$). For integer input image values, this transform can be carried out in integer arithmetic only which may lead to computational savings. As in the case of the à trous algorithm, the original image can be expressed as a sum of the scales and the smoothed array.

13.2.2 Filtering in wavelet space

We review in this section some important strategies for treating the noise, once the data have been transformed.

Non-Gaussian noise

If the noise in the data I is Poisson, the transformation [6]

$$t(I) = 2\sqrt{I + \frac{3}{8}} \quad (13.3)$$

acts as if the data arose from a Gaussian white noise model, with $\sigma = 1$, under the assumption that the mean value of I is sufficiently large. The arrival of photons, and their expression by electron counts, on CCD detectors may be modeled by a Poisson distribution. In addition, there is additive Gaussian read-out noise. The Anscombe transformation has been extended to take this combined noise into account. The generalization of the variance stabilizing Anscombe formula is derived as [139]:

$$t(I) = \frac{2}{g} \sqrt{gI + \frac{3}{8}g^2 + \sigma^2 - gm} \quad (13.4)$$

where g is the electronic gain of the detector, σ and m the standard deviation and the mean of the read-out noise.

This implies that for the filtering of an image with Poisson noise or a mixture of Poisson and Gaussian noise, we will first pre-transform the image I into another one $t(I)$ with Gaussian noise. Then $t(I)$ will be filtered, and the filtered image will be inverse-transformed.

For other kinds of noise, modeling must be performed in order to define the noise probability distribution of the wavelet coefficients [191]. In the following, we will consider only stationary Gaussian noise.

Hard thresholding

This consists of setting to 0 all wavelet coefficients which have an absolute value lower than a threshold T_j ($T_j = K\sigma_j$, where j is the scale of the wavelet coefficient, σ_j is the noise standard deviation at the scale j , and K is a constant generally chosen equal to 3). For an energy-normalized wavelet transform algorithm, we have $\sigma_j = \sigma$ for all j .

The appropriate value of σ_j in the succession of wavelet scales is assessed from the standard deviation of the noise σ in the original signal and from study of the noise in the wavelet space. This study consists of simulating a signal containing Gaussian noise with a standard deviation equal to 1, and taking the wavelet transform of this signal. Then we compute the standard deviation σ_j^e at each scale. We get a curve σ_j^e as a function of j , giving the behavior of the noise in the wavelet space. Due to the properties of the wavelet transform, we have $\sigma_j = \sigma\sigma_j^e$ (see [189] for a description of how σ can be automatically calculated directly from the data).

Soft thresholding

Soft thresholding consists of replacing each wavelet coefficient $w_{j,k}$ (j being the scale index, and k the position index) by the value $\tilde{w}_{j,k}$ where

$$\tilde{w}_{j,k} = sgn(w_{j,k})(|w_{j,k}| - T_j) \text{ if } |w_{j,k}| \geq T_j \quad (13.5)$$

$$= 0 \text{ otherwise} \quad (13.6)$$

Donoho universal approach

Donoho [65, 63] has suggested to take $T_j = \sqrt{2\log(n)}\sigma_j$ (where n is the number of pixels) instead of the standard $K\sigma$ value. This leads to a new soft and hard thresholding approach.

Other threshold-based approaches are available. SURE, Stein unbiased risk estimator ([43]) is adaptive in that it is resolution-dependent. The SURE estimator can break down when the wavelet coefficients are mostly around zero. In contrast, the Donoho *universal* hard and soft thresholding approach may overly smooth the data, which is potentially rectified by the minimax criterion proposed in [65]. Note also that Chipman et al. [38] found that SURE create high frequency artifacts.

Multiresolution Wiener filtering

Multiresolution Wiener filtering [181] consists of multiplying all coefficients $w_{j,k}$ of a given scale j by

$$\alpha_j = \frac{S_j}{S_j + N_j} \quad (13.7)$$

where S_j and N_j are respectively the variance of the signal and of the noise at the scale j ($N_j = \sigma_j^2$). In the absence of any information about the signal, we take S_j equal to the difference between the variance of the data w_j and the variance of the noise N_j .

Hierarchical Wiener filtering

Hierarchical Wiener filtering [181] tries to introduce a prediction $w_{j,k}^h$ into the estimation of $\tilde{w}_{j,k}$.

$$\tilde{w}_{j,k} = \frac{H_j}{N_j + H_j + Q_j} w_{j,k} + \frac{N_j}{N_j + H_j + Q_j} w_{j,k}^h \quad (13.8)$$

with:

$$Q_j = \frac{H_j N_j}{S_j} \quad (13.9)$$

where H_j is the variance of the image D obtained by taking the difference of the scale j and the following one $j+1$ ($D = w_j - w_{j+1}$, and $H_j = \frac{1}{N} \sum_k (D_k - m_D)^2$, where N is the number of pixels and m_D the mean of D). If a pyramidal transform is used, the scale w_{j+1} must be first interpolated to the size of the scale of w_j .

This prediction $w_{j,k}^h$ is obtained from the coefficient at the same position but at the following scale. In the case of the à trous algorithm $w_{j,k}^h = w_{j+1,k}$, while for a pyramidal transform, $w_{j,k}^h = w_{j+1,\frac{k}{2}}$.

Hierarchical hard thresholding

The threshold used here, T_h [181], is equal to $T_j = K\sigma_j$ if $|w_{j,k}| \geq T_j$, and $T_h = T_j f(|\frac{w_{j,k}}{\sigma_{j+1}}|)$ otherwise. The function $f(a)$ must return a value between 0 and 1. A possible function for f is:

- $f(a) = 0$ if $a \geq k$
- $f(a) = 1 - \frac{1}{K}a$ if $a < K$

If the predicted wavelet coefficient has a high signal to noise ratio (SNR) (this means that there is certainly some information at this position), the threshold level becomes null, and the wavelet coefficient will not be thresholded, even if its value is small. The threshold level becomes adaptive.

Conclusion

In a soft thresholding, we consider that all coefficients must be corrected because there are all noisy. Hard thresholding principle is that wavelet coefficient with high signal-to-noise ratio should not be corrected because we may lose some significative information. Depending on the quality criterion on the solution, we may prefer one or the other thresholding approach. If the visual aspect quality is the main criterion, the soft thresholding is generally better, because there is less visual artifact. For astronomical images, a soft thresholding should never be used because it leads to a photometry loss of all objects, which can easily be verified by looking to the

residual map (i.e. data - filtered data). Concerning the threshold level, Donoho one corresponds to a minimum risk. Larger is the number of pixels, larger is the risk, and it is normal that the threshold depends on the number of pixels. The $k\sigma$ threshold corresponds to a false detection probability, the probability to detect a coefficient as significant when it is due to the noise. The 3σ value corresponds to 0.27 % of false detection. The Multiresolution Wiener filter is derived from the hypothesis that the signal and the noise follow a Gaussian distribution. This hypothesis is in general not true for the signal. The Hierarchical Wiener filtering introduces the concept of interscale dependence between the wavelet coefficients. Indeed, if we find a wavelet coefficient with a high signal-to-noise ratio at a given scale, we will certainly also find one at the following scale. This interscale dependence is actually used by the best image compression methods [167, 163]. The hierarchical threshold level is varying with this interscale dependence. In the following, these methods will be evaluated using two different images.

13.3 Multiscale Entropy Filtering

13.3.1 Filtering

The problem of filtering or restoring data D can be expressed by the following: We search for a solution \tilde{D} such that the difference between D and \tilde{D} minimizes the information due to the signal, and such that \tilde{D} minimizes the information due to the noise.

$$J(\tilde{D}) = H_s(D - \tilde{D}) + H_n(\tilde{D}) \quad (13.10)$$

Furthermore, the smoothness of the solution can be controlled by adding a parameter:

$$J(\tilde{D}) = H_s(D - \tilde{D}) + \alpha H_n(\tilde{D}) \quad (13.11)$$

In practice [36], we minimize for each wavelet coefficient $w_{j,k}$:

$$j(\tilde{w}_{j,k}) = h_s(w_{j,k} - \tilde{w}_{j,k}) + \alpha h_n(\tilde{w}_{j,k}) \quad (13.12)$$

$j(\tilde{w}_{j,k})$ can be obtained be any minimization routine. In our examples, we have used a simple dichotomy.

Figure 13.1 shows the result when minimizing the functional j with different α values, and a noise standard deviation equal to 1. The corrected wavelet coefficient is plotted versus the wavelet coefficient. From the top curve to the bottom one, α is respectively equal to 0, 0.1, 0.5, 1, 2, 5, 10. The higher the value of α , the more the corrected wavelet coefficient is reduced. When α is equal to 0, there is no regularization and the data are unchanged.

13.3.2 The regularization parameter

The α parameter can be used in different ways:

- It can be fixed to a given value (user parameter): $\alpha = \alpha_u$. This method leads to very fast filtering using the optimization proposed in the following.
- It can be calculated under the constraint that the residual should have some specific characteristic. For instance, in the case of Gaussian noise, we expect a residual with a standard deviation equal to the noise standard deviation. In this case, $\alpha = \alpha_c \alpha_u$. The parameter finally used is taken as the product of a user parameter (defaulted to 1) and the calculated value α_c . This allows the user to keep open the possibility of introducing an under-smoothing, or an over-smoothing. It is clear that such an algorithm is iterative, and will always take more time than a simple hard thresholding approach.
- We can permit more constraints on α by using the fact that we expect a residual with a given standard deviation at each scale j equal to the noise standard deviation σ_j at the same scale. Then rather than a single α we have an α_j per scale.

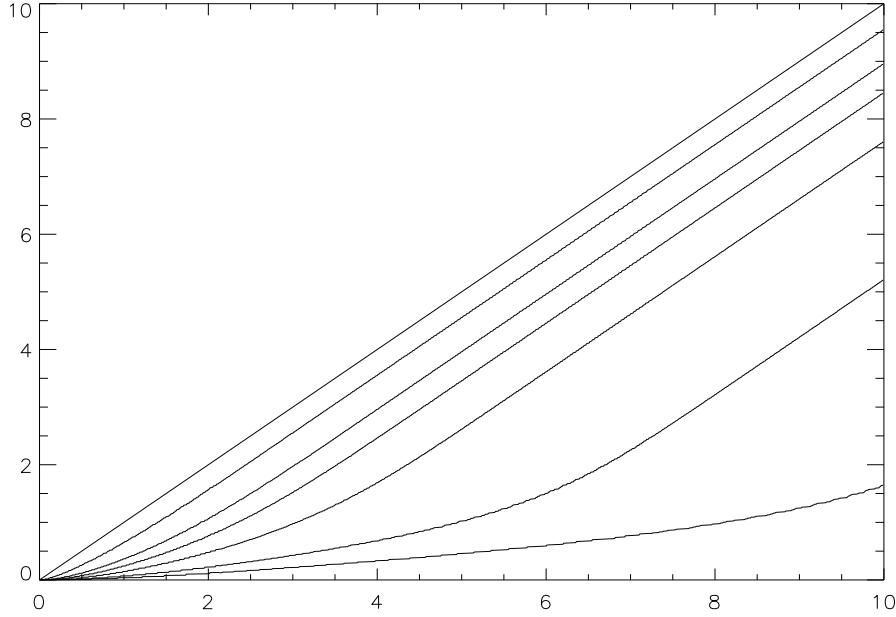


Figure 13.1: Corrected wavelet coefficient versus the wavelet coefficient with different α values (from the top curve to the bottom one, α is respectively equal to 0,0.01,0.5, 1, 2, 5,10).

A more sophisticated way to fix the α value is to introduce a distribution (or a priori knowledge) of how the regularization should work. For instance, in astronomical image restoration, the analyst generally prefers that the flux (total intensity) contained in a star or in a galaxy is not modified by the restoration process. This means that the residual at positions of astronomical objects will approximately be equal to zero. All zero areas in the residual map obviously do not relate to realistic noise behavior, but from the user's point of view they are equally important. For the user, all visible objects in the filtered map contain the same flux as in the raw data. In order to obtain this kind of regularization, the α parameter is no longer a constant value, but depends on the raw data. Hence we have one α per wavelet coefficient, which will be denoted $\alpha_s(w_{j,k})$, and it can be derived by

$$\alpha_s(w_{j,k}) = \alpha_j \frac{1 - L(w_{j,k})}{L(w_{j,k})} \quad (13.13)$$

with $L(w_{j,k}) = \text{MIN}(1, \frac{|w_{j,k}|}{k_s \sigma_j})$, where k_s is a user parameter (typically defaulted to 3).

When $L(w_{j,k})$ is close to 1, $\alpha_s(w_{j,k})$ becomes equal to zero, and there is no regularization anymore, and the obvious solution is $\tilde{w}_{j,k} = w_{j,k}$. Hence, the wavelet coefficient is preserved from any regularization. If $L(w_{j,k})$ is close to 0, $\alpha_s(w_{j,k})$ tends toward infinity, then the first term in equation (13.12) is negligible, and the solution will be $\tilde{w}_{j,k} = 0$. In practice, this means that all coefficients higher than $k_s \sigma_j$ are untouched as in the hard thresholding approach. We also notice that by considering a distribution $L(w_{j,k})$ equal to 0 or 1 (1 when $|w| > K\sigma$ for instance), the solution is then the same as a hard thresholding solution.

13.3.3 The use of a model

Using a model in wavelet space has been successfully applied for denoising (see for example [38, 47, 97]). If we have a model D_m for the data, this can also naturally be inserted into the

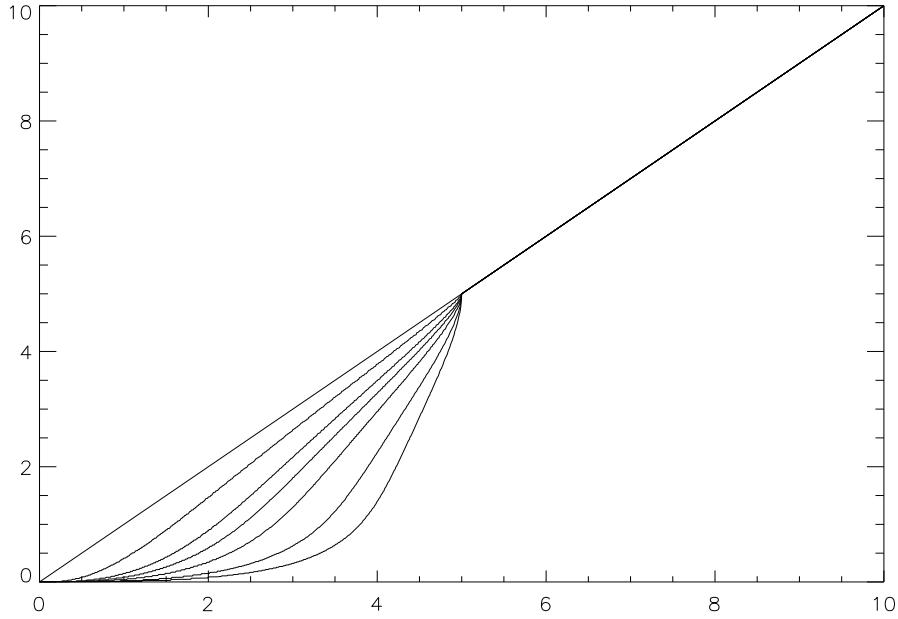


Figure 13.2: Corrected wavelet coefficient versus the wavelet coefficient with different α values.

filtering equation:

$$J_m(\tilde{D}) = H_s(D - \tilde{D}) + \alpha H_n(\tilde{D} - D_m) \quad (13.14)$$

or, for each wavelet coefficient $w_{j,k}$:

$$j_m(\tilde{w}_{j,k}) = h_s(w_{j,k} - \tilde{w}_{j,k}) + \alpha h_n(\tilde{w}_{j,k} - w_{j,k}^m) \quad (13.15)$$

where $w_{j,k}^m$ is the corresponding wavelet coefficient of D_m .

The model can be of quite different types. It can be an image, and in this case, the coefficients $w_{j,k}^m$ are obtained by a simple wavelet transform of the model image. It can also be expressed by a distribution or a given function which furnishes a model wavelet coefficient w^m from the data. For instance, the case where we want to keep intact high wavelet coefficients (see equation 13.13) can also be treated by the use of a model, just by calculating $w_{j,k}^m$ by

$$w_{j,k}^m = P_s(w_{j,k})w_{j,k} \quad (13.16)$$

when $w_{j,k}$ has a high signal to noise ratio, $P_s(w_{j,k})$ is close to 1, and $w_{j,k}^m$ is equal to $w_{j,k}$. Then $\alpha h_n(\tilde{w}_{j,k} - w_{j,k}^m)$ is equal to zero and $\tilde{w}_{j,k} = w_{j,k}$, i.e. no regularization is carried out on $w_{j,k}$.

Other models may also be considered. When the image contains contours, it may be interesting to derive the model from the detected edge. Zero-crossing wavelet coefficients indicate where the edges are [125]. By averaging three wavelet coefficients in the direction of the detected edge, we get a value w_a , from which we derive the SNR S_e of the edge ($S_e = 0$ if there is no detected edge). The model value w^m is set to w_a if a contour is detected, and 0 otherwise. This approach has the advantage to filter the wavelet coefficient, and even if an edge is clearly detected the smoothing operates in the direction of the edge.

There is naturally no restriction on the model. When we have a priori information of the content of an image, we should use it in order to improve the quality of the filtering. It is clear that the way we use the knowledge of the presence of edges in an image is not a closed question. The model in the entropy function is an interesting direction to investigate in the future.

13.3.4 The multiscale entropy filtering algorithm

The Multiscale Entropy Filtering algorithm (MEF) consists of minimizing for each wavelet coefficient $w_{j,k}$ at scale j

$$j_m(\tilde{w}_{j,k}) = h_s(w_{j,k} - \tilde{w}_{j,k}) + \alpha_j h_n(\tilde{w}_{j,k} - w_{j,k}^m) \quad (13.17)$$

or

$$j_{ms}(\tilde{w}_{j,k}) = h_s(w_{j,k} - \tilde{w}_{j,k}) + \alpha_j \alpha_s(w_{j,k}) h_n(\tilde{w}_{j,k} - w_{j,k}^m) \quad (13.18)$$

if the SNR is used. By default the model $w_{j,k}^m$ is set to 0. There is no user parameter because the α_j are calculated automatically in order to verify the noise properties. If an over-smoothing (or an under-smoothing) is desired, a user parameter must be introduced. We propose in this case to calculate the α_j in the standard way, and then to multiply the calculated values by a user value α_u defaulted to 1. Increasing α_u will lead to an over-smoothing, while decreasing α_u implies an under-smoothing.

Using a simple dichotomy, the algorithm becomes:

1. Estimate the noise in the data σ (see [150, 189]).
2. Wavelet transform of the data.
3. Calculate from σ the noise standard deviation σ_j at each scale j .
4. Set $\alpha_j^{min} = 0$, $\alpha_j^{max} = 200$.
5. For each scale j do
 - (a) Set $\alpha_j = \frac{\alpha_j^{min} + \alpha_j^{max}}{2}$
 - (b) For each wavelet coefficient $w_{j,k}$ of scale j , find $\tilde{w}_{j,k}$ by minimizing $j_m(\tilde{w}_{j,k})$ or $j_{ms}(\tilde{w}_{j,k})$
 - (c) Calculate the standard deviation of the residual:

$$\sigma_j^r = \sqrt{\frac{1}{N_j} \sum_{k=1}^{N_j} (w_{j,k} - \tilde{w}_{j,k})^2}$$
 - (d) If $\sigma_j^r > \sigma_j$ then the regularization is too strong, and we set α_j^{max} to α_j , otherwise we set α_j^{min} to α_j (σ_j is derived from the method described in section 13.2.2).
6. If $\alpha_j^{max} - \alpha_j^{min} > \epsilon$ then go to 5.
7. Multiply all α_j by the constant α_u .
8. For each scale j and for each wavelet coefficient w find $\tilde{w}_{j,k}$ by minimizing $j_m(\tilde{w}_{j,k})$ or $j_{ms}(\tilde{w}_{j,k})$.
9. Reconstruct the filtered image from $\tilde{w}_{j,k}$ by the inverse wavelet transform.

The minimization of j_m or j_{ms} (step 5b) can be done by any method. For instance, a simple dichotomy can be used in order to find \tilde{w} such that

$$\nabla(j_m(\tilde{w}_{j,k})) = \frac{\partial h_s(w_{j,k} - \tilde{w}_{j,k})}{\partial \tilde{w}_{j,k}} + \alpha_j \frac{\partial h_n(\tilde{w}_{j,k})}{\partial \tilde{w}_{j,k}} = 0 \quad (13.19)$$

In the case of Gaussian noise, and by using N2-MSE approach, we derive from equations 13.41 and 13.46 described in the next section that $\nabla(j_m(\tilde{w}_{j,k}))$ is defined by:

$$\begin{aligned} \nabla(j(\tilde{w}_{j,k})) &= -\frac{w_{j,k} - \tilde{w}_{j,k}}{\sigma_j^2} \operatorname{erf}\left(\frac{w_{j,k} - \tilde{w}_{j,k}}{\sqrt{2}\sigma_j}\right) + \sqrt{\frac{2}{\pi}} \frac{1}{\sigma_j} \left[1 - e^{-\frac{(w_{j,k} - \tilde{w}_{j,k})^2}{2\sigma_j^2}} \right] + \\ &\quad \alpha_j \left(\frac{\tilde{w}_{j,k}}{\sigma_j^2} \operatorname{erfc}\left(\frac{\tilde{w}_{j,k}}{\sqrt{2}\sigma_j}\right) + \frac{1}{\sigma_j} \sqrt{\frac{2}{\pi}} \left[1 - e^{-\frac{\tilde{w}_{j,k}^2}{2\sigma_j^2}} \right] \right) \end{aligned} \quad (13.20)$$

The idea to treat the wavelet coefficients such that the residual respects some constraint has also been used in [145, 146, 4] using cross-validation. However, cross validation appears to overfit the data [199].

13.3.5 Derivative calculation of h_s and h_n for N2-MSE

The erf function

The erf and erc functions are defined by

$$\begin{aligned} \operatorname{erf}(x) &= \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \\ \operatorname{erfc}(x) &= 1 - \operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt \end{aligned} \quad (13.21)$$

These functions have the following symmetries

$\operatorname{erf}(0) = 0$	$\operatorname{erf}(\infty) = 1$
$\operatorname{erfc}(0) = 1$	$\operatorname{erfc}(\infty) = 0$
$\operatorname{erf}(-x) = \operatorname{erf}(x)$	$\operatorname{erfc}(-x) = 2 - \operatorname{erfc}(x)$

(13.22)

Gaussian case

We compute the contribution of the wavelet coefficient x to the noise information:

$$h_n(x) = \frac{1}{\sigma^2} \int_0^x t \operatorname{erfc}\left(\frac{x-t}{\sqrt{2}\sigma}\right) dt \quad (13.23)$$

$$\begin{aligned} \frac{dh_n(x)}{dx} &= h_n(x+dx) - h_n(x) \\ &= \frac{1}{\sigma^2} \int_0^{x+dx} t \operatorname{erfc}\left(\frac{x+dx-t}{\sqrt{2}\sigma}\right) dt - \frac{1}{\sigma^2} \int_0^x t \operatorname{erfc}\left(\frac{x-t}{\sqrt{2}\sigma}\right) dt \\ &= \frac{1}{\sigma^2} \int_0^x \left[t \operatorname{erfc}\left(\frac{x+dx-t}{\sqrt{2}\sigma}\right) - t \operatorname{erfc}\left(\frac{x-t}{\sqrt{2}\sigma}\right) \right] dt + \frac{1}{\sigma^2} \int_x^{x+dx} t \operatorname{erfc}\left(\frac{x+dx-t}{\sqrt{2}\sigma}\right) dt \\ &= \frac{1}{\sigma^2} \int_0^x t \frac{\partial \operatorname{erfc}\left(\frac{x-t}{\sqrt{2}\sigma}\right)}{\partial x} + \frac{1}{\sigma^2} \int_x^{x+dx} t \operatorname{erfc}\left(\frac{x+dx-t}{\sqrt{2}\sigma}\right) dt \\ &= \frac{1}{\sigma^2} \int_0^x t \frac{\partial \operatorname{erfc}\left(\frac{x-t}{\sqrt{2}\sigma}\right)}{\partial x} dt + \frac{x}{\sigma^2} \operatorname{erfc}(0) \end{aligned} \quad (13.24)$$

Now, because $\text{erfc}(0) = 1$ we have:

$$\frac{dh_n(x)}{dx} = \frac{x}{\sigma^2} + \frac{1}{\sigma^2} \int_0^x t \frac{\partial \text{erfc}(\frac{x-t}{\sqrt{2}\sigma})}{\partial x} dt \quad (13.25)$$

We derive the function erfc :

$$\frac{\partial \text{erfc}(x)}{\partial x} = -\frac{2}{\sqrt{\pi}} e^{-x^2} \quad (13.26)$$

$$\frac{\partial \text{erfc}(\frac{(x-t)}{\sqrt{2}\sigma})}{\partial x} = -\frac{2}{\sqrt{\pi}} \frac{1}{\sqrt{2}\sigma} e^{-\frac{(x-t)^2}{2\sigma^2}} = -\sqrt{\frac{2}{\pi}} \frac{1}{\sigma} e^{-\frac{(x-t)^2}{2\sigma^2}} \quad (13.27)$$

Now we deduce for the derivative of h_n :

$$\frac{dh_n(x)}{dx} = \frac{x}{\sigma^2} + \frac{1}{\sigma^2} \int_0^x -\sqrt{\frac{2}{\pi}} \frac{1}{\sigma} t e^{-\frac{(x-t)^2}{2\sigma^2}} dt \quad (13.28)$$

$$\frac{dh_n(x)}{dx} = \frac{x}{\sigma^2} + \frac{1}{\sigma^3} \sqrt{\frac{2}{\pi}} \int_0^x t e^{-\frac{(x-t)^2}{2\sigma^2}} dt \quad (13.29)$$

We create the variable J

$$J = \int_0^x t e^{-\frac{(x-t)^2}{2\sigma^2}} dt \quad (13.30)$$

We create the variable u

$$\begin{aligned} u &= \frac{t-x}{\sqrt{2}\sigma} & t &= x + u \sqrt{2}\sigma \\ && dt &= \sqrt{2}\sigma du \\ t = 0 \Rightarrow u &= \frac{-x}{\sqrt{2}\sigma} & t = x \Rightarrow u &= 0 \end{aligned} \quad (13.31)$$

The variable J can be written with u

$$J = \int_{\frac{-x}{\sqrt{2}\sigma}}^0 (x + u \sqrt{2}\sigma) e^{-u^2} \sqrt{2}\sigma du \quad (13.32)$$

$$J = \sqrt{2}\sigma x \int_{\frac{-x}{\sqrt{2}\sigma}}^0 e^{-u^2} du + 2\sigma^2 \int_{\frac{-x}{\sqrt{2}\sigma}}^0 u e^{-u^2} du \quad (13.33)$$

The first part of J can be rewritten as:

$$J_0 = \sqrt{2}\sigma x \int_0^{\frac{x}{\sqrt{2}\sigma}} e^{-u^2} du \quad (13.34)$$

J_0 can be expressed with the error function.

$$J_0 = \sqrt{2}\sigma \frac{\sqrt{\pi}}{2} x \text{erf} \left(\frac{x}{\sqrt{2}\sigma} \right) = \sigma \sqrt{\frac{\pi}{2}} x \text{erf} \left(\frac{x}{\sqrt{2}\sigma} \right) \quad (13.35)$$

Now the second part of J is obvious

$$J_1 = 2\sigma^2 \int_{\frac{-x}{\sqrt{2}\sigma}}^0 u e^{-u^2} du \quad (13.36)$$

or

$$\frac{de^{-u^2}}{du} = -2ue^{-u^2} \quad (13.37)$$

We replace

$$J_1 = -\sigma^2 \int_{\frac{-x}{\sqrt{2}\sigma}}^0 d(e^{-u^2}) \quad (13.38)$$

$$J_1 = \sigma^2 [e^{-\frac{x^2}{2\sigma^2}} - 1] \quad (13.39)$$

Now we can write J

$$J = J_0 + J_1 = \sigma \sqrt{\frac{\pi}{2}} x \operatorname{erf}\left(\frac{x}{\sqrt{2}\sigma}\right) + \sigma^2 \left[e^{-\frac{x^2}{2\sigma^2}} - 1 \right] \quad (13.40)$$

We can write the derivative of h_n

$$\begin{aligned} \frac{dh_n(x)}{dx} &= \frac{x}{\sigma^2} - \frac{1}{\sigma^3} \sqrt{\frac{2}{\pi}} J \\ &= \frac{x}{\sigma^2} - \frac{x}{\sigma^2} x \operatorname{erf}\left(\frac{x}{\sqrt{2}\sigma}\right) + \frac{1}{\sigma} \sqrt{\frac{2}{\pi}} \left[1 - e^{-\frac{x^2}{2\sigma^2}} \right] \\ &= \frac{x}{\sigma^2} \operatorname{erfc}\left(\frac{x}{\sqrt{2}\sigma}\right) + \frac{1}{\sigma} \sqrt{\frac{2}{\pi}} \left[1 - e^{-\frac{x^2}{2\sigma^2}} \right] \end{aligned} \quad (13.41)$$

In order to minimize the functional (13.10), we may want to calculate the derivative of $h_s(y - x)$, where $h_s(y - x)$ measures the amount of information contained in the residual (y being the data).

$$h_s(y - x) = \frac{1}{\sigma^2} \int_0^{y-x} t \operatorname{erf}\left(\frac{y-x-t}{\sqrt{2}\sigma}\right) dt \quad (13.42)$$

Denoting $z = y - x$, we have

$$\begin{aligned} h_s(z) &= \frac{1}{\sigma^2} \int_0^z t \operatorname{erf}\left(\frac{z-t}{\sqrt{2}\sigma}\right) dt \\ &= \frac{1}{\sigma^2} \int_0^z t dt - \frac{1}{\sigma^2} \int_0^z t \operatorname{erfc}\left(\frac{z-t}{\sqrt{2}\sigma}\right) dt \end{aligned} \quad (13.43)$$

and

$$\frac{dh_s(x)}{dx} = \frac{dh_s(z)}{dz} \frac{dz}{dx} \quad (13.44)$$

$$\frac{dh_s(z)}{dz} = \frac{z}{\sigma^2} - \frac{dh_n(z)}{dz} \quad (13.45)$$

then

$$\begin{aligned} \frac{dh_s(y-x)}{dx} &= -\frac{y-x}{\sigma^2} + \frac{y-x}{\sigma^2} \operatorname{erfc}\left(\frac{y-x}{\sqrt{2}\sigma}\right) + \sqrt{\frac{2}{\pi}} \frac{1}{\sigma} \left[1 - e^{-\frac{(y-x)^2}{2\sigma^2}} \right] \\ &= -\frac{y-x}{\sigma^2} \operatorname{erf}\left(\frac{y-x}{\sqrt{2}\sigma}\right) + \sqrt{\frac{2}{\pi}} \frac{1}{\sigma} \left[1 - e^{-\frac{(y-x)^2}{2\sigma^2}} \right] \end{aligned} \quad (13.46)$$

13.3.6 General Case

The contribution of the wavelet coefficient x to the noise and signal information in the general case is

$$\begin{aligned} h_n(x) &= \int_0^{|x|} P_n(x-u) \left(\frac{\partial h(x)}{\partial x} \right)_{x=u} du \\ h_s(x) &= \int_0^{|x|} P_s(x-u) \left(\frac{\partial h(x)}{\partial x} \right)_{x=u} du \end{aligned} \quad (13.47)$$

Assuming $h(x) = \frac{1}{2}x^2$, we have

$$\begin{aligned} h_n(x) &= \int_0^{|x|} P_n(x-u) u du \\ h_s(x) &= \int_0^{|x|} P_s(x-u) u du \end{aligned} \quad (13.48)$$

$$\frac{dh_s(x)}{dx} = \int_0^x \left(\frac{\partial P_s(x-u)}{\partial x} \right)_{x=u} u du + \frac{1}{dx} \int_x^{x+dx} P_s(x-u) u du \quad (13.49)$$

Since $P_s(0) = 0$, the second term tends to zero.

Denoting $\frac{\partial P_s(x-u)}{\partial x} = -\frac{\partial P_s(x-u)}{\partial u}$, we have

$$\begin{aligned} \frac{dh_s(x)}{dx} &= - \int_0^x \frac{\partial P_s(x-u)}{\partial u} u du \\ &= -([uP_s(x-u)]_0^x - \int_0^x P_s(x-u) du) \\ &= \int_0^x P_s(x-u) du \\ &= \int_0^x P_s(u) du \end{aligned} \quad (13.50)$$

and from $h_n = h - h_s$ we get

$$\frac{dh_n(x)}{dx} = x - \int_0^x P_s(u) du \quad (13.51)$$

and

$$\frac{dh_s(y-x)}{dx} = - \int_0^{y-x} P_s(u) du \quad (13.52)$$

It is easy to verify that replacing $P_s(x) = \text{erf}(x)$, and $P_n(x) = \text{erfc}(x)$ (case of Gaussian noise) we find the same equation as in the Gaussian case.

13.3.7 Optimization

In the case of Gaussian noise, the calculation of erf and erfc functions could lead to a considerable time computation, when compared to a simple filtering method. This can be easily avoided by precomputing tables, which is possible due to the specific properties of $\frac{\partial h_s}{\partial \bar{w}}$ and $\frac{\partial h_n}{\partial \bar{w}}$. h_s and h_n

are functions of the standard deviation of the noise, and we denote the reduced functions by h_s^r and h_n^r , i.e. h_s and h_n for noise standard deviation equal to 1. It is easy to verify that

$$\frac{\partial h_s(w_{j,k})}{\partial \tilde{w}} = \sigma_j \frac{\partial h_s^r(\frac{w_{j,k}}{\sigma_j})}{\partial \tilde{w}} \quad (13.53)$$

$$\frac{\partial h_n(w_{j,k})}{\partial \tilde{w}} = \sigma_j \frac{\partial h_n^r(\frac{w_{j,k}}{\sigma_j})}{\partial \tilde{w}} \quad (13.54)$$

Furthermore, $\frac{\partial h_n^r}{\partial \tilde{w}}$ and $\frac{\partial h_s^r}{\partial \tilde{w}}$ are symmetric functions, $\frac{\partial h_n^r}{\partial \tilde{w}}$ converges to a constant value C ($C=0.798$), and $\frac{\partial h_s^r}{\partial \tilde{w}}$ tends to $C - w$ when w is large enough (> 5). In our implementation, we precomputed the tables using a step-size of 0.01 from 0 to 5. If no model is introduced and if the SNR is not used, the filtered wavelet of coefficients is a function of α and $\frac{w_j}{\sigma_j}$, and a second level of optimization can be performed by precomputed tables of solutions for different values of α .

13.4 Examples

13.4.1 1D data filtering

Figures 13.3, 13.4 and 13.5 show the results of the multiscale entropy method on simulated data (2048 pixels). From top to bottom, each figure shows simulated data, the noisy data, the filtered data, and both noisy and filtered data overplotted. For the two first filterings, all default parameters were taken (noise standard deviation and α_j automatically calculated, $\alpha_u = 1$, and the chosen wavelet transform algorithm is the à trous one). For the block signal (Fig. 13.3), default parameters were also used, but the multiresolution transform we used is the multiresolution median transform.

Figure 13.6 shows the result after applying the MEF method to a real spectrum (512 pixels). The last plot shows the difference between the original and the filtered spectrum. As we can see, the residual contains only noise. In this case, we used also default parameters, but we introduce the SNR in the calculation of α .

13.4.2 Image filtering

A simulated 256×256 image containing stars and galaxies is shown in Fig. 13.7 (top left). The simulated noisy image, the filtered image and the residual image are respectively shown in Fig. 13.7 top right, bottom left, and bottom right. We can see that there is no structure in the residual image.

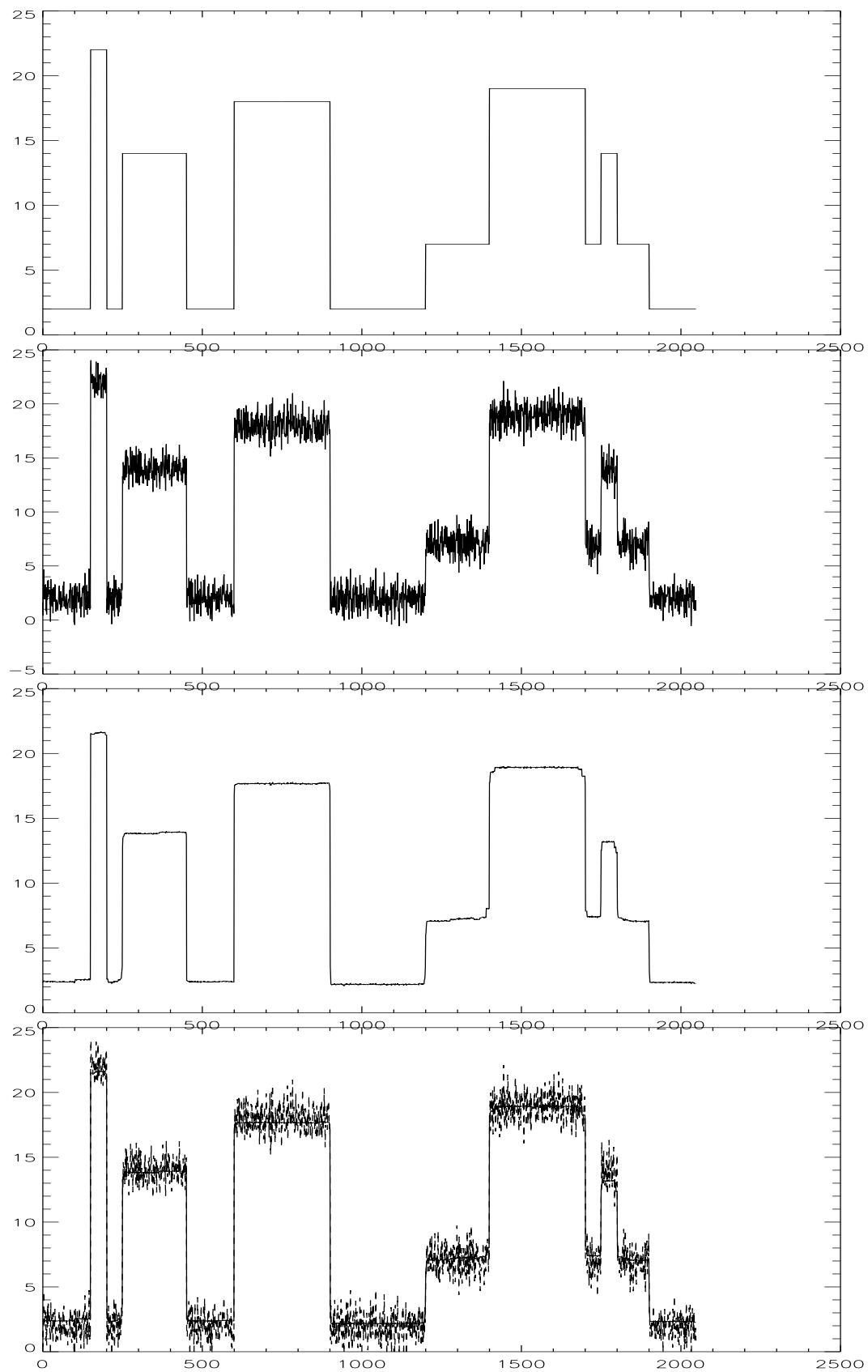


Figure 13.3: From top to bottom, simulated block data, noise blocks, filtered blocks, and both noisy and filtered blocks overplotted.

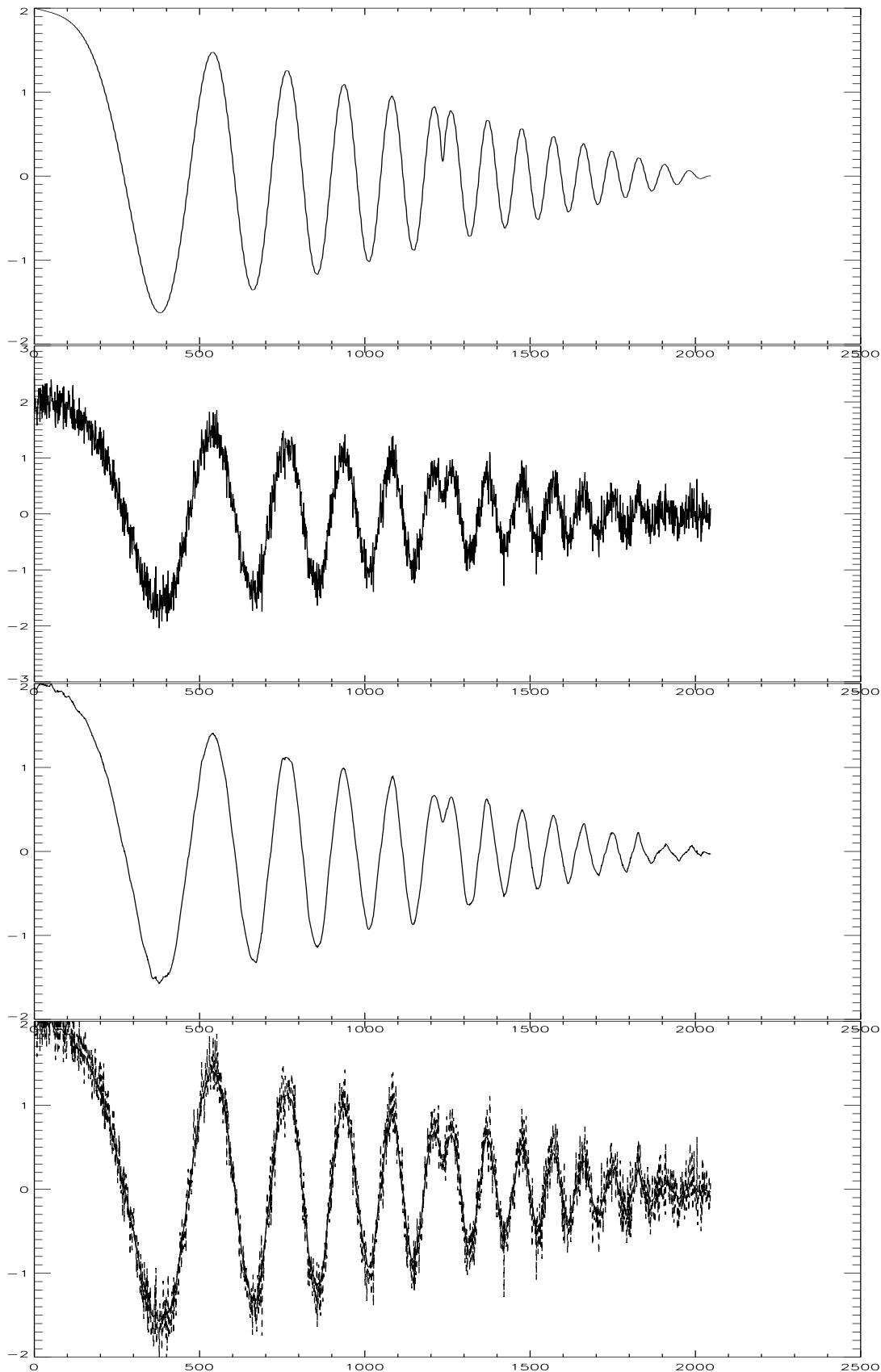


Figure 13.4: From top to bottom, simulated data, noisy data, filtered data, and both noisy and filtered data overplotted.

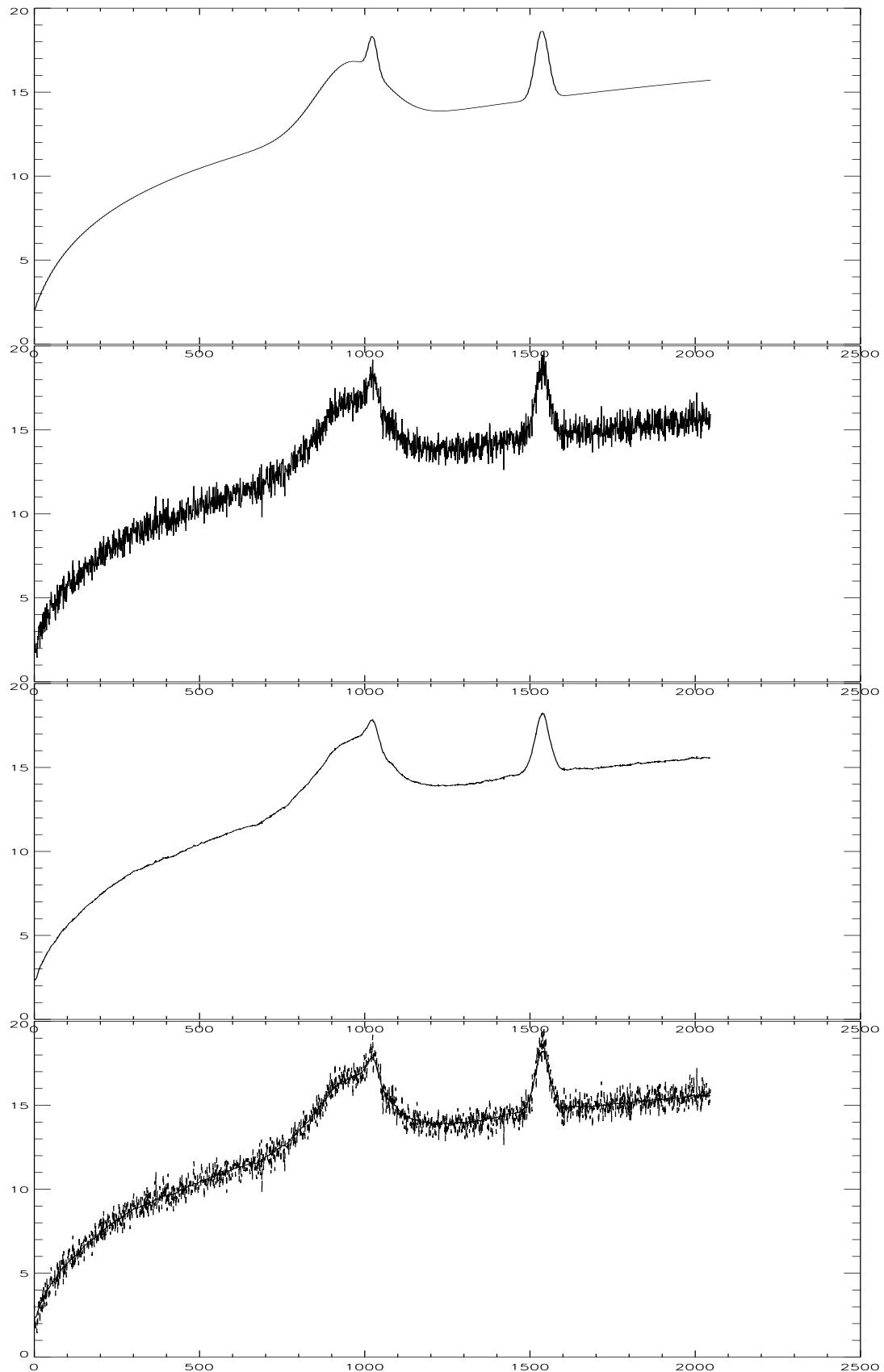


Figure 13.5: From top to bottom, simulated data, noisy data, filtered data, and both noisy and filtered data overplotted.

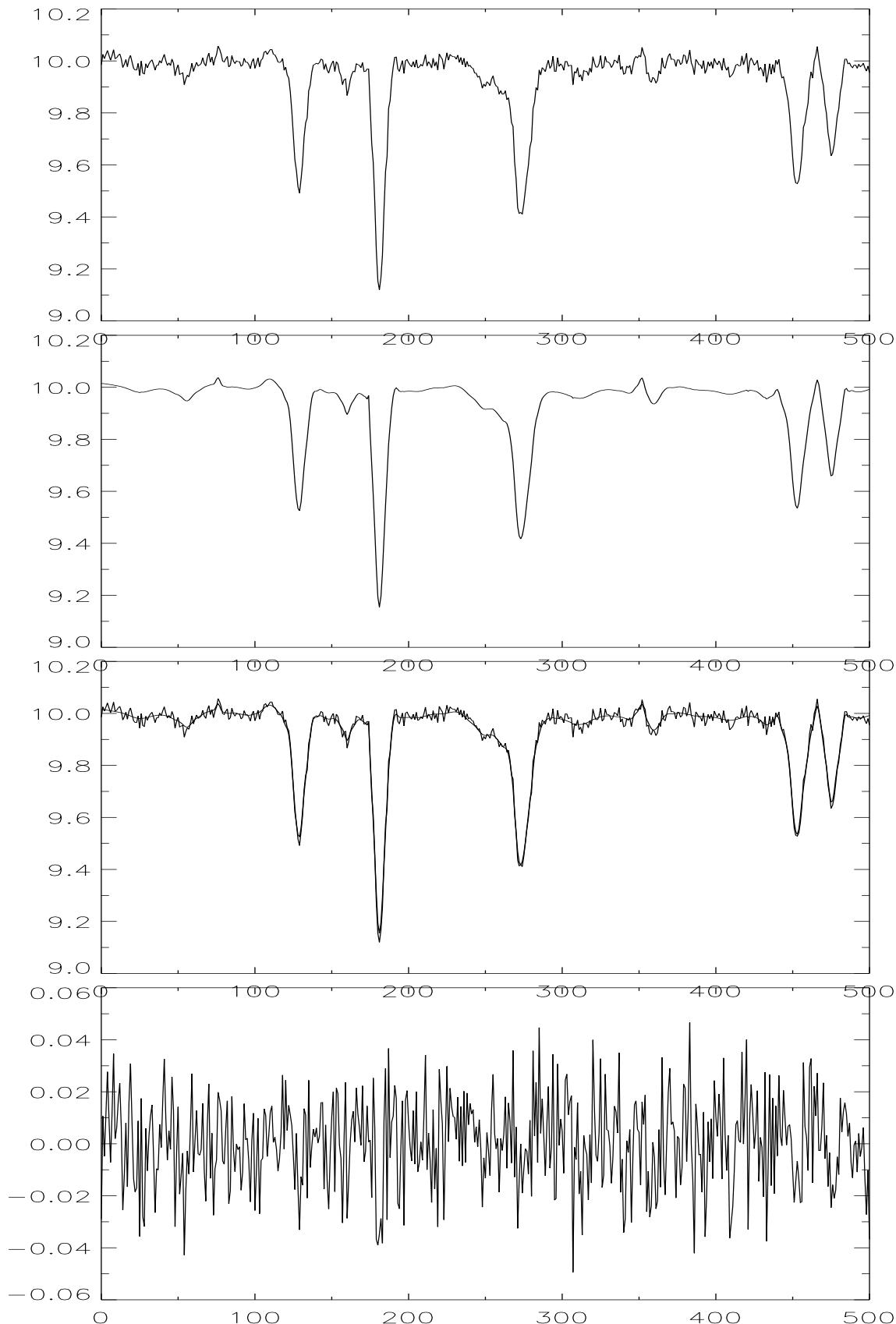


Figure 13.6: From top to bottom, real spectrum, filtered spectrum, both noisy and filtered spectrum overplotted, and difference between the spectrum and the filtered data. As we can see, the residual contains only noise.

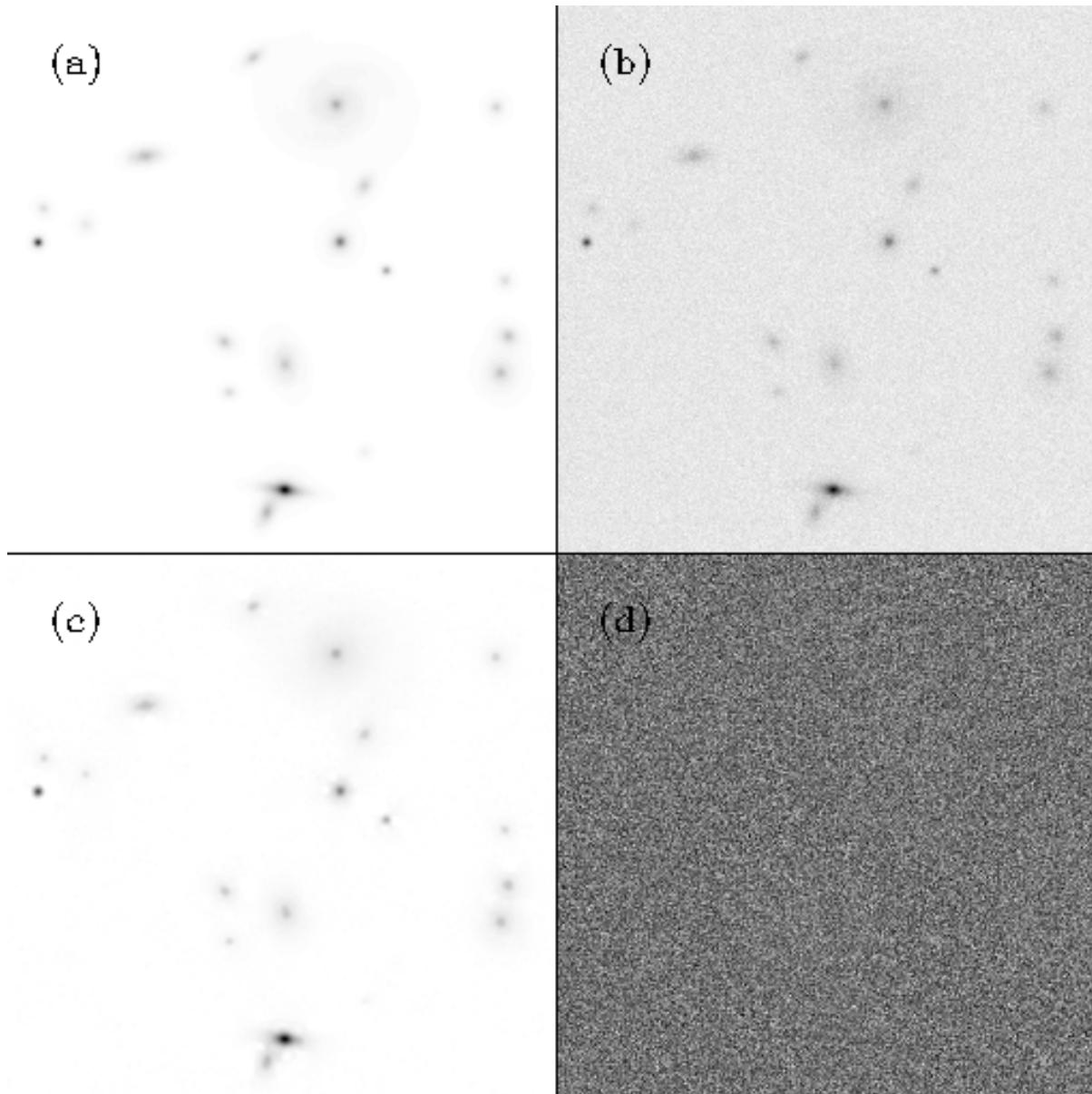


Figure 13.7: (a) Simulated image, (b) simulated image and Gaussian noise, (c) filtered image, and (d) residual image.

13.5 Comparison with Other Methods from Simulations

13.5.1 Simulation descriptions

A set of simulations were carried out based on two images: the classical Lena 512×512 image, and a 512×512 landscape image. From each image, three images were created by adding Gaussian noise with standard deviations of 5, 10, 30. These six images were filtered using different multiresolution methods and different noise treatment methods. The multiresolution methods were:

1. Haar wavelet transform (FWT-Haar).
2. Mallat-Daubechies biorthogonal wavelet transforms using the Daubechies-Antonini 7/9 filters [7] (FWT-7/9).
3. Feauveau wavelet transform.
4. À trous algorithm using a B-spline scaling function (see [184, 191] for more details).
5. Multiresolution median transform (MMT) [195].

The first two belong to the class of fast wavelet transforms. The third is also a non-redundant transform, but compared to the FWT, the wavelet function is isotropic. The à trous algorithm is redundant and symmetric, and finally the MMT is not a wavelet transform, but does allow a multiresolution representation.

Using these five transforms, we used eight different strategies for correcting the multiresolution coefficients from the noise:

- k-sigma hard and soft thresholding
- Donoho hard and soft thresholding
- Multiscale entropy method
- Hierarchical hard thresholding
- Multiresolution Wiener filtering
- Hierarchical Wiener filtering

The last three strategies have up to now only been used with redundant transforms (à trous algorithm and MMT in our case).

Finally, close to two hundred filtered images were created. Four resolution scales were used for the filtering, and the constant k for the hard thresholding was always taken as equal to 4 for the first scale, and 3 for the others. For the multiscale entropy method, the parameter α was determined by the program in order to get a standard deviation of the residual (i.e. image minus filtered image) of the same order as the noise standard deviation.

For each filtered image, the PSNR (peak signal-to-noise) ratio between the original image I and the filtered image F was calculated as:

$$PSNR_{dB} = 10 \log_{10} \frac{255}{NRMSE^2} \quad (13.55)$$

where NRMSE is the normalized root mean square error:

$$NRMSE^2 = \frac{\sum_{pix} (I - F)^2}{\sum_{pix} I^2} \quad (13.56)$$

We also calculated the correlation factor, but we found that this does not furnish more information than the PSNR. If the PSNR is an objective measure, it is however not sufficient, because it does not allow us to control whether artifacts are present or not. Images were therefore also visually assessed, in order to decide if artifacts are visible.

Results of the simulations are presented in Tables 13.5.1, 13.5.1.

13.6 Simulation Analysis

Multiresolution algorithm

Filtering using the Haar transform always produces artifacts, even at low noise levels. When using other filters, artifacts appear only beyond a given noise level. Improving the filter set improves the filtered image quality, which is a well-known result. When the noise increases, artifacts appear, even with a good filter set such as the Antonini 7/9 one.

- **Feauveau WT.** The standard orthogonal WT is always better than the Feauveau method for filtering.
- **À trous algorithm.** This does not create artifacts when thresholding, and results are significantly better (from the visual point of view) at high noise levels, compared to orthogonal WT approaches. As opposed to the standard WT method, this transform is symmetric and performs better on isotropic structures compared to faint contours. This is the reason for its success on astronomical images where objects are diffuse and more or less isotropic (stars, galaxies, etc.).
- **Multiresolution median transform.** This transform is non-linear, and noise estimation at the different scales cannot be carried out in the same rigorous way as with linear transforms. For pure Gaussian noise, there is clearly no interest in using this transform, even if it respects well the morphology of the objects contained in the image. For some other kinds of noise, the non-linearity can be an advantage, and it can then be considered.

Conclusion

The Feauveau WT and the MMT are not competitive for filtering in the case of Gaussian noise. FWT-7/9 allows better restoration of the edges than the à trous algorithm, but the à trous algorithm is more robust from the visual point of view. The important point to be made is clearly that the way the information is represented is fundamental. At high noise levels, whatever the chosen filter set, we will always have more artifacts using the FWT than with the à trous algorithm.

Noise treatment strategies

- **The optimal method depends on the noise level.**
At low noise levels, simple thresholding using an orthogonal wavelet transform leads to very good results. When the noise increases, artifacts appear. Non-orthogonal transforms produce better results, and soft thresholding strategies lead to more acceptable image quality.
- **Donoho soft and hard thresholding versus the k-sigma approach.**
Whatever the multiresolution transform and the noise level, the k-sigma hard (respectively soft) thresholding is always better than the Donoho hard (respectively soft) thresholding. Both PSNR ratio and visual aspect are better using the k-sigma approach. This outcome is not too surprising. Indeed the threshold, in the Donoho approach, is increasing with the number of pixels (justified in order to have a fixed number of “artifacts”). For our 512×512 image, this approach is equivalent to thresholding at 5σ . But then the thresholding level is too high, because the main coefficients between 3σ and 5σ are significant. The larger the image size, the stronger will be the over-smoothing.
- **Hierarchical thresholding.**
The modification of the thresholding level at a given scale using the information at the following scale improves the result. The PSNR is better, and the visual aspect is similar to the hard thresholding. This procedure could certainly be also introduced into orthogonal transforms.

- **Quality of the multiscale entropy method.**

The multiscale entropy method proposes a visually good solution whatever the noise level. It is in fact a method which preserves high wavelet coefficients, and corrects other wavelet coefficients in an adaptive, soft, manner.

13.7 Conclusion: Toward Combined Filtering

If a hard or a soft thresholding approach is used, the k-sigma value should be preferred to the universal $\sqrt{2 \log(n)}$ value. Multiresolution Wiener filtering and hierarchical Wiener filtering are not at all competitive.

The multiscale entropy method is an adaptive soft approach which is certainly the best when considering both visual quality and the PSNR criterion. At low noise levels, a FWT can be used, which allows better restoration of edges (assuming the image does contain edges!), and at high noise levels, the à trous algorithm must be chosen since otherwise artifacts related to decimation appear. However, these artifacts are less severe than those produced by poor thresholding.

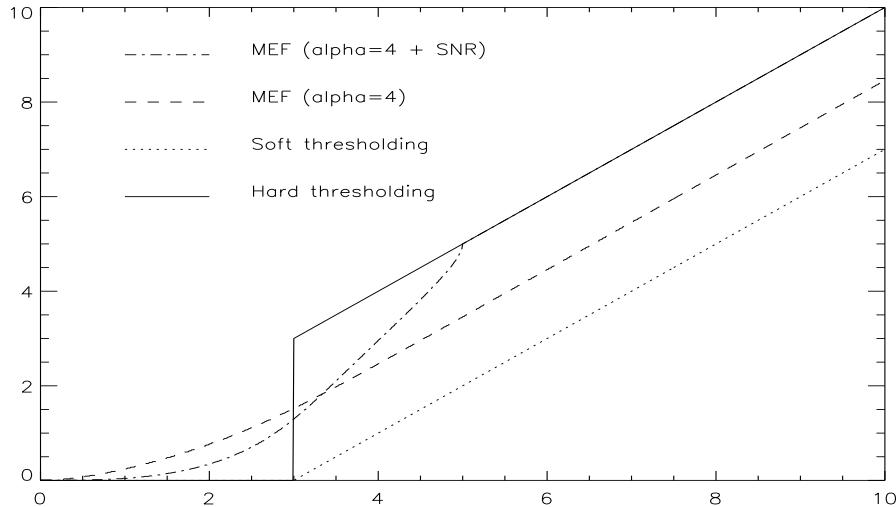


Figure 13.8: Filtered wavelet coefficients versus wavelet coefficients (for a noise standard deviation equal to 1) by four methods: hard thresholding, soft thresholding, multiscale entropy filtering, and multiscale entropy filtering with a non-constant α (SNR-dependent) value.

Figure 13.8 shows how a wavelet coefficient is modified using a hard thresholding, a soft thresholding, MEF method, and MEF method with α as a function of the SNR. As we can see, MEF methods are intermediate between hard and soft thresholding, but do not present any discontinuity as the hard thresholding. This is the reason why good SNR is obtained with the MEF method, while retaining also good visual quality.

As the previous section demonstrates, it is not easy to find an optimal method for all noise amplitudes. At low noise levels, hard thresholding methods produce perfect results, and when the noise increases, the situation changes. Orthogonal transforms produce artifacts related to the decimation, hard thresholding produces visual artifacts which are less severe with soft methods, such as soft thresholding and multiscale entropy. The isotropic wavelet transform allows better extraction of isotropic structures, whereas the biorthogonal transform describes the contours in a better way. The MMT permits also a relatively good object shape description. The conclusion is that there is no perfect method. Each method has its advantages and its drawbacks, and can be in some situations better than others. Parameters for choosing one method rather than another include the noise level, the nature of the noise, and the shape of the significant information.

Having a default optimal method seems impossible when using any of the strategies described above. This can also be interpreted in another way: the complexity of an image is so high, that all

vision models are too simplistic to fully describe the information, and subsequently to separate the signal from the noise.

We have therefore developed the idea to combine the results obtained from different filtering strategies. Seven filtered images have been simply averaged. These seven images were obtained from the soft thresholding, the hard thresholding, and the multiscale entropy method using both the à trous algorithm and the biororthogonal transform, and the hierarchical thresholding with the à trous algorithm. PSNR is represented in Table 13.7. Comparing Table 13.7 with previous results, we see that PSNR of the combined method is clearly always above that of all other methods. The visual aspect is also always improved. This means that the results obtained from different vision models do not present the same artifacts, which tend to disappear when averaging. The significant information is perhaps lost in some of the combined images, but not in all, and the combined filtered image is always better.

Method	FWT-Haar	FWT-7/9	Feauveau	à trous	MMT
Hard thresh.	34.63	35.95	33.27	35.20	34.82
Soft thresh.	32.35	33.83	30.67	32.30	32.43
Donoho hard thresh.	33.19	34.62	31.05	33.98	33.68
Donoho soft thresh.	30.69	32.09	28.73	30.76	31.19
Hierarchical thresh.	-	-	-	35.26	34.89
Hierarchical Wiener	-	-	-	33.35	31.91
Multiresol. Wiener	-	-	-	33.42	31.93
Multiscale Entropy	35.86	36.76	-	35.82	35.56

Table 13.1: PSNR after filtering the simulated image (Lena + Gaussian noise ($\sigma=5$)).

Method	FWT-Haar	FWT-7/9	Feauveau	à trous	MMT
Hard thresh.	31.31	32.97	29.87	32.63	31.80
Soft thresh.	29.72	31.29	28.05	30.03	30.15
Donoho hard thresh.	29.94	31.55	27.68	31.33	30.88
Donoho soft thresh.	28.18	29.66	26.77	28.49	29.09
Hierarchical thresh.	-	-	-	32.75	31.93
Hierarchical Wiener	-	-	-	31.71	30.33
Multiresol. Wiener	-	-	-	31.68	30.24
Multiscale Entropy	32.12	33.39	-	32.41	31.95

Table 13.2: PSNR after filtering the simulated image (Lena + Gaussian noise ($\sigma=10$)).

Method	FWT-Haar	FWT-7/9	Feauveau	à trous	MMT
Hard thresh.	26.82	27.97	26.00	28.58	28.19
Soft thresh.	26.27	27.67	25.85	26.85	27.27
Donoho hard thresh.	25.99	27.46	25.80	27.03	27.42
Donoho soft thresh.	25.29	26.78	25.80	25.85	26.58
Hierarchical thresh.	-	-	-	28.97	28.42
Hierarchical Wiener	-	-	-	28.08	27.96
Multiresol. Wiener	-	-	-	27.25	26.81
Multiscale Entropy	27.45	28.75	-	28.37	27.96

Table 13.3: PSNR after filtering the simulated image (Lena + Gaussian noise ($\sigma=30$)).

Method	FWT-Haar	FWT-7/9	Feauveau	à trous	MMT
Hard thresh.	32.50	33.02	30.48	32.49	31.79
Soft thresh.	30.35	30.97	28.27	29.87	29.59
Donoho hard thresh.	31.04	31.53	28.38	31.23	30.67
Donoho soft thresh.	28.80	29.40	26.70	28.46	28.45
Hierarchical thresh.	-	-	-	32.51	31.82
Hierarchical Wiener	-	-	-	30.59	30.32
Multiresol. Wiener	-	-	-	30.65	30.35
Multiscale Entropy	34.63	34.94	-	34.30	33.97

Table 13.4: PSNR after filtering the simulated image (Landscape + Gaussian noise (sigma=5)).

Method	FWT-Haar	FWT-7/9	Feauveau	à trous	MMT
Hard thresh.	29.32	30.00	27.38	29.88	28.91
Soft thresh.	27.89	28.66	26.18	27.78	27.45
Donoho hard thresh.	28.05	28.74	25.86	28.58	27.98
Donoho soft thresh.	26.53	27.32	25.32	26.50	26.52
Hierarchical thresh.	-	-	-	29.99	28.99
Hierarchical Wiener	-	-	-	29.59	28.04
Multiresol. Wiener	-	-	-	29.64	28.04
Multiscale Entropy	30.80	31.35	-	30.70	30.16

Table 13.5: PSNR after filtering the simulated image (Landscape + Gaussian noise (sigma=10)).

Method	FWT-Haar	FWT-7/9	Feauveau	à trous	MMT
Hard thresh.	25.44	26.01	24.80	26.55	25.90
Soft thresh.	25.03	25.88	24.75	25.33	25.10
Donoho hard thresh.	24.77	25.60	24.72	25.36	25.19
Donoho soft thresh.	24.26	25.29	24.725	24.61	24.51
Hierarchical thresh.	-	-	-	27.07	26.21
Hierarchical Wiener	-	-	-	26.52	25.83
Multiresol. Wiener	-	-	-	25.89	25.24
Multiscale Entropy	26.33	27.11	-	26.88	26.16

Table 13.6: PSNR after filtering the simulated image (Landscape + Gaussian noise (sigma=30)).

Images	PSNR
Lena+5 σ	37.09
Lena+10 σ	33.90
Lena+30 σ	29.44
Landscape+5 σ	34.70
Landscape+10 σ	31.39
Landscape+30 σ	27.42

Table 13.7: PSNR after averaging of seven filtered images.

Chapter 14

Multiscale Entropy Applied to Deconvolution

14.1 Introduction to Deconvolution

Consider an image characterized by its intensity distribution (the “data”) I , corresponding to the observation of a “real image” O through an optical system. If the imaging system is linear and shift-invariant, the relation between the data and the image in the same coordinate frame is a convolution:

$$I = O * P + N \quad (14.1)$$

P is the point spread function (PSF) of the imaging system, and N is additive noise. In practice $O * P$ is subject to non-stationary noise which one can tackle by simultaneous object estimation and restoration [103]. The issue of more extensive statistical modeling will not be further addressed here (see [116, 117, 134]), beyond noting that multiresolution frequently represents a useful framework, allowing the user to introduce a priori knowledge of objects of interest.

In Fourier space we have:

$$\hat{I} = \hat{O}\hat{P} + \hat{N} \quad (14.2)$$

We want to determine $O(x, y)$ knowing I and P . This inverse problem has led to a large amount of work, the main difficulties being the existence of: (i) a cut-off frequency of the point spread function, and (ii) the additive noise (see for example [45]).

Eqn. 14.1 is usually in practice an ill-posed problem. This means that there is not a unique solution.

14.2 Non-Regularized Deconvolution Methods

Inversed filtering

This method, sometimes called *Fourier-quotient method*, computes the Fourier transform of the deconvolved object \hat{O} by a simple division between the image \hat{I} and the PSF \hat{P}

$$\hat{\hat{O}}_u = \frac{\hat{I}_u}{\hat{P}_u} = \hat{O}_u + \frac{\hat{N}_u}{\hat{P}_u} \quad (14.3)$$

This algorithm is very fast. We only need to do a Fourier transform and an inverse Fourier transform. For frequencies close the frequency cut-off, the noise term becomes important, and the noise is amplified. Then in the presence of noise, this method cannot be used. To reduce the

artifact, it is possible to convolve the solution with a smoothing function (Gaussian). Another possibility is to use Wiener filtering which is defined by:

$$\hat{W}_u = \frac{|\hat{S}_u|^2}{|\hat{S}_u|^2 + |\hat{N}_u|^2} \quad (14.4)$$

where $|\hat{S}_u|^2$ and $|\hat{N}_u|^2$ are the spectral density of the signal and the noise. The filter is

$$\hat{W}_{du} = \frac{\hat{P}_u^*}{|\hat{P}_u|^2 + \frac{|\hat{N}_u|^2}{|\hat{O}_u|^2}} \quad (14.5)$$

Wiener filtering has serious drawbacks (artifact creation such as ringing effects), and needs spectral noise estimation. Its advantage is that it is very fast.

Jansson-Van Cittert

Van Cittert [40] restoration is relatively easy to write. We start with $k = 0$ and $O^{(0)} = I$ and we iterate:

$$O^{(n+1)} = O^{(n)} + \alpha(I - P * O^{(n)}) \quad (14.6)$$

where α is a convergence parameter generally taken as 1. When k tends to infinity, we have $O = O + I - p * O$, so $I = P * O$. In Fourier space, the convolution product becomes a product

$$\hat{O}^{(n+1)} = \hat{O}^{(n)} + \alpha(\hat{I} - \hat{P}\hat{O}^{(n)}) \quad (14.7)$$

In this equation, the object distribution is modified by adding a term proportional to the residual. The algorithm converges quickly after only 5 or 6 iterations. But the algorithm generally diverges in the presence of noise. Jansson [98] modified this technique in order to give it more robustness by considering constraints on the solution. If we wish that $A \leq O_k \leq B$, the iteration is

$$O_k^{(n+1)} = O_k^{(k)} + r(k)[I_k - P_k * O_k^{(n)}] \quad (14.8)$$

with:

$$r_k = C[1 - 2(B - A)^{-1} | O_k^{(n)} - 2^{-1}(A + B) |]$$

C being a constant.

Gradient method

The one-step gradient method is provided by the minimization of the norm $\| I - P * O \|$ [110] and leads to:

$$O^{(n+1)} = O^{(n)} + \alpha P^* * [I - (P * O^{(n)})] \quad (14.9)$$

where $P^*(x, y) = P(-x, -y)$. P^* is the transpose of the point spread function, and $O^{(n)}$ is the current estimate of the desired “real image”. In Fourier space we have:

$$\hat{O}^{(n+1)} = \hat{O}^{(n)} + \alpha \hat{P}^* (\hat{I} - \hat{P}\hat{O}^{(n)}) \quad (14.10)$$

This method is more robust than Van Cittert's. The conjugate gradient method provides a faster way to minimize this norm with a somewhat more complex algorithm.

Richardson-Lucy

The Richardson-Lucy method [158, 118] can be derived from Bayes' theorem on conditional probabilities. Given Poisson noise, Shepp and Vardi [169] showed that a maximum likelihood solution was obtained, by use of an expectation-maximization algorithm. Richardson-Lucy image restoration leads to:

$$\begin{aligned} O^{(n+1)} &= O^{(n)}[(I/I^{(n)}) * P^*] \\ I^{(n)} &= P * O^{(n)} \end{aligned} \quad (14.11)$$

This method is commonly used in astronomy. Flux is preserved and the solution is always positive. The positivity of the solution can be obtained too with Van Cittert's and the one-step gradient methods by thresholding negative values in $O^{(n)}$ at each iteration.

Conclusion

All these methods have a severe drawback: noise amplification, which prevents the detection of weak objects, and leads to false detections. To resolve these problems, some constraints must be added to the solution (positivity is already one such constraint, but it is not enough). The addition of such constraints is called regularization. Several regularization methods exist.

14.3 Tikhonov Regularization

Tikhonov regularization [202] consists of minimizing the term:

$$\| I(x, y) - (P * O)(x, y) \| + \lambda \| H * O \| \quad (14.12)$$

where H corresponds to a high-pass filter. This criterion contains two terms. The first, $\| I(x, y) - P(x, y) * O(x, y) \|^2$, expresses fidelity to the data $I(x, y)$, and the second, $\lambda \| H * O \|^2$, expresses smoothness of the restored image. λ is the regularization parameter and represents the trade-off between fidelity to the data and the smoothness of the restored image. Finding the optimal value λ necessitates use of numerical techniques such as cross-validation [84, 81]. This method works well, but computationally it is relatively lengthy and produces smoothed images. This second point can be a real problem when we seek compact structures such as is the case in astronomical imaging.

Tikhonov regularization and wavelet transform

If $w_j^{(I)}$ are the wavelet coefficients of the image I at the scale j , we have:

$$\begin{aligned} \hat{w}_j^{(I)}(u, v) &= \hat{g}(2^{j-1}u, 2^{j-1}v) \prod_{i=j-2}^{i=0} \hat{h}(2^iu, 2^iv) \hat{I}(u, v) \\ &= \frac{\hat{\psi}(2^ju, 2^jv)}{\hat{\phi}(u, v)} \hat{P}(u, v) \hat{O}(u, v) \\ &= \hat{w}_j^{(P)} \hat{O}(u, v) \end{aligned} \quad (14.13)$$

where $w_j^{(P)}$ are the wavelet coefficients of the PSF at the scale j . The wavelet coefficients of the image I are the product of convolution of object O by the wavelet coefficients of the PSF.

To deconvolve the image, we have to minimize for each scale j :

$$\left\| \frac{\hat{\psi}(2^ju, 2^jv)}{\hat{\phi}(u, v)} \hat{P}(u, v) \hat{O}(u, v) - \hat{w}_j^{(I)}(u, v) \right\|^2 \quad (14.14)$$

and for the plane at the lower resolution:

$$\| \frac{\hat{\phi}(2^{n-1}u, 2^{n-1}v)}{\hat{\phi}(u, v)} \hat{P}(u, v) \hat{O}(u, v) - \hat{c}_{n-1}^{(I)}(u, v) \|_2^2 \quad (14.15)$$

n being the number of planes of the wavelet transform (($n - 1$) wavelet coefficient planes and one plane for the image at the lower resolution). The problem does not generally have a unique solution, and we need to do a regularization [202]. At each scale, we add the term:

$$\gamma_j \| w_j^{(O)} \|_2^2 \text{ min} \quad (14.16)$$

This is a smoothness constraint. We want to have the minimum information in the restored object. From equations 14.14, 14.15, 14.16, we find:

$$\hat{D}(u, v) \hat{O}(u, v) = \hat{N}(u, v) \quad (14.17)$$

with:

$$\begin{aligned} \hat{D}(u, v) &= \sum_j | \hat{\psi}(2^j u, 2^j v) |^2 (| \hat{P}(u, v) |^2 + \gamma_j) \\ &\quad + | \hat{\phi}(2^{n-1}u, 2^{n-1}v) \hat{P}(u, v) |^2 \end{aligned}$$

and:

$$\begin{aligned} \hat{N}(u, v) &= \hat{\phi}(u, v) \left[\sum_j \hat{P}^*(u, v) \hat{\psi}^*(2^j u, 2^j v) \hat{w}_j^{(I)} \right. \\ &\quad \left. + \hat{P}^*(u, v) \hat{\phi}^*(2^{n-1}u, 2^{n-1}v) \hat{c}_{n-1}^{(I)} \right] \end{aligned}$$

if the equation is well constrained, the object can be computed by a simple division of \hat{N} by \hat{D} . An iterative algorithm can be used to do this inversion if we want to add other constraints such as positivity. We have in fact a multiresolution Tikhonov regularization. This method has the advantage to furnish a solution quickly, but optimal regularization parameters γ_j cannot be found directly, and several tests are generally necessary before finding an acceptable solution. However, the method can be useful if we need to deconvolve a large number of images with the same noise characteristics. In this case, parameters have to be determined only the first time. In a more general perspective, we prefer to use one of the following iterative algorithms.

14.4 The CLEAN Approach

14.4.1 The CLEAN algorithm

This approach assumes the object is composed of point sources. It tries to decompose the image (called the dirty map), obtained by inverse Fourier transform of the calibrated uv data, into a set of δ -functions. This is done iteratively by finding the point with the largest absolute brightness and subtracting the point spread function (dirty beam) scaled with the product of the loop gain and the intensity at that point. The resulting residual map is then used to repeat the process. The process is stopped when some prespecified limit is reached. The convolution of the δ -functions with an ideal point spread function (clean beam) plus the residual equals the restored image (clean map). This solution is only possible if the image does not contain large-scale structures. The algorithm is:

1. Compute the dirty map $I^{(0)}(x, y)$ and the dirty beam $A(x, y)$.
2. Find the maximum value, and the coordinate (x_{\max}, y_{\max}) of the corresponding pixel in $I^{(i)}(x, y)$.

3. Compute $I^{(i+1)}(x, y) = I^{(i)}(x, y) - \gamma I_{\max} A_m(x, y)$ with $A_m(x, y) = A(x - x_{\max}, y - y_{\max})$ and the loop gain γ inside $[0, 1]$.
4. If the residual map is at the noise level, then go to step 5.
Else $i \leftarrow i + 1$ and go to step 2.
5. The clean map is the convolution of the list of maxima with the clean beam (which is generally a Gaussian).
6. Addition of the clean map and the residual map produces the deconvolved image.

14.4.2 Multiresolution CLEAN

The CLEAN solution is only available if the image does not contain large-scale structures. Wakker and Schwarz (1988) introduced the concept of Multiresolution CLEAN (MRC) in order to alleviate the difficulties occurring in CLEAN for extended sources. The MRC approach consists of building two intermediate images, the first one (called the smooth map) by smoothing the data to a lower resolution with a Gaussian function, and the second one (called the difference map) by subtracting the smoothed image from the original data. Both these images are then processed separately. By using a standard CLEAN algorithm on them, the smoothed clean map and difference clean map are obtained. The recombination of these two maps gives the clean map at the full resolution.

In order to describe how the clean map at the full resolution is obtained from the smoothed and difference clean map, a number of symbols must be defined:

- G = the normalized ($\int G(x)dx = 1$) smoothing function; the width of the function is chosen such that the full-width at half maximum of the smoothed dirty beam is f times larger than the full-width at half maximum of the original dirty beam.
- A = dirty beam
- D = dirty map
- δ = δ -functions
- R = residual after using CLEAN on the map
- B = clean beam with peak value 1
- C = clean map
- s = the scale factor of the dirty beam needed to rescale the smooth dirty beam back to a peak value 1
- r = the scale factor of the dirty beam needed to rescale the smooth clean beam back to a peak value 1
- A_s = normalized smooth dirty beam = $sA * G$
- A_d = normalized difference dirty beam = $1/(1 - \frac{1}{s})(A - \frac{A_s}{s})$
- B_s = normalized smooth clean beam = $rB * G$
- B_d = normalized difference clean beam = $1/(1 - \frac{1}{r})(B - \frac{B_s}{r})$

From the δ -functions found by the CLEAN algorithm, one can restore the dirty map by convolving with the dirty beam and adding the residuals:

$$D = D_s + D_d = \delta_s * A_s + R_s + \delta_d * A_d + R_d \quad (14.18)$$

which can be written also as:

$$D = \left[s\delta_S * G + \frac{s}{s-1}\delta_D * (1-G) \right] * A + R_S + R_D \quad (14.19)$$

If we replace the dirty beam by the clean beam, we obtain the clean map:

$$C = \frac{s}{r}\delta_S * B_S + \frac{s(r-1)}{r(s-1)}\delta_D * B_D + R_S + R_D \quad (14.20)$$

The MRC algorithm needs three parameters. The first fixes the smoothing function G , and the other two are the loop gain and the extra loop gain which are used by CLEAN respectively on the smooth dirty map and difference dirty map.

This algorithm may be viewed as an artificial recipe, but we have shown [187] that it is linked to multiresolution analysis as defined by Mallat [124]. Mallat's theory provides a new representation where a function is a sum of detail structures obtained with the same pattern, the wavelet, with suitable translations and dilations. Wavelet analysis leads to a generalization of MRC from a set of scales.

Our approach allows MRC algorithms to be harmonized with the classical theory of deconvolution.

14.4.3 CLEAN and wavelets

The wavelet transform chosen.

We have seen that there are many wavelet transforms. For interferometric deconvolution, we choose the wavelet transform based on the FFT for the following reasons:

- The convolution product is kept at each scale.
- The data are already in Fourier space, so this decomposition is natural.
- There is a pyramidal implementation available which does not take much memory.

Hence until the end of this chapter, we will consider the use of the pyramidal transform based on the FFT.

Deconvolution by CLEAN in wavelet space.

If $w_j^{(I)}$ are the wavelet coefficients of the image I at the scale j , we get:

$$\hat{w}_j^{(I)}(u, v) = \hat{w}_j^{(P)}\hat{O}(u, v) \quad (14.21)$$

where $w_j^{(P)}$ are the wavelet coefficients of the point spread function at the scale j . The wavelet coefficients of the image I are the convolution product of the object O by the wavelet coefficients of the point spread function.

At each scale j , the wavelet plane $w_j^{(I)}$ can be decomposed by CLEAN ($w_j^{(I)}$ represents the dirty map and $w_j^{(P)}$ the dirty beam) into a set, denoted δ_j , of weighted δ -functions.

$$\begin{aligned} \delta_j = \{ &A_{j,1}\delta(x - x_{j,1}, y - y_{j,1}), A_{j,2}\delta(x - x_{j,2}, y - y_{j,2}), \dots, \\ &A_{j,n_j}\delta(x - x_{j,n_j}, y - y_{j,n_j}) \} \end{aligned} \quad (14.22)$$

where n_j is the number of δ -functions at the scale j and $A_{j,k}$ represents the height of the peak k at the scale j .

By repeating this operation at each scale, we get a set \mathcal{W}_δ composed of weighted δ -functions found by CLEAN ($\mathcal{W}_\delta = \{\delta_1, \delta_2, \dots\}$). If B is the ideal point spread function (clean beam), the estimation of the wavelet coefficients of the object at the scale j is given by:

$$\begin{aligned} w_j^{(E)}(x, y) &= \delta_j * w_j^{(B)}(x, y) + w_j^{(R)}(x, y) \\ &= \sum_k A_{j,k} w_j^{(B)}(x - x_{j,k}, y - y_{j,k}) + w_j^{(R)}(x, y) \end{aligned} \quad (14.23)$$

where $w_j^{(R)}$ is the residual map. The clean map at the full resolution is obtained by the reconstruction algorithm. If we take a Gaussian function as the scaling function, and the difference between two resolutions as the wavelet $(\frac{1}{2}\psi(\frac{x}{2}, \frac{y}{2}) = \phi(x, y) - \frac{1}{2}\phi(\frac{x}{2}, \frac{y}{2}))$, we find the algorithm proposed by Wakker and Schwarz [210]. The MRC algorithm in the wavelet space is:

1. We compute the wavelet transforms of the dirty map, the dirty beam and the clean beam.
2. For each scale j , we decompose by CLEAN the wavelet coefficients of the dirty map into a list of weighted δ -functions δ_j .
3. For each scale j , we convolve δ_j by the wavelet coefficients of the clean beam and we add the residual map $w_j^{(R)}$ to the result in order to obtain the wavelet coefficients of the clean map.
4. We compute the clean map at the full resolution by using the reconstruction algorithm.

Improvements to multiresolution CLEAN.

We apply CLEAN to each plane of the wavelet transform. This allows us to detect at each scale the significant structure. The reconstructed image gives the estimation \hat{O} found by MRC of the object. But MRC does not assume that this estimation is compatible with the measured visibilities. We want:

$$|\hat{O}(u, v) - V_m(u, v)| < \Delta_m(u, v) \quad (14.24)$$

where $\Delta_m(u, v)$ is the error associated with the measure V_m .

To achieve this, we use the position of the peaks determined by the MRC algorithm. We have seen that after the use of CLEAN, we get a list of positions δ_j on each plane j , with approximate heights A_j . In fact, we get a nice description of the significant structures in the wavelet space. The height values are not sufficiently accurate, but CLEAN enhances these structures. So we have to determine heights which reduce the error. We do so using Van Cittert's algorithm [40] which converges, even in the presence of noise, because our system is well regularized. Then, heights of the peaks contained in \mathcal{W}_δ will be modified by the following iterative algorithm:

1. Set $n = 0$ and $\mathcal{W}_\delta^{(0)} = \mathcal{W}_\delta$.
2. Compute $A_{j,l}^{(n+1)} = A_{j,l}^{(n)} + Q_{j,l} \cdot \mathcal{W}_\delta^{(n)}$ so that we then have:

$$\delta_j^{(n+1)} = \{A_{j,1}^{(n+1)} \delta(x - x_{j,1}, y - y_{j,1}),$$

and:

$$\mathcal{W}_\delta^{(n+1)} = \{\delta_1^{(n+1)}, \delta_2^{(n+1)}, \dots\}$$

3. $n = n + 1$ and go to step 1.

\mathcal{Q} is the operator that:

- computes the wavelet coefficients of the clean map $w^{(C)}$ by convolving at each scale $\delta_j^{(n)}$ by the clean beam wavelet $w_j^{(B)}$

$$w_j^{(C)} = \delta_j^{(n)} * w_j^{(B)}$$

- reconstructs the estimated object $O^{(n)}$ at full resolution from $w^{(C)}$
- thresholds the negative values of $O^{(n)}$
- computes the residual $r^{(n)}$ by:

$$\hat{r}^{(n)} = p(V - \hat{O}^{(n)})$$

where p is a weight function which depends on the quality of the measurement V (error bars). A possible choice for p is:

- $p(u, v) = 0$ if we do not have any information at this frequency (i.e. a frequency hole).
- $p(u, v) = 1 - 2\frac{\Delta m(u, v)}{V_m(0, 0)}$ if $\Delta m(u, v)$ is the error associated with the measurement $V_m(u, v)$.

- computes the wavelet transform $w^{(r^{(n)})}$ of $r^{(n)}$
- extracts the wavelet coefficient of $w^{(r^{(n)})}$ which is at the position of the peak $A_{j,l}\delta(x - x_l, y - y_l)$.

The final deconvolution algorithm is:

1. Convolution of the dirty map and the dirty beam by the scaling function.
2. Computation of the wavelet transform of the dirty map which yields $w^{(I)}$.
3. Computation of the wavelet transform of the dirty beam which yields $w^{(D)}$.
4. Estimation of the standard deviation of the noise N_0 of the first plane from the histogram of w_0 . Since we process oversampled images, the values of the wavelet image corresponding to the first scale ($w_0^{(I)}$) are nearly always due to the noise. The histogram shows a Gaussian peak around 0. We compute the standard deviation of this Gaussian function, with a 3-sigma clipping, rejecting pixels where the signal could be significant.
5. Computation of the wavelet transform of the clean beam. We get $w^{(B)}$. If the clean beam is a Dirac delta, then $\hat{w}_j^{(B)}(u, v) = \frac{\psi(2^j u, 2^j v)}{\phi(u, v)}$.
6. Set j to 0.
7. Estimation of the standard deviation of the noise N_j from N_0 . This is done from the study of the variation of the noise between two scales, with the hypothesis of a white Gaussian noise.
8. Detection of significant structures by CLEAN: we get δ_j from $w_j^{(I)}$ and $w_j^{(D)}$. The CLEAN algorithm is very sensitive to the noise. Step 1 of this algorithm offers more robustness. CLEAN can be modified in order to optimize the detection.
9. $j = j + 1$ and go to step 7.

10. Reconstruction of the clean map from $\mathcal{W}_\delta = \{\delta_1, \delta_2, \dots\}$ by the iterative algorithm using Van Cittert's method.

The limited support constraint is implicit because we put information only at the position of the peaks, and the positivity constraint is introduced in the iterative algorithm. We have made the hypothesis that MRC, by providing the coordinates of the peaks, gives the exact position of the information in the wavelet space and we limited the deconvolution problem by looking for the height of the peaks which give the best results. It is a very strong limited support constraint which allows our problem to be regularized. CLEAN is not used as a deconvolution algorithm, but only as a tool to detect the position of structures.

14.5 Regularization from the Multiresolution Support

14.5.1 Noise suppression based on the wavelet transform

We have noted how, in using an iterative deconvolution algorithm such as Van Cittert or Richardson-Lucy, we define $R^{(n)}(x, y)$, the residual at iteration n :

$$R^{(n)}(x, y) = I(x, y) - (P * O^{(n)})(x, y) \quad (14.25)$$

By using the à trous wavelet transform algorithm ([21, 177, 178]), $R^{(n)}$ can be defined as the sum of its p wavelet scales and the last smooth array:

$$R^{(n)}(x, y) = c_p(x, y) + \sum_{j=1}^p w_j(x, y) \quad (14.26)$$

where the first term on the right is the last smoothed array, and w denotes a wavelet scale.

The wavelet coefficients provide a mechanism to extract only the significant structures from the residuals at each iteration. Normally, a large part of these residuals are statistically non-significant. The significant residual ([136, 187]) is then:

$$\bar{R}^{(n)}(x, y) = c_p(x, y) + \sum_{j=1}^p T(w_j(x, y))w_j(x, y) \quad (14.27)$$

T is a function which is defined by:

$$T(w) = \begin{cases} 1 & \text{if } w \text{ is significant} \\ 0 & \text{if } w \text{ is non-significant} \end{cases} \quad (14.28)$$

Assuming that the noise follows a given law, $w_j(x, y)$ is significant if the probability that the wavelet coefficient is due to noise is small.

$$w_j(x, y) \text{ is significant if } \begin{cases} P(W > w_j(x, y)) < \epsilon & \text{if } w_j(x, y) \geq 0 \\ P(W < w_j(x, y)) < \epsilon & \text{if } w_j(x, y) < 0 \end{cases} \quad (14.29)$$

14.5.2 Noise suppression based on the multiresolution support

The multiresolution support

The multiresolution support [184] of an image describes in a logical or boolean way whether an image I contains information at a given scale j and at a given position (x, y) . If $M^{(I)}(j, x, y) = 1$ (or *true*), then I contains information at scale j and at the position (x, y) . M depends on several parameters:

- The input image.
- The algorithm used for the multiresolution decomposition.
- The noise.
- All constraints we want the support additionally to satisfy.

Such a support results from the data, the treatment (noise estimation, etc.), and from knowledge on our part of the objects contained in the data (size of objects, alignment, etc.). In the most general case, a priori information is not available to us.

The multiresolution support of an image is computed in several steps:

1. We compute the wavelet transform of the image.
2. We estimate the noise standard deviation at each scale. We deduce the statistically significant level at each scale.
3. The binarization of each scale leads to the multiresolution support.
4. Modification using a priori knowledge (if desired).

Step 4 is optional. A typical use of a priori knowledge is the suppression of isolated pixels in the multiresolution support in the case where the image is obtained with a point spread function (PSF) of more than one pixel. Then we can be sure that isolated pixels are residual noise which has been detected as significant coefficients. If we use a pyramidal algorithm or a nonlinear multiresolution transform, the same method can be used.

The multiresolution support is obtained by detecting at each scale the significant coefficients. The multiresolution support is defined by:

$$M(j, x, y) = \begin{cases} 1 & \text{if } w_j(x, y) \text{ is significant} \\ 0 & \text{if } w_j(x, y) \text{ is not significant} \end{cases} \quad (14.30)$$

Regularization

In the approach presented in the preceding section, a wavelet coefficient is significant if it is above a threshold. Therefore a coefficient which is less than this threshold is not considered, even if a significant coefficient had been found at the same scale as this coefficient, during previous iterations; and consequently we were justified in thinking that we had found signal at this scale, and at this position. Arising out of this approach, it follows that the wavelet coefficients of the residual image could contain signal, above the set threshold, which is ignored.

In order to conserve such signal, we use the notion of multiresolution support. Whenever we find signal at a scale j and at a position (x, y) , we will consider that this position in the wavelet space belongs to the multiresolution support of the image.

Eqn. 14.27 becomes:

$$\bar{R}^{(n)}(x, y) = c_p(x, y) + \sum_{j=1}^p M(j, x, y) w_j(x, y) \quad (14.31)$$

An alternative approach was outlined in [139] and [184]: the support was initialized to zero, and built up at each iteration of the restoration algorithm. Thus in eqn. 14.31 above, $M(j, x, y)$ was additionally indexed by n , the iteration number. In this case, the support was specified in terms of significant pixels at each scale, j ; and in addition pixels could become significant as the iterations proceeded, but could not be made non-significant. In practice, we have found both of these strategies to be equally acceptable.

Regularization of Van Cittert's Algorithm

Van Cittert's iteration [40] is:

$$O^{(n+1)}(x, y) = O^{(n)}(x, y) + \alpha R^{(n)}(x, y) \quad (14.32)$$

with $R^{(n)}(x, y) = I^{(n)}(x, y) - (P * O^{(n)})(x, y)$. Regularization using significant structures leads to:

$$O^{(n+1)}(x, y) = O^{(n)}(x, y) + \alpha \bar{R}^{(n)}(x, y) \quad (14.33)$$

The basic idea of our method consists of detecting, at each scale, structures of a given size in the residual $R^{(n)}(x, y)$ and putting them in the restored image $O^{(n)}(x, y)$. The process finishes when no more structures are detected. Then, we have separated the image $I(x, y)$ into two images $\tilde{O}(x, y)$ and $R(x, y)$. \tilde{O} is the restored image, which ought not to contain any noise, and $R(x, y)$ is the final residual which ought not to contain any structure. R is our estimate of the noise $N(x, y)$.

Regularization of the One-Step Gradient Method

The one-step gradient iteration is:

$$O^{(n+1)}(x, y) = O^{(n)}(x, y) + P(-x, -y) * R^{(n)}(x, y) \quad (14.34)$$

with $R^{(n)}(x, y) = I(x, y) - (P * O^{(n)})(x, y)$. Regularization by significant structures leads to:

$$O^{(n+1)}(x, y) = O^{(n)}(x, y) + P(-x, -y) * \bar{R}^{(n)}(x, y) \quad (14.35)$$

Regularization of the Richardson-Lucy Algorithm

From eqn. 14.1, we have $I^{(n)}(x, y) = (P * O^{(n)})(x, y)$. Then $R^{(n)}(x, y) = I(x, y) - I^{(n)}(x, y)$, and hence $I(x, y) = I^{(n)}(x, y) + R^{(n)}(x, y)$.

The Richardson-Lucy equation is:

$$O^{(n+1)}(x, y) = O^{(n)}(x, y) \left[\frac{I^{(n)}(x, y) + R^{(n)}(x, y)}{I^{(n)}(x, y)} * P(-x, -y) \right] \quad (14.36)$$

and regularization leads to:

$$O^{(n+1)}(x, y) = O^{(n)}(x, y) \left[\frac{I^{(n)}(x, y) + \bar{R}^{(n)}(x, y)}{I^{(n)}(x, y)} * P(-x, -y) \right] \quad (14.37)$$

Convergence

The standard deviation of the residual decreases until no more significant structures are found. Convergence can be estimated from the residual. The algorithm stops when a user-specified threshold is reached:

$$(\sigma_{R^{(n-1)}} - \sigma_{R^{(n)}}) / (\sigma_{R^{(n)}}) < \epsilon \quad (14.38)$$

14.6 Regularization using a Markov Model

14.6.1 Introduction

Let us briefly review the ingredients of the proposed regularization method. The basic procedure uses a Markov random field as the prior model, the relation (14.1) for image formation consisting typically of blur and superposition of Gaussian noise, and Bayes' formula to obtain the posterior distribution $P(x|y)$.

Random Markov field

A random sequence x is called a Markov random field if for each site s :

$$P(x) > 0 \quad \forall x \in \Omega \quad (14.39)$$

$$P(x_s | x_t; t \in S - \{s\}) = P(x_s | x_t; t \in \mathcal{V}_s) \quad (14.40)$$

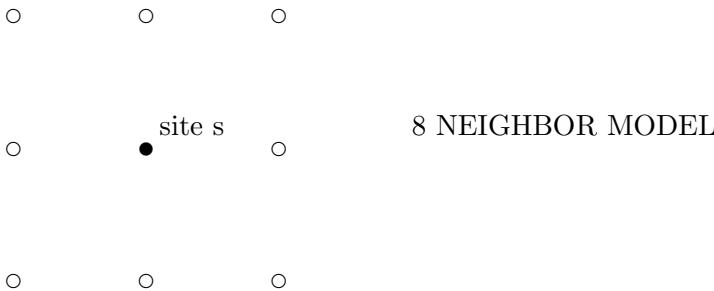
where Ω represents all possible configurations and S is the dimension of the field x . At every site s of the field x , the conditional probability depends on a neighborhood \mathcal{V}_s which is defined by the neighbors t of the site s . Note that \mathcal{V}_s increases with the order of the model.

Examples of models

The four-neighbor system is defined as follows:



Because the edges of the image are not only parallel to the axes, there seems to be need to include diagonal neighbor pixels. Then, the four-neighbor model can be improved as follows:



Gibbs distribution

The prior distribution $P(x)$ can be interpreted as a Gibbs energy. Then, x is a Markov random field if $P(x)$ is a Gibbs distribution such that:

$$P(x) = \frac{1}{Z} \exp(-U(x)) \quad (14.41)$$

where Z is a partition function and $U(x)$ a prior energy such that:

$$U(x) = \sum_{c \in \mathcal{C}} V_c(x) \quad (14.42)$$

where $V_c(x)$ are potentials functions and c is a clique which is defined as a pair of adjacent pixels.

Potential function and line process

In order to preserve edges, the use of a *line process* was introduced by Geman and Geman [83]. In practice, a line process is defined via a potential function ϕ which has particular properties (see [82]).

Suppose $\phi(u)$ has the following properties on $[0, +\infty]$:

- $\phi(0) = 0$
- $\phi(\sqrt{u})$ concave
- $\lim_{u \rightarrow +\infty} \phi(u) = 1$

Then, there exists a function $\psi(l)$ defined on an interval $[0, L]$ such that:

$$\phi(u) = \inf_{0 \leq l \leq L} (lu^2 + \psi(l)) \quad (14.43)$$

and such that $\psi(l)$ has the properties:

- $\psi(0) = 1$
- $\psi(L) = 0$
- $\psi(l)$ strictly decreasing.

The line process is defined as the derivative of $\phi(\sqrt{u})$:

$$l = \phi'(\sqrt{u}) \quad (14.44)$$

Posterior modeling

In the context of Bayesian estimation, the posterior distribution $P(x | y)$ can be written as:

$$P(x | y) \propto P(y | x)P(x) \quad (14.45)$$

where

$$P(y | x) \propto \exp(-U(y | x)) \quad (14.46)$$

and

$$P(x) \propto \exp(-U(x)) \quad (14.47)$$

Then, the posterior $P(x | y)$ is also a Gibbs energy such that:

$$U(x | y) = U(y | x) + U(x) \quad (14.48)$$

The traditional choice in image restoration is:

$$U(y | x) = \frac{\|y - Hx\|^2}{2\sigma^2} \quad (14.49)$$

In the first order case, the prior $P(x)$ is defined such that:

$$U(x) = \sum_{cliques} \phi(x_r - x_s) \quad (14.50)$$

where ϕ is the potential function and $(x_s - x_r)$ is the difference between the values x_r and x_s of the two neighbors inside the clique c . The solution is estimated by maximizing the posterior distribution $P(x | y)$. Then, the energy $U(x | y)$ must be minimized.

$$U(x | y) = \frac{\|y - Hx\|^2}{2\sigma^2} + \lambda \sum_{cliques} \phi(x_r - x_s) \quad (14.51)$$

where λ is the regularizing parameter.

14.6.2 Application

Minimization of the prior energy

The difficulty is to minimize the non-quadratic energy $U(x)$. From the work of Geman and Reynolds [82], the minimization of $U(x)$ is equivalent to minimizing $U(x | l)$ such that:

$$\min_x(U(x)) = \min_{x,l}(U(x, l)) \quad (14.52)$$

where l is the vector of the line variables l_c . According to equation (14.43), it follows that $U(x | l)$ can be written as:

$$U(x, l) = \sum_{\text{cliques } c} l_c (x_r - x_s)^2 + \psi(l_c) \quad (14.53)$$

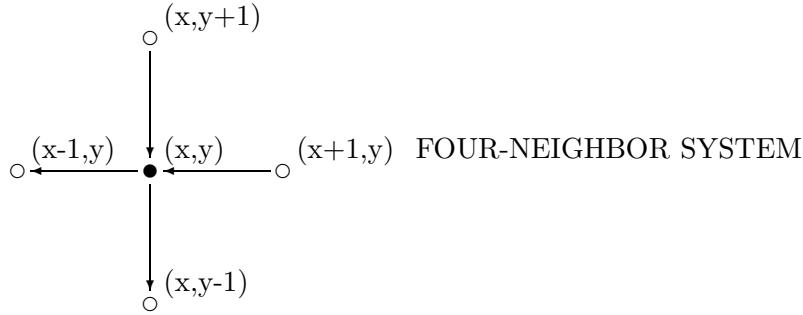
Note that the regularization becomes “half-quadratic” (see [22]) by using equation (14.53):

- With l fixed, $U(x, l)$ is quadratic in x . The minimization in x reduces to the resolution of a linear system.
- With x fixed, the minimum \hat{l}_c is given by the expression $\hat{l}_c = \phi'(u)/2u$ where $u = (x_r - x_s)$.

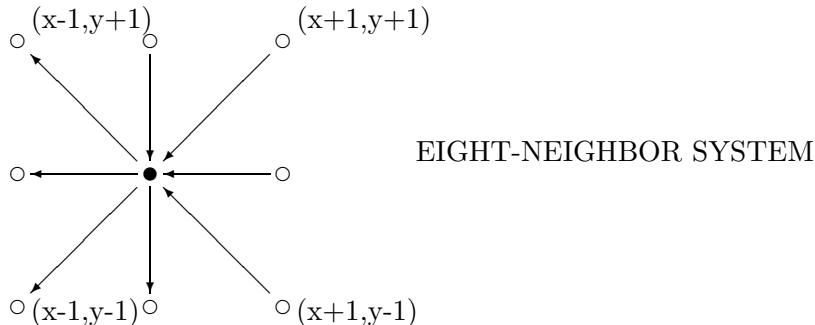
We suppose that the variables \hat{l}_c do not interact with each other. In fact, these *line variables* map the discontinuities (or the edges) of the image x . \hat{l}_c takes a value near zero at the edges and a value L in homogeneous areas.

Line variables system

For the first order Markov case, a line variable is labeled as an arrow between two horizontal or vertical and adjacent pixels (x_r, x_s) . Then, each arrow is associated with a first order clique.



The four neighbor system can be improved by including diagonal adjacencies.



The potential function

Concerning the choice of ϕ , the following convex function proposed in [26] is used:

$$\phi_\delta(u) = |u/\delta| - \ln(1 + |u/\delta|) \quad (14.54)$$

where $u = (x_r - x_s)$ and δ is a scaling parameter. An example is given for $\delta = 500$ in Figure 14.1. It shows that ϕ is quadratic with $u \ll \delta$ and linear with $u \gg \delta$. The derivative is equal to

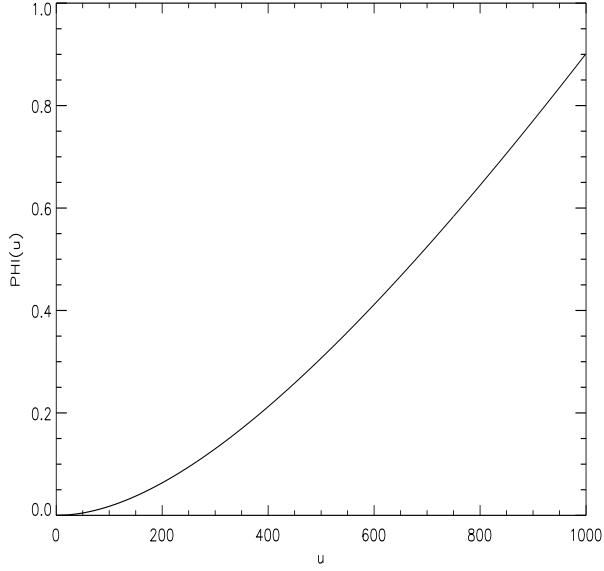


Figure 14.1:

$\phi'_\delta(u) = u/\delta(|u| + \delta)$. Note that updating the line variables is a simplified operation by using the following formula:

$$\hat{l}_c = \phi'_\delta(u)/2u = \frac{1}{2\delta}/(\delta + |u|) \quad (14.55)$$

The prior energy formula

In the case of the four nearest neighbors, the prior energy is defined as:

$$U(x) = \sum_{vertical\ cliques} \phi_\delta(I_{y,x} - I_{y+1,x}) \quad (14.56)$$

$$+ \sum_{horizontal\ cliques} \phi_\delta(I_{y,x} - I_{y,x+1}) \quad (14.57)$$

where $I_{y,x}$ is the pixel intensity in image I at row y and column x .

By adding the following expressions to the terms in equation (14.57), we obtain the energy $U(x)$ of the eight neighbor system:

$$\sum_{diagonal\ cliques} \phi_\delta(I_{y,x} - I_{y+1,x+1}) \quad (14.58)$$

and

$$\sum_{diagonal\ cliques} \phi_\delta(I_{y,x} - I_{y-1,x+1}) \quad (14.59)$$

Instead of using a stochastic approach, we use a single site update algorithm [26] in order to minimize the half-quadratic criterion (14.51). The solution is computed by visiting the entire set of pixels in a determined fashion (checkerboard) to update each value x_{ij} of the estimated solution x at row i and column j . We know that the energy $U(x)$ is quadratic as a function of x_{ij} . Its minimum value is reached at m_{ij} :

$$m_{ij} = x_{ij} + \frac{[H^t y]_{ij} - [H^t H x]_{ij} - 2\sigma^2 \lambda \sum l_c (x_{ij} - x_c)}{[H^t H]_{ij,ij} + 2\sigma^2 \lambda \sum l_c} \quad (14.60)$$

where the sums extend to the neighborhood of the currently visited pixel x_{ij} . The algorithm is initialized with an image where all pixels are equal to zero.

Parameter estimation

This parameter allows to fix the threshold under which the smoothness of the solution is preserved and above which discontinuities or edges stay in the estimated solution. The value δ is estimated from the observed image y by looking at the evolution of edges between adjacent pixels. In fact, one method of selecting δ would be to determine the global maximal difference between two adjacent pixels for all possible cliques of the observed image. If δ is too high, then the estimated solution will be smooth.

This parameter balances fidelity to the prior constraints and fidelity to the data. In every experiment, λ is empirically tuned to obtain visually good results. In fact, since appropriate values for δ are more or less estimated, the value for λ is not an evident choice. However, if λ is too high, the solution is over-regularized and if λ is too small, the solution is not stabilized.

14.7 Multiscale Entropy Deconvolution

14.7.1 The principle

The standard maximum entropy method (MEM) has been described in the preceding chapter. The most realistic solution is that which minimizes the amount of information, but remains compatible with the data. By the MEM method, minimizing the information is equivalent to maximizing the entropy and the functional to minimize is

$$J(O) = \sum_{k=1}^N \frac{(I_k - (P * O)_k)^2}{2\sigma_I^2} - \alpha H(O) \quad (14.61)$$

where H is either the Frieden or the Gull and Skilling entropy.

Using the Shannon entropy or the multiscale entropy, minimizing the information is equivalent to minimizing the entropy and the functional to minimize is

$$J(O) = \sum_{k=1}^N \frac{(I_k - (P * O)_k)^2}{2\sigma_I^2} + \alpha H(O) \quad (14.62)$$

We have seen that in the case of Gaussian noise, H is given by the energy of the wavelet coefficients. We have

$$J(O) = \sum_{k=1}^N \frac{(I_k - (P * O)_k)^2}{2\sigma_I^2} + \alpha \sum_{j=1}^l \sum_{k=1}^{N_j} \frac{w_{j,k}^2}{2\sigma_j^2} \quad (14.63)$$

where σ_j is the noise at scale j , N_j the number of pixels at the scale j , σ_I the noise standard deviation in the data, and l the number of scales.

Rather than minimizing the amount of information in the solution, we may prefer to minimize the amount of information which can be due to the noise. The function is now:

$$J(O) = \sum_{k=1}^N \frac{(I_k - (P * O)_k)^2}{2\sigma_I^2} + \alpha H_n(O) \quad (14.64)$$

Using the N2-MSE approach, H_n is defined by

$$H_n(w_{j,k}) = \sum_{j=1}^l \sum_{k=1}^{N_j} \int_0^{|w_{j,k}|} p_n(|w_{j,k}| - u) \left(\frac{\partial h(x)}{\partial x} \right)_{x=u} du \quad (14.65)$$

which gives for Gaussian noise

$$H_n(X) = \sum_{j=1}^l \sum_{k=1}^{N_j} \frac{1}{\sigma_j^2} \int_0^{|w_{j,k}|} u \operatorname{erf}\left(\frac{|w_{j,k}| - u}{\sqrt{2}\sigma_j}\right) du \quad (14.66)$$

The solution is found by computing the gradient $\nabla(J(O))$ and performing the following iterative schema:

$$O^{n+1} = O^n - \gamma \nabla(J(O^n)) \quad (14.67)$$

As for the multiscale entropy filtering, there are several ways to use the parameter α . In the same way, we can consider an α_j per scale, and introduce a kind of adaptive regularization, depending on the signal-to-noise ratio of the input data wavelet coefficients.

14.7.2 The parameters

In order to introduce flexibility in the way we restore the data, we introduce two parameters $\beta_{j,k}$ and $\alpha_{j,k}$ which allow us to weight respectively the two terms of the equation to be minimized:

$$J(O) = \frac{1}{2\sigma_I^2} \sum_{k=1}^N (c_{l,k}(R) + \sum_{j=1}^l \beta_{j,k} w_{j,k}(R))^2 + \sum_{j=1}^l \sum_{k=1}^{N_j} \alpha_{j,k} h(w_{j,k}(O))$$

where $R = I - P * O$, and $R = c_{l,k}(R) + \sum_{j=1}^l w_{j,k}(R)$ ($w_{j,k}(R)$ are the wavelet coefficients of R using the à trous algorithm, and $w_{j,k}(O)$ are the wavelet coefficients of O).

We consider three approaches for estimating $\beta_{j,k}$

1. no weighting: $\beta_{j,k} = 1$
2. soft weighting: $\beta_{j,k} = p_s(w_{j,k}(I))$
In this case, $\beta_{j,k}$ is equal to the probability that the input data wavelet coefficient is due to signal (and not to noise).
3. hard weighting: $\beta_{j,k} = 0$ or 1 depending on $p_n(w_{j,k}(I))$ ($p_n(w_{j,k}(I)) = 1 - p_s(w_{j,k}(I))$). This corresponds to using only significant input data wavelet coefficients. If M is the multiresolution support of the input image I , then $\beta_{j,k} = M(j, k)$.

$\alpha_{j,k}$ is the product of three values: $\alpha_{j,k} = \alpha_u \alpha_j \beta'_{j,k}$.

- α_u is a user parameter (defaulted to 1) which allows us to control the smoothness of the solution. Increasing α_u produces a smoother solution.
- α_j is a constant which depends only on the PSF.
- $\beta'_{j,k}$ depends on the input data and can take the following value:
 1. no regularization ($\beta'_{j,k} = 0$): only the first term of the functional is minimized.
 2. no protection from regularization ($\beta'_{j,k} = 1$): the regularization is applied at all positions and at all the scales.
 3. soft protection ($\beta'_{j,k} = p_n(w_{j,k}(I))$): the regularization becomes adaptive, depending on the probability that the input wavelet coefficient is due to noise.
 4. hard protection ($\beta'_{j,k} = 0$ or 1 depending on $p_n(w_{j,k}(I))$). If M is the multiresolution support of the input image I , then $\beta_{j,k} = 1 - M(j, k)$.
 5. Soft + hard protection: ($\beta'_{j,k} = 0$ or $p_n(w_{j,k}(I))$ depending on $p_n(w_{j,k}(I))$).

We easily see that choosing a hard weighting and no regularization leads to deconvolution from the multiresolution support.

The problem is now to find how to calculate the α_j values. Taking all α_j equal to 1 would mean that we try to restore all bands in the same way. However if the PSF is very large, structures which may appear roughly the size of the pixel are certainly artifacts due to the noise. At the contrary, if the PSF is very extended, there is no reason to not consider as true a compact structure in the solution. This reason leads us to consider that the regularization should not be applied in the same way at all scales, but should depend on the PSF. Since at each iteration the solution is updated by adding a part the residual (which contains the noise) convolved by the PSF, we estimate α_j by the following method:

- we generate a random noise N_r of standard deviation equal to 1.
- we convolve it by the PSF: $C_r = N_r * P$.

- we take the wavelet transform of C_r .
- we take the wavelet transform of N_r .
- we calculate the standard deviation at each scale of C_r : $\sigma_j^r(w_j(C_r))$
- we calculate the standard deviation at each scale of N_r : $\sigma_j^e(w_j(N_r))$
- we calculate the value: $a_j = \frac{\sigma_j^e}{\sigma_j^r}$
- we normalize a_j by $a_j = a_j / \max(a_i)$ (for $i = 1..l$).

By this method, the regularization will be stronger at small scales if the PSF is large.

The Intrinsic Correlation Function

In many cases, there is no sense to try to deconvolve an image at the resolution of the pixel (especially when the PSF is very large). The idea to limit the resolution is relatively old, because it is already this concept which is used in the CLEAN algorithm [91]. Indeed the Clean-Beam fixes the resolution in the final solution. This principle was also developed by Lannes [111] under a different form. Finally this concept have been re-invented, first by Gull and Skilling, who have called the Clean-Beam the *Intrinsic Correlation Function* (ICF), and more recently by Magain [119].

The ICF is usually a Gaussian, but in some cases it may be useful to take another function. For example, if we want to compare two images I_1 and I_2 which are obtained with two wavelengths or with two different instruments, their PSFs P_1 and P_2 will certainly be different. The classic way would be convolve I_1 with P_2 and I_2 with P_1 , so we are sure that both are at the same resolution. But unfortunately, we lose some resolution. Deconvolving both images is generally not possible because we can never be sure that both solutions O_1 and O_2 will have the same resolution.

A solution would be to deconvolve only the image which has the worse resolution (say I_1), and to limit the deconvolution to the second image resolution (I_2). Then, we just have to take P_2 for the ICF. The deconvolution problem is to find \tilde{O} (hidden solution) such that:

$$I_1 = P_1 * P_2 * \tilde{O} \quad (14.68)$$

and our real solution O_1 at the same resolution as I_2 is just obtained by convolving \tilde{O} by P_2 . O_1 and I_2 can then be compared.

Introducing an ICF G in the deconvolution equation leads to just considering a new PSF P' which is the convolution between P and G . The deconvolution is carried out using P' , and the solution must be reconvolved by G at the end. By this way, the solution has a limited resolution, but aliasing may occur during the iterative process, and it is not sure that the artifacts will disappear after the re-convolution with G . Magain [119] has proposed an original alternative to this problem, by assuming that the PSF can be considered as the convolution product of the terms, the ICF G and an unknown S , $P = G * S$. Using S instead of P in the deconvolution process, and a sufficient large FWHM value for G imply that the Shannon sampling theorem is never violated. But the problem is now to calculate S , knowing P and G , which is again a deconvolution problem. Unfortunately, this delicate point was not discussed in the original paper. Propagation of the error on S estimation in the final solution has also until now never been investigated, but this approach seems promising.

ICF calculation

This section describes how to calculate the Full Width at Half Maximum for a given sampling, in order to not violate the Shannon sampling theorem. Gaussian functions are generally chosen for the ICF. The resolution to achieve is fixed by its standard deviation, or its Full Width at Half Maximum (FWHM=2.34 σ). As the Fourier transform of a Gaussian of standard deviation σ is

also a Gaussian of standard deviation $\sigma_\nu = \frac{N}{2\pi\sigma_g}$, (N being the number of pixels), we can estimate the smallest FWHM which does not violate the Shannon sampling theorem. It is impossible in theory, but in practice we can consider that values under a given ϵ have no computation effect. The Shannon theorem is experimentally respected if

$$\exp \frac{\nu^2}{2\sigma_\nu^2} < \epsilon \text{ when } u > \frac{N}{2} \quad (14.69)$$

For $u = \frac{N}{2}$, we have: $\exp \frac{-\pi^2\sigma_g^2}{2} < \epsilon$
Then the smallest ICF standard deviation σ_g is given by

$$\sigma_g = \sqrt{\frac{-2 \log \epsilon}{\pi^2}} \quad (14.70)$$

Table 14.1 gives the σ_g values for different values of ϵ . If the resolution to achieve is smaller than σ_g , it means that the solution sampling must be fainter than the data sampling.

Table 14.1: ICF Standard deviation.

ϵ	σ_g	FWHM
10^{-3}	1.18	2.77
10^{-4}	1.37	3.20
10^{-5}	1.53	3.57
10^{-7}	1.81	4.23
10^{-10}	2.16	5.05
10^{-20}	3.05	7.15

Chapter 15

Multiscale Entropy applied to Background Fluctuation Analysis

The mean entropy vector

The multiscale entropy has been defined by:

$$H(X) = \sum_{j=1}^l \sum_{k=1}^N h(w_j) \quad (15.1)$$

with $h(w_j = \ln(p(w_j(k))))$. In order to study the behavior of the information at a given scale, we prefer to calculate the mean entropy vector E defined by:

$$E(j) = \frac{1}{N} \sum_{k=1}^N h(w_j) \quad (15.2)$$

$E(j)$ gives the mean entropy at the scale j . From the mean entropy vector, we have statistical information on each scale separately. Having a noise model, we are able to calculate (generally from simulations) the mean entropy vector $E^{(noise)}(j)$ resulting from a pure noise. Then we define the normalized mean entropy vector by

$$E_n(j) = \frac{E(j)}{E^{(noise)}(j)} \quad (15.3)$$

Figure 15.1 shows the result of a simulation. Five simulated images were created by adding n sources (i.e., point sources, or idealized stars) to an 1024×1024 image containing Gaussian noise of standard deviation equal to 1. The n sources are identical, with a maximum equal to 1, and standard deviation equal to 2. Defining the signal to noise ratio (SNR) as the ratio between the standard deviation in the smallest box which contains at least 90% of the flux of the source, and the noise standard deviation, we have a SNR equal to 0.25. The sources are not detectable in the simulated image, nor in its wavelet transform. Figure 15.2 shows a region which contains a source at the center. It is clear there is no way to find this kind of noisy signal. The five images were created using a number of sources respectively equal to 0,50,100,200 and 400, and the simulation was repeated ten times with different noise maps in order to have an error bar on each entropy measurement. For the image which contains 400 sources, the number of pixels affected by a source is less than 2.5%.

When the number of sources increases, the difference between the multiscale entropy curves increases. Even if the sources are very faint, the presence of signal can be clearly detected using the mean entropy vector. But it is obvious that the positions of these sources remain unknown.

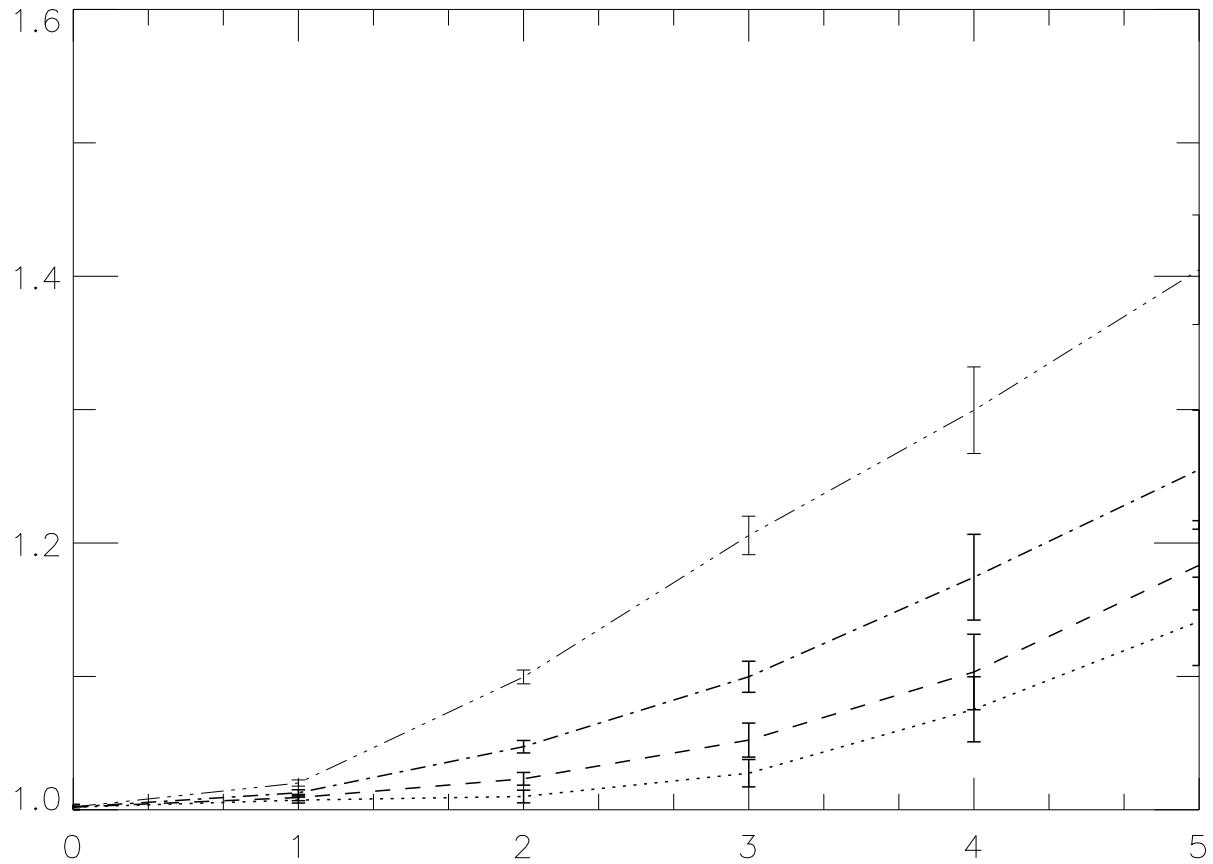


Figure 15.1: Mean entropy versus the scale of 5 simulated images containing undetectable sources and noise. Each curve corresponds to the multiscale transform of one image. From top to bottom, the image contains respectively 400, 200, 100, 50 and 0 sources.

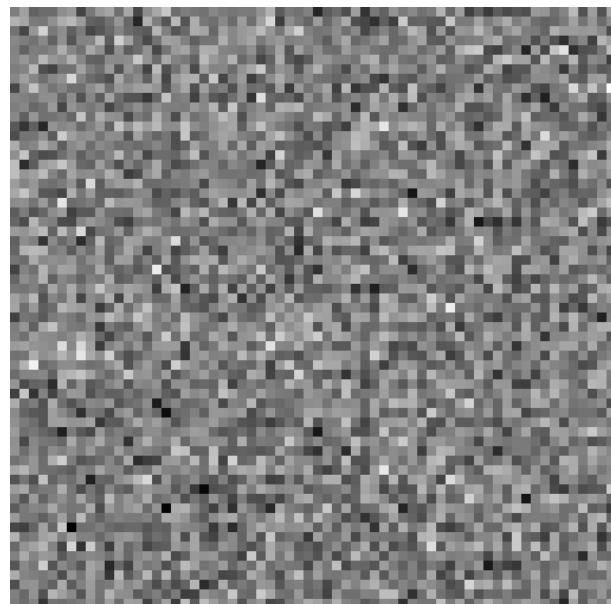


Figure 15.2: Region of a simulated image containing an undetectable source at the center.

Chapter 16

MR/2 Programs

16.1 Probability in wavelet space: mw_proba

The program *mw_proba* computes the probability of each wavelet coefficient to be due to signal (i.e. not due to noise).

USAGE: `mw_proba options image_in mr_file_out`

where options are :

- `[-t type_of_multiresolution_transform]`
- `[-m type_of_noise]`
- `[-g sigma]`
- `[-c gain,sigma,mean]`
- `[-n number_of_scales]`
- `[-S SizeBlock]`

- `[-N NiterSigmaClip]`

- `[-R RMS_Map_File_Name]`

Examples:

- `mw_prob image_in.d MR_File.out.mr`
Compute the probability of each wavelet coefficient of an image to be due to signal (i.e. not due to noise).
- `mr_extract -s 1 MR_File.out.mr scale1.d`
Create an image from the first scale of the multiresolution file.

16.2 Entropy of an image: mw_entrop

The program *mw_entrop* computes the information (entropy) and the signal information of an image (N1-MSE approach). The two output files *mr_file1_out* and *mr_file2_out* are multiresolution files (“.mr”) which contain respectively the information and the signal information relative to the wavelet coefficients. The program calculates and prints to the screen the mean entropy and signal entropy per band.

USAGE: mw_entrop options image_in mr_file1_out mr_file2_out

where options are:

- [-t type_of_multiresolution_transform]
- [-m type_of_noise]
- [-g sigma]
- [-c gain,sigma,mean]
- [-n number_of_scales]
- [-S SizeBlock]
- [-N NiterSigmaClip]
- [-R RMS_Map_File_Name]

Example:

- mw_entrop image_in.d MR_File1_out.mr MR_File2_out.mr
Compute the multiscale entropy of an image

16.3 Filtering

16.3.1 1D filtering: mw1d_filter

The program *mw1d_filter* filters a one dimensional signal using the multiscale entropy method (N2-MSE approach).

USAGE: mw1d_filter options signal_in signal_out

where options are :

- [-t type_of_multiresolution_transform]
- [-m type_of_noise]
- [-g sigma]
- [-c gain,sigma,mean]
- [-s NSigma]
- [-n number_of_scales]
- [-e epsilon]
Convergence parameter. Default is $1e^{-4}$.
- [-i number_of_iterations]
Maximum number of iterations. Default is 10.
- [-G RegulParam]
Regularization parameter. Default is 1.
- [-D]
The regularization parameter is a function of the SNR in the data. Default is no.
- [-w FilterCoefFileName]
Write to disk the filtered wavelet coefficient.
- [-v]
Verbose. Default is no.

Examples:

- `mw1d_filter sig_in.fits sig_out.d`
filters a signal by the multiscale entropy method, assuming Gaussian noise (its standard deviation is automatically estimated).
- `mw1d_filter -G 2 sig_in.fits sig_out.fits`
Same as before, but the regularization will be stronger, and the solution more smooth.
- `mw1d_filter -G 2 -D sig_in.fits sig_out.fits`
The regularization is adaptive, depending on the wavelet SNR.
- `mw_filter -G 2 -D -s 5 sig_in.fits sig_out.fits`
Same as before, preserving feature in the wavelet space greater than 5σ instead of the default 3σ value.

16.3.2 2D filtering: mw_filter

The program *mw_filter* filters an image using the multiscale entropy method (N2-MSE approach).

USAGE: mw_filter options image_in image_out

where options are:

- **[-T Type_of_Regularization]**
 1. Use a fixed user Alpha value.
 2. Estimate the optimal Alpha.
 3. Estimate one Alpha value per band.

Default is 1.
- **[-D]**
Wavelet coefficients with a high signal to noise ratio are not regularized. For a lower SNR, the Alpha parameter is modified using the SNR. Default is no regularization.
- **[-t type_of_multiresolution_transform]**
- **[-m type_of_noise]**
Noise models 1 to 9 are available.
- **[-g sigma]**
- **[-c gain,sigma,mean]**
- **[-s NSigma]**
- **[-S SizeBlock]**
- **[-N NiterSigmaClip]**
- **[-R RMS_Map_File_Name]**
- **[-n number_of_scales]**
- **[-e epsilon]**
Convergence parameter. Default is $1e^{-4}$.
- **[-i MaxIter]**
Maximum number of iterations. Default is 20.
- **[-G RegulParam]**
Regularization parameter. Default is 1.

- **[-C]**
Convergence parameter. Only used when regularization type is equal to 2 or 3. Default is 0.01.
- **[-P]**
Apply the positivity constraint. If set, the solution cannot have negative values.
- **[-v]**
Verbose. Default is no.

Examples:

- `mw_filter image_in.d ima_out.d`
Filters an image by the multiscale entropy method, assuming Gaussian noise (its standard deviation is automatically estimated).
- `mw_filter -G 10 -P image_in.d ima_out.d`
Same as before, but the regularization will be stronger, and the solution more smooth. Positivity constraint is imposed.
- `mw_filter -G 10 -D image_in.d ima_out.d`
The regularization is adaptive, depending on the wavelet SNR.
- `mw_filter -T 2 image_in.d ima_out.d`
The regularization parameter is automatically estimated in an iterative way.
- `mw_filter -T 2 -G 2.5 image_in.d ima_out.d`
Same as before, but the estimated parameter is multiplied by 2.5 (the solution becomes more smooth).
- `mw_filter -T 3 image_in.d ima_out.d`
On regularization parameter per scale is now used. All are automatically estimated in an iterative way.
- `mw_filter -T 3 -D -s 5 image_in.d ima_out.d`
Same as before, preserving also feature in the wavelet space greater than 5σ .

16.3.3 2D combined filtering: `mw_comb_filter`

The program `mw_comb_filter` filters an image using the combined filtering method.

USAGE: `mw_comb_filter options image_in image_out`

where options are :

- **[-t type_of_multiresolution_transform]**
- **[-O]**
Filtering by an opening (erosion+dilation).
The structural element is a circle of size 3
- **[-M type_of_multiresolution_transform]**
Filtering using the MEM method (estimating automatically one Alpha value per band). Default multiresolution transform is the à trous algorithm.
- **[-D]**
Wavelet coefficient with a high signa to noise ratio are not regularized. The Alpha parameter is modified using the SNR. Default is no. Valid only when “-d” or “-M” option is set.
- **[-G RegulParam]**
Regularization parameter for MEM method. Valid only when “-d” or “-M” option is set. Default is 1.

- [-m type_of_noise]
 1. Gaussian noise
 2. Poisson noise
 3. Poisson noise + Gaussian noise
 4. Multiplicative noise

Default is Gaussian noise.

- [-g sigma]
- [-c gain,sigma,mean]
- [-s NSigma]
- [-n number_of_scales]
- [-d]

Use default combined methods (methods are a thresholding by the à trous algorithm, the multiscale median transform, the orthogonal wavelet transform, the haar transform, and the multiscale entropy method. “-d” option is equivalent to the set of following options together: “-t 2 -t 4 -t 14 -t 18 -M 2”.

- [-v]
Verbose. Default is no.

Examples:

- mw_comb_filter -d image_in.d ima_out.d
Filters an image by the combined filtered method (default option), assuming Gaussian noise (its standard deviation is automatically estimated).
- mw_comb_filter -t 2 -t 4 -t 14 -t 18 -M 2 image_in.d ima_out.d
Ditto.
- mw_comb_filter -d -O -t 1 image_in.d ima_out.d
Same as before, with in addition a filtering using the linear à trous algorithm, and the morphological operators.
- mw_comb_filter -t 14 -M 14 image_in.d ima_out.d
Filtering using only two methods: thresholding the orthogonal WT, and the MEM method.
- mw_comb_filter -t 14 -M 14 -G 2.5 image_in.d ima_out.d
Ditto, but the estimated MEM parameter is multiplied by 2.5.

16.4 Deconvolution: mw_deconv

The program *mw_deconv* deconvolves an image using the multiscale entropy method.

USAGE: mw_deconv options image_in psf_in image_out

where options are :

- [-t type_of_multiresolution_transform]
Default is 2.
- [-H EntropyFunction]
 1. Entropy = H = Wavelet coefficient energy.

2. Entropy = H_n = Noise information (for Gaussian noise only), using N2-MSE approach.

Default is 1. For Gaussian noise, default is 2.

- **[-g sigma]**
- **[-c gain,sigma,mean]**
- **[-m type_of_noise]**
Noise models 1 to 9 are available.
- **[-n number_of_scales]**
- **[-s NSigma]**
- **[-i number_of_iterations]**
Maximum number of iterations. Default is 500.
- **[-K]**
Suppress the last scale. Default is no.
- **[-R RMS_Map_File_Name]**

- **[-P]**
Suppress the positivity constraint.
- **[-C]**
Convergence parameter. Default is 1.
- **[-f ICF_Fwhm]**
Intrinsic correlation function.
Fwhm = Full-width at half maximum.
- **[-I ICF_FileName]**
Intrinsic correlation function file.
- **[-F First_Guess]**
Input solution file.
- **[-W DataWeightingType]**
 1. no weighting
 2. soft weighting
 3. hard weighting

Default is 3.
- **[-A RegulProtectType]**
 - 0: no regularization (all protected)
 - 1: no protection from regularization
 - 2: soft protection
 - 3: hard protection
 - 4: soft + hard protection

Default is 3.
- **[-G RegulParam]**
Regularization parameter. Default is 1.

- **[-M NSigmaObj]**
NSigma level for object multiresolution determination. Default is 3. With hard weighting (-W 3), default is 1.
- **[-r residual_file_name]**
Write the residual to disk.
- **[-S]**
Do not shift automatically the maximum of the PSF at the center.
- **[-v]**
Verbose. Default is no.

Example:

- `mw_deconv image_in.d psf ima_out.d`
Deconvolve an image using default options
- `mw_deconv -W 3 -A 0 image_in.d psf ima_out.d`
Deconvolution from the multiresolution support.
- `mw_deconv -A 3 image_in.d psf ima_out.d`
Regularization with a protection of high SNR wavelet coefficients.
- `mw_deconv -A 3 -G 2.5 image_in.d psf ima_out.d`
Ditto, but increases the regularization (solution more smooth).
- `mw_deconv -f 3 image_in.d psf ima_out.d`
Impose that the solution is the result of the convolution product between a Gaussian (full-width at half maximum equal to 3) and a hidden solution.

16.5 IDL Routines

16.5.1 Introduction

A set of routines has been developed in IDL. Starting IDL using the script program *mre* allows the user to get the multiresolution environment, and all routines described in the following can be called. An online help facility is also available by invoking the *mrh* program under IDL.

16.5.2 mw1d_filter

Filter a 1D signal by the the multiscale entropy.

USAGE: `mw1d_filter, Signal, Result, Opt=Opt`

where

- *Signal*: input one-dimensional IDL array.
- *Result*: output one-dimensional IDL array (filtered signal).
- *Opt*: string which contains the different options (see the *mw1d_filter* C++ program).

16.5.3 mw1d_predict

Considering a temporal signal S with n measures ($S[0..n - 1]$), *mr1d_predict* estimates (or predicts) the next values $S[n..n + dt]$. A multiscale transformation is first applied, followed by filtering in the wavelet space. At each scale, a predictor is then applied, and the predicted signal is obtained from the reconstruction of the predicted signal.

USAGE: `Result = MW1D_PREDICT(Signal, wave=wave, PredWave=PredWave, NPredict=NPredict, Nscale=Nscale, OPT=OPT, NCoef=NCoef)`

where

- *wave*: output 2D IDL array; filtered wavelet coefficient
- *PredWave*: output 2D IDL array; Predicted wavelet coefficient
- *NPredict*: number of values to predict. Default is one.
- *Nscale*: number of scales used for the wavelet transform
- *NCoef*: Number of coefficients used for the prediction. Default is 3.
- *OPT*: string which contains the differents options accepted by the *mw1d_filter* C++ program.

16.5.4 mw_filter

Filter an image by the multiscale entropy. This routine is calling the C++ executable *mw_filter*. The keyword “OPT” allows all options described in the section corresponding to the *mw_filter* program.

USAGE: `mw_filter, Data, FilterData, opt=opt`

Data is an image (2D IDL array), and *FilterData* is the result of the filtering.

16.5.5 mw_deconv

Deconvolve an image by the multiscale entropy. This routine calls the C++ executable `mw_deconv`. The keyword “OPT” allows all options described in the section corresponding to the *mw_deconv* program.

USAGE: `mw_deconv, Data, PSF, DeconvData, opt=opt`

Examples:

- `mw_deconv, Imag, Psf, Result`
deconvolve an image with all default options.
- `mw_deconv, Imag, Psf, Result, OPT=' -i 30 -e 0'`
same example, but impose the number of iterations to be 30.

Appendix A: The “À Trou” Wavelet Transform Algorithm

In a wavelet transform, a series of transformations of a signal is generated, providing a resolution-related set of “views” of the signal. The properties satisfied by a wavelet transform, and in particular by the *à trous* wavelet transform, are further discussed by Bijaoui et al. [21]. Extensive literature exists on the wavelet transform and its applications ([51, 39, 161, 191]). The discrete *à trous* algorithm is described in [92, 168].

We consider spectra, $\{c_0(k)\}$, defined as the scalar product at samples k of the function $f(x)$ with a scaling function $\phi(x)$ which corresponds to a low pass filter:

$$c_0(k) = \langle f(x), \phi(x - k) \rangle \quad (16.1)$$

The scaling function is chosen to satisfy the dilation equation:

$$\frac{1}{2} \phi\left(\frac{x}{2}\right) = \sum_l h(l) \phi(x - l) \quad (16.2)$$

where h is a discrete low-pass filter associated with the scaling function ϕ . This means that a low-pass filtering of the signal is, by definition, closely linked to another resolution level of the signal. The distance between levels increases by a factor 2 from one scale to the next.

The smoothed data $c_j(k)$ at a given resolution j and at a position k is the scalar product

$$c_j(k) = \frac{1}{2^j} \langle f(x), \phi\left(\frac{x - k}{2^j}\right) \rangle \quad (16.3)$$

This is consequently obtained by the convolution:

$$c_j(k) = \sum_l h(l) c_{j-1}(k + 2^{j-1}l) \quad (16.4)$$

The signal difference w_j between two consecutive resolutions is:

$$w_j(k) = c_{j-1}(k) - c_j(k) \quad (16.5)$$

or:

$$w_j(k) = \frac{1}{2^j} \langle f(x), \psi\left(\frac{x - k}{2^j}\right) \rangle \quad (16.6)$$

Here, the wavelet function ψ is defined by:

$$\frac{1}{2} \psi\left(\frac{x}{2}\right) = \phi(x) - \frac{1}{2} \phi\left(\frac{x}{2}\right) \quad (16.7)$$

Equation 16.6 defines the discrete wavelet transform, for a resolution level j .

For the scaling function, $\phi(x)$, the B-spline of degree 3 was used in our calculations. As a filter we use $h = (\frac{1}{16}, \frac{1}{4}, \frac{3}{8}, \frac{1}{4}, \frac{1}{16})$. See Starck (1993) for discussion of linear and other scaling functions. Here we have derived a simple algorithm in order to compute the associated wavelet transform:

1. We initialize j to 0 and we start with the data $c_j(k)$.
2. We increment j , and carry out a discrete convolution of the data $c_{j-1}(k)$ using the filter h . The distance between the central sample and the adjacent ones is 2^{j-1} .
3. After this smoothing, we obtain the discrete wavelet transform from the difference $c_{j-1}(k) - c_j(k)$.
4. If j is less than the number p of resolutions we want to compute, then we go to step 2.
5. The set $\mathcal{W} = \{w_1, \dots, w_p, c_p\}$ represents the wavelet transform of the data.

A series expansion of the original signal, c_0 , in terms of the wavelet coefficients is now given as follows. The final smoothed array $c_p(x)$ is added to all the differences w_j :

$$c_0(k) = c_p + \sum_{j=1}^p w_j(k) \quad (16.8)$$

This equation provides a reconstruction formula for the original signal. At each scale j , we obtain a set $\{w_j\}$ which we call a wavelet scale. The wavelet scale has the same number of samples as the signal.

Appendix B: Noise Modeling the Wavelet Space

Gaussian noise

We have the probability density

$$p(w_j(x, y)) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-w_j(x, y)^2/2\sigma_j^2} \quad (16.9)$$

So we need to estimate, in the case of Gaussian noise models, the noise standard deviation at each scale. These standard deviations can be determined analytically in the case of some transforms, including the à trous transform, but the calculations can become complicated.

The appropriate value of σ_j in the succession of wavelet planes is assessed from the standard deviation of the noise σ_I in the original image I , and from study of the noise in the wavelet space. This study consists of simulating an image containing Gaussian noise with a standard deviation equal to 1, and taking the wavelet transform of this image. Then we compute the standard deviation σ_j^e at each scale. We get a curve σ_j^e as a function of j , giving the behavior of the noise in the wavelet space. (Note that if we had used an orthogonal wavelet transform, this curve would be linear.) Due to the properties of the wavelet transform, we have $\sigma_j = \sigma_I \sigma_j^e$. The standard deviation of the noise at a scale j of the image is equal to the standard deviation of the noise of the image multiplied by the standard deviation of the noise of scale j of the wavelet transform.

Poisson noise

If the noise in the data I is Poisson, the Anscombe transform

$$t(I(x, y)) = 2\sqrt{I(x, y) + \frac{3}{8}} \quad (16.10)$$

acts as if the data arose from a Gaussian white noise model (Anscombe, [6]), with $\sigma = 1$, under the assumption that the mean value of I is large.

For Poisson parameter values under about 20, the Anscombe transformation loses control over the bias. In this case, an alternative approach to variance stabilization is needed. An approach for very small numbers of counts, including frequent zero cases, has been described in [173], [16] and [28], and will be described below. Small numbers of detector counts will most likely be associated with the image background. Note that errors related to small values carry the risk of removing real objects, but not of amplifying noise.

Gaussian and Poisson Noise

The arrival of photons, and their expression by electron counts, on CCD detectors may be modeled by a Poisson distribution. In addition, there is additive Gaussian read-out noise. The Anscombe transformation (eqn. 16.10) has been extended [139] to take this combined noise into

account. As an approximation, consider the signal's value, $I(x, y)$, as a sum of a Gaussian variable, γ , of mean g and standard-deviation σ ; and a Poisson variable, n , of mean m_0 : we set $I(x, y) = \gamma + \alpha n$ where α is the gain.

The generalization of the variance stabilizing Anscombe formula is:

$$t = \frac{2}{\alpha} \sqrt{\alpha I(x, y) + \frac{3}{8}\alpha^2 + \sigma^2 - \alpha g} \quad (16.11)$$

With appropriate values of α , σ and g , this reduces to Anscombe's transformation (eqn. 16.10).

Poisson Noise with Few Photons or Counts

A wavelet coefficient at a given position and at a given scale j is

$$w_j(x, y) = \sum_{k \in K} n_k \psi\left(\frac{x_k - x}{2^j}, \frac{y_k - y}{2^j}\right) \quad (16.12)$$

where K is the support of the wavelet function ψ and n_k is the number of events which contribute to the calculation of $w_j(x, y)$ (i.e. the number of photons included in the support of the dilated wavelet centered at (x, y)).

If a wavelet coefficient $w_j(x, y)$ is due to the noise, it can be considered as a realization of the sum $\sum_{k \in K} n_k$ of independent random variables with the same distribution as that of the wavelet function (n_k being the number of photons or events used for the calculation of $w_j(x, y)$). Then we compare the wavelet coefficient of the data to the values which can be taken by the sum of n independent variables.

The distribution of one event in the wavelet space is directly given by the histogram H_1 of the wavelet ψ . Since independent events are considered, the distribution of the random variable W_n (to be associated with a wavelet coefficient) related to n events is given by n autoconvolutions of H_1

$$H_n = H_1 \otimes H_1 \otimes \dots \otimes H_1 \quad (16.13)$$

For a large number of events, H_n converges to a Gaussian.

In order to facilitate the comparisons, the variable W_n of distribution H_n is reduced by

$$c = \frac{W_n - E(W_n)}{\sigma(W_n)} \quad (16.14)$$

and the cumulative distribution function is

$$F_n(c) = \int_{-\infty}^c H_n(u) du \quad (16.15)$$

From F_n , we derive c_{min} and c_{max} such that $F(c_{min}) = \epsilon$ and $F(c_{max}) = 1 - \epsilon$.

Therefore a reduced wavelet coefficient $w_j^r(x, y)$, calculated from $w_j(x, y)$, and resulting from n photons or counts is significant if:

$$F(w^r) > c_{max} \quad (16.16)$$

or

$$F(w^r) < c_{min} \quad (16.17)$$

and $w_j^r(x, y)$ is obtained by

$$w_j^r(x, y) = \frac{w_j(x, y)}{\sqrt{n} \sigma_{\psi_j}} \quad (16.18)$$

$$= \frac{w_j(x, y)}{\sqrt{n} \sigma_{\psi}} 4^j \quad (16.19)$$

where σ_{ψ} is the standard deviation of the wavelet function, and σ_{ψ_j} is the standard deviation of the dilated wavelet function ($\sigma_{\psi_j} = \sigma_{\psi}/4^j$).

Root mean square map

If, associated to the data $I(x, y)$, we have the root mean square map $R_\sigma(x, y)$, the noise in I is non homogeneous. For each wavelet coefficient $w_j(x, y)$ of I , the exact standard deviation $\sigma_j(x, y)$ have to be calculated from R_σ . A wavelet coefficient $w_j(x, y)$ is obtained by the correlation product between the image I and a function g_j :

$$w_j(x, y) = \sum_k \sum_l I(x, y) g_j(x + k, y + l) \quad (16.20)$$

Then we have

$$\sigma_j^2(x, y) = \sum_k \sum_l R_\sigma^2(x, y) g_j^2(x + k, y + l). \quad (16.21)$$

In the case of the à trous algorithm, the coefficients $g_j(x, y)$ are not known exactly, but they can easily be computed by taking the wavelet transform of a Dirac w^δ . The map σ_j^2 is calculated by correlating the square of the wavelet scale j of w^δ by $R_\sigma^2(x, y)$.

Speckle Noise

Speckle occurs in all types of coherent imagery such as synthetic aperture radar (SAR) imagery, acoustic imagery and laser illuminated imagery. The probability density function (pdf) of the modulus of a homogeneous scene is a Rayleigh distribution:

$$p(\rho) = \frac{\rho}{\sigma^2} e^{-\frac{\rho^2}{2\sigma^2}} \quad M_\rho = \sqrt{\frac{\pi}{2}}\sigma \quad \sigma_\rho = \sqrt{\frac{4-\pi}{2}}\sigma$$

The ratio σ_ρ/M_ρ is a constant of value $\sqrt{\frac{4-\pi}{\pi}}$. This means that the speckle is a multiplicative noise. The pdf of the modulus of a log-transformed speckle noise is:

$$p(\ell) = \frac{e^{2\ell}}{\sigma^2} e^{-\frac{e^{2\ell}}{2\sigma^2}} \quad M_\ell = 0.058 + \log(\sigma) \quad \sigma_\ell = \sqrt{\frac{\pi^2}{24}} = 0.641$$

A better estimator for Rayleigh distribution is the energy ($I = \rho^2$) which is known to have an exponential (*i.e.* Laplace) distribution of parameter $a = 2\sigma^2$

$$p(I) = \frac{1}{a} e^{-\frac{I}{a}}$$

Other Types of Noise

For any type of noise, an analogous study can be carried out in order to find the detection level at each scale and at each position. The types of noise considered so far in this chapter correspond to the general cases in astronomical imagery. We now describe briefly methods which can be used for non-uniform and multiplicative noise.

Additive Non-Uniform Noise

If the noise is additive, but non-uniform, we cannot estimate a standard deviation for the whole image. However, we can often assume that the noise is locally Gaussian, and we can compute a local standard deviation of the noise for each pixel. In this way, we obtain a standard deviation map of the noise, $I_\sigma(x, y)$. A given wavelet coefficient $w_j(x, y)$ is calculated from the pixels of the input image I in the range $I(x - l \dots x + l, y - l \dots y + l)$ where l is dependent on the wavelet transform algorithm, the wavelet function, and the scale j . An upper limit

$u_j(x, y)$ for the noise associated with $w_j(x, y)$ is found by just considering the maximum value in $I_\sigma(x - l \dots x + l, y - l \dots y + l)$ and by multiplying this value by the constant σ_j^e (defined in the subsection “Gaussian noise” at the beginning of this Appendix).

$$u_j(x, y) = \max(I_\sigma(x - l \dots x + l, y - l \dots y + l))\sigma_j^e \quad (16.22)$$

The detection level is not constant over each scale.

Multiplicative Noise

If the noise is multiplicative, the image can be transformed by taking its logarithm. In the resulting image, the noise is additive, and a hypothesis of Gaussian noise can be used in order to find the detection level at each scale.

Multiplicative Non-Uniform Noise

In this case, we take the logarithm of the image, and the resulting image is treated as for additive non-uniform noise above.

Unknown Noise

If the noise does not follow any known distribution, we can consider as significant only wavelet coefficients which are greater than their local standard deviation multiplied by a constant: $w_j(x, y)$ is significant if

$$|w_j(x, y)| > k\sigma(w_j(x - l \dots x + l, y - l \dots y + l)) \quad (16.23)$$

Appendix C: Derivative Needed for the Minimization

The error function

The error function is written $\text{erf}(x)$ and the complementary error function $\text{erfc}(x)$. Their definitions are:

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \quad (16.24)$$

$$\text{erfc}(x) = 1 - \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt \quad (16.25)$$

These functions have the following limits and symmetries:

$\text{erf}(0) = 0$ $\text{erfc}(0) = 1$ $\text{erf}(-x) = \text{erf}(x)$	$\text{erf}(\infty) = 1$ $\text{erfc}(\infty) = 0$ $\text{erfc}(-x) = 2 - \text{erfc}(x)$
---	---

(16.26)

N1-MSE

Now we compute the contribution of the wavelet coefficient x to the noise information:

$$h_n(x) = \frac{x^2}{2\sigma^2} \text{erfc}\left(\frac{x}{\sqrt{2}\sigma}\right) \quad (16.27)$$

$$\frac{dh_n(x)}{dx} = \frac{x}{\sigma^2} \text{erfc}\left(\frac{x}{\sqrt{2}\sigma}\right) + \frac{x^2}{2\sigma^2} \frac{\partial \text{erfc}\left(\frac{x}{\sqrt{2}\sigma}\right)}{\partial x} \quad (16.28)$$

We derive the function erfc :

$$\frac{\partial \text{erfc}(x)}{\partial x} = -\frac{2}{\sqrt{\pi}} e^{-x^2} \quad (16.29)$$

then

$$\frac{dh_n(x)}{dx} = \frac{x}{\sigma^2} \text{erfc}\left(\frac{x}{\sqrt{2}\sigma}\right) - \frac{x^2}{\sqrt{2\pi}\sigma^3} e^{-\frac{x^2}{2\sigma^2}} \quad (16.30)$$

In order to minimize the functional 13.10, we may want to calculate the derivative of $h_s(x-y)$, $h_s(x-y)$ measuring the amount of information contained in the residual (y being the data).

$$h_s(x-y) = \frac{(y-x)^2}{2\sigma^2} \text{erf}\left(\frac{y-x}{\sqrt{2}\sigma}\right) \quad (16.31)$$

Denoting $z = y - x$, we have

$$h_s(z) = \frac{z^2}{2\sigma^2} \operatorname{erf}\left(\frac{z}{\sqrt{2}\sigma}\right) \quad (16.32)$$

$$\frac{dh_s(z)}{dz} = \frac{z}{\sigma^2} \operatorname{erf}\left(\frac{z}{\sqrt{2}\sigma}\right) + \frac{z^2}{2\sigma^2} \frac{\partial \operatorname{erf}\left(\frac{z}{\sqrt{2}\sigma}\right)}{\partial z} \quad (16.33)$$

$$= \frac{z}{\sigma^2} \operatorname{erf}\left(\frac{z}{\sqrt{2}\sigma}\right) + \frac{z^2}{\sqrt{2\pi}\sigma^3} e^{-\frac{z^2}{2\sigma^2}} \quad (16.34)$$

and

$$\frac{dh_s(x-y)}{dx} = -\frac{dh_s(z)}{dz} \quad (16.35)$$

N2-MSE

We compute the contribution of the wavelet coefficient x to the noise information:

$$h_n(x) = \frac{1}{\sigma^2} \int_0^x t \operatorname{erfc}\left(\frac{x-t}{\sqrt{2}\sigma}\right) dt \quad (16.36)$$

$$\frac{dh_n(x)}{dx} = h_n(x+dx) - h_n(x) \quad (16.37)$$

$$= \frac{1}{\sigma^2} \int_0^{x+dx} \operatorname{erfc}\left(\frac{x+dx-t}{\sqrt{2}\sigma}\right) dt - \frac{1}{\sigma^2} \int_0^x \operatorname{erfc}\left(\frac{x-t}{\sqrt{2}\sigma}\right) dt \quad (16.38)$$

$$\frac{dh_n(x)}{dx} = \frac{x}{\sigma^2} \operatorname{erfc}\left(\frac{x-x}{\sqrt{2}\sigma}\right) + \frac{1}{\sigma^2} \int_0^x \frac{\partial t \operatorname{erfc}\left(\frac{x-t}{\sqrt{2}\sigma}\right)}{\partial x} dt \quad (16.39)$$

Now, because $\operatorname{erfc}(0) = 1$ we have:

$$\frac{dH_n(x)}{dx} = \frac{x}{\sigma^2} + \frac{1}{\sigma^2} \int_0^x \frac{\partial t \operatorname{erfc}\left(\frac{x-t}{\sqrt{2}\sigma}\right)}{\partial x} dt \quad (16.40)$$

We derive the function erfc :

$$\frac{\partial \operatorname{erfc}(x)}{\partial x} = -\frac{2}{\sqrt{\pi}} e^{-x^2} \quad (16.41)$$

$$\frac{\partial \operatorname{erfc}\left(\frac{x-t}{\sqrt{2}\sigma}\right)}{\partial x} = -\frac{2}{\sqrt{\pi}} \frac{1}{\sqrt{2}\sigma} e^{-\frac{(x-t)^2}{2\sigma^2}} = -\sqrt{\frac{2}{\pi}} \frac{1}{\sigma} e^{-\frac{(x-t)^2}{2\sigma^2}} \quad (16.42)$$

Now we deduce for the derivative of h_n :

$$\frac{dh_n(x)}{dx} = \frac{x}{\sigma^2} + \frac{1}{\sigma^2} \int_0^x -\sqrt{\frac{2}{\pi}} \frac{1}{\sigma} t e^{-\frac{(x-t)^2}{2\sigma^2}} dt \quad (16.43)$$

$$\frac{dh_n(x)}{dx} = \frac{x}{\sigma^2} + \frac{1}{\sigma^3} \sqrt{\frac{2}{\pi}} \int_0^x t e^{-\frac{(x-t)^2}{2\sigma^2}} dt \quad (16.44)$$

We create the variable J

$$J = \int_0^x t e^{-\frac{(x-t)^2}{2\sigma^2}} dt \quad (16.45)$$

We create the variable u

$$\begin{aligned} u &= \frac{t-x}{\sqrt{2}\sigma} & t &= x + u \sqrt{2}\sigma \\ && dt &= \sqrt{2}\sigma du \\ t = 0 \Rightarrow u &= \frac{-x}{\sqrt{2}\sigma} & t = x \Rightarrow u &= 0 \end{aligned} \quad (16.46)$$

The variable J can be written with u

$$J = \int_{\frac{-x}{\sqrt{2}\sigma}}^0 (x + u \sqrt{2}\sigma) e^{-u^2} \sqrt{2}\sigma du \quad (16.47)$$

$$J = \sqrt{2}\sigma x \int_{\frac{-x}{\sqrt{2}\sigma}}^0 e^{-u^2} du + 2\sigma^2 \int_{\frac{-x}{\sqrt{2}\sigma}}^0 u e^{-u^2} du \quad (16.48)$$

The first part of J can be rewritten as:

$$J_0 = \sqrt{2}\sigma x \int_0^{\frac{x}{\sqrt{2}\sigma}} e^{-u^2} du \quad (16.49)$$

J_0 can be expressed with the error function.

$$J_0 = \sqrt{2}\sigma \frac{\sqrt{\pi}}{2} x \operatorname{erf}\left(\frac{x}{\sqrt{2}\sigma}\right) \quad (16.50)$$

Now the second part of J is obvious

$$J_1 = 2\sigma^2 \int_{\frac{-x}{\sqrt{2}\sigma}}^0 u e^{-u^2} du \quad (16.51)$$

or

$$\frac{de^{-u^2}}{du} = -2ue^{-u^2} \quad (16.52)$$

We replace

$$J_1 = -\sigma^2 \int_{\frac{-x}{\sqrt{2}\sigma}}^0 d(e^{-u^2}) \quad (16.53)$$

$$J_1 = \sigma^2 [e^{-\frac{x^2}{2\sigma^2}} - 1] \quad (16.54)$$

Now we can write J

$$J = J_0 + J_1 = \sigma \sqrt{\frac{\pi}{2}} x \operatorname{erf}\left(\frac{x}{\sqrt{2}\sigma}\right) + \sigma^2 [e^{-\frac{x^2}{2\sigma^2}} - 1] \quad (16.55)$$

Now we can write the derivative of h_n

$$\frac{dh_n(x)}{dx} = \frac{x}{\sigma^2} - \frac{1}{\sigma^3} \sqrt{\frac{2}{\pi}} J \quad (16.56)$$

$$= \frac{x}{\sigma^2} - \frac{x}{\sigma^2} x \operatorname{erf}\left(\frac{x}{\sqrt{2}\sigma}\right) + \frac{1}{\sigma} \sqrt{\frac{2}{\pi}} [1 - e^{-\frac{x^2}{2\sigma^2}}] \quad (16.57)$$

$$= \frac{x}{\sigma^2} \operatorname{erfc}\left(\frac{x}{\sqrt{2}\sigma}\right) + \frac{1}{\sigma} \sqrt{\frac{2}{\pi}} [1 - e^{-\frac{x^2}{2\sigma^2}}] \quad (16.58)$$

In order to minimize the functional 13.10, we may want to calculate the derivative of $h_s(x | y)$, $h_s(x | y)$ measuring the amount of information contained in the residual (y being the data).

$$h_s(x | y) = \frac{1}{\sigma^2} \int_0^{y-x} t \operatorname{erf}\left(\frac{y-x-t}{\sqrt{2}\sigma}\right) dt \quad (16.59)$$

Denoting $z = y - x$, we have

$$h_s(z) = \frac{1}{\sigma^2} \int_0^z t \operatorname{erf}\left(\frac{z-t}{\sqrt{2}\sigma}\right) dt \quad (16.60)$$

$$= \frac{1}{\sigma^2} \int_0^z t dt - \frac{1}{\sigma^2} \int_0^z t \operatorname{erfc}\left(\frac{z-t}{\sqrt{2}\sigma}\right) dt \quad (16.61)$$

and

$$\frac{dh_s(x)}{dx} = \frac{\partial h_s(z)}{\partial z} \frac{\partial z}{\partial x} \quad (16.62)$$

$$\frac{\partial h_s(z)}{\partial z} = \frac{z}{\sigma^2} - \frac{\partial h_n(z)}{\partial z} \quad (16.63)$$

then

$$\frac{dh_s(x)}{dx} = -\frac{y-x}{\sigma^2} + \frac{y-x}{\sigma^2} \operatorname{erfc}\left(\frac{y-x}{\sqrt{2}\sigma}\right) + \sqrt{\frac{2}{\pi}} \frac{1}{\sigma} [1 - e^{-\frac{(y-x)^2}{2\sigma^2}}] \quad (16.64)$$

$$= -\frac{y-x}{\sigma^2} \operatorname{erf}\left(\frac{y-x}{\sqrt{2}\sigma}\right) + \sqrt{\frac{2}{\pi}} \frac{1}{\sigma} [1 - e^{-\frac{(y-x)^2}{2\sigma^2}}] \quad (16.65)$$

Appendix D: Generalization of Derivative Needed for Minimization

The contribution of the wavelet coefficient x to the noise and signal information in the general case is

$$\begin{aligned} h_n(x) &= \int_0^{|x|} p_n(u|x) \left(\frac{\partial h(x)}{\partial x} \right)_{x=u} du \\ h_s(x) &= \int_0^{|x|} p_s(u|x) \left(\frac{\partial h(x)}{\partial x} \right)_{x=u} du \end{aligned} \quad (16.66)$$

Assuming $h(x) = \frac{1}{2}x^2$, we have

$$\begin{aligned} h_n(x) &= \int_0^{|x|} p_n(x-u) u du \\ h_s(x) &= \int_0^{|x|} p_s(x-u) u du \end{aligned} \quad (16.67)$$

$$\frac{dh_s(x)}{dx} = \int_0^x \frac{\partial P_s(x-u)}{\partial x} \Big|_{x=u} u du + \frac{1}{dx} \int_x^{x+dx} P_s(x-u) u du \quad (16.68)$$

As $P_s(0) = 0$, the second term tends to zero.

Denoting $\frac{\partial P_s(x-u)}{\partial x} = -\frac{\partial P_s(x-u)}{\partial u}$, we have

$$\frac{dh_s(x)}{dx} = - \int_0^x \frac{\partial P_s(x-u)}{\partial u} u du \quad (16.69)$$

$$= -([uP_s(x-u)]_0^x - \int_0^w P_s(x-u) du) \quad (16.70)$$

$$= \int_0^x P_s(x-u) du \quad (16.71)$$

$$= \int_0^x P_s(u) du \quad (16.72)$$

and from $h_n = h - h_s$ we get

$$\frac{dh_n(x)}{dx} = x - \int_0^x P_s(u) du \quad (16.73)$$

and

$$\frac{dh_s(y-x)}{dx} = - \int_0^{y-x} P_s(u) du \quad (16.74)$$

It is easy to verify that replacing $P_s(x) = \text{erf}(x)$, and $P_n(x) = \text{erfc}(x)$, (case of Gaussian noise) we find the same equation as in Appendix B.

Part III

MR/3 : Multichannel Data

Chapter 17

MR/3 Data Processing Tools

17.1 Introduction

MR/3 deals with the analysis of multi-channel data or 3D data. Multi-channel data can be either 1D or 2D multi-channel:

- 1D multi-channel: a 1D signal is observed at several wavelength or at different times. The result is an image, $I(*, N_v)$, where N_v is the number of vectors.
- 2D multi-channel: a 2D signal (image) is observed at several wavelengths or at different times. The result is a cube, $C(*, *, N_i)$, where N_i is the number of images. A special case of multi-channel images is color images. In this case, we have three channels. Several color coordinate systems exist. The most widely-used is the RGB system. Each pixel is identified by three values R, G, and B corresponding to the three colors red, green, and blue.

Programs which deal with a 1D multi-channel image (i.e., a 2D data set) read/write all image formats described in MR/1, raw format (.d), FITS format (.fits), GIF format (.gif), JPEG format (.jpg) and PGM format (.pgm).

Programs relative to a 3D data set, cubes or multi-channel images, read/write FITS format (.fits), GIF format (.gif), TIFF format (.tiff), and JPEG format (.jpg). As for single channel images, and for the same reasons (i.e, to avail by default of 32-bit pixel data types), it is recommended to use FITS format. In the FITS format, the two first dimensions must be the spatial dimensions, and the third one the time or the wavelength.

17.2 3D-Image Manipulation Tools

A number of programs have been developed for basic 3D-image manipulation.

17.2.1 Image conversion: im3d_convert

The program *im3d_convert* converts an image from one data format to another.

USAGE: **im3d_convert options file_name_in file_name_out**

where options are:

- [-b]
Normalize the data between 0 and 255.
- [-x]
Flip x-axis.

- [-y]
Flip y-axis.
- [-r]
90 degrees rotation.

If the “-b” option is set, data are scaled in order to have a minimum value equal to 0, and a maximum value equal to 255.

Examples:

- im3d_convert image.jpeg image.fits
Converts an image from JPEG format to FITS format.
- im3d_convert -b image.fits image.gif
Converts an image from raw format to GIF format with normalization of scale.

17.2.2 Image information: im3d_info

im3d_info gives information about a cube or a multi-channel image. If the “-a” option is used, it returns also information about each individual frame:

- the number of rows and columns
- the minimum, the maximum, the average
- the average and the standard deviation
- the flux: $\sum_k I_k$
- the energy: $\sum_k I_k^2$

The command line is:

USAGE: im3d_info file_name.in

where options are:

- [-a]
Print also information about all individual frames.

Examples:

- im3d_info image.tiff
Gives information about the image.
- im3d_info -a image.tiff
Ditto, but print also information about all individual frames.

17.2.3 Extract a subcube: im3d_get

im3d_get extracts a part of a cube:

USAGE: im3d_get options cube_in cube_out

where options are:

- [-x Xstart:XEnd]
Area to extract in the first dimension.
- [-y Ystart:YEnd]
Area to extract in the second dimension.
- [-z Zstart:ZEnd]
Area to extract in the third dimension.

17.2.4 Insert a subcube: im3d_put

im3d_put inserts an image into a larger one:

USAGE: im3d_put options cube_in cube_inout

where options are:

- **[-x Xstart]**
x-coordinate of the cube to insert.
- **[-y Ystart]**
y-coordinate of the cube to insert.
- **[-Z Zstart]**
z-coordinate of the cube to insert.

17.2.5 Operation on two cubes: im3d_op

im3d_op calculates an operation on two cubes.

USAGE: im3d_put cube_in1 [+ - */] cube_in2 cube_out

17.2.6 Image simulation: im3d_simu

im3d_simu adds noise (Poisson, Gaussian, or both kinds of noise) to the input image, and/or convolves it beforehand with a point spread function (PSF) which can either be read from a file, or created by the program. If the PSF is simulated, it is a Gaussian and a parameter fixes the full-width at half-maximum (FWHM).

USAGE: im3d_simu options image_in image_out

where options are:

- **[-p]**
Add Poisson Noise. Default is not to do this.
- **[-g sigma]**
Add Gaussian noise. *sigma* is the standard deviation of the added noise. Default is not to do this.
- **[-c sigma]**
Add Poisson noise and Gaussian noise. *sigma* is standard deviation of the Gaussian noise. Default is not to do this.
- **[-r psf_image]**
Convolve the input image with a PSF. The PSF is read from a file of name *psf_image*. Default is not to do this.
- **[-f FWHM]**
Convolve the input image with a PSF. The PSF is a Gaussian which has a full-width at half-maximum equal to *FWHM*. Default is not to do this.
- **[-w PsffFileName]**
Write the calculated PSF to disk. Valid only if “-f” option is set. Default is no.
- **[-I InitRandomVal]**
Value used for random value generator initialization.
Default is 100.

The r and f options cannot be used together.

The g, p, and c options cannot be used together.

Examples:

- `im3d_simu -g 200 image.fits image_g200.fits`
Add Gaussian noise to an image.
- `im3d_simu -f 3. -g 200 image.fits image_g200.fits`
Convolve the image with a Gaussian (FWHM=3), and add Gaussian noise of standard deviation equal to 200.
- `im3d_simu -r PSF.fits -c 20 image.fits image_p.fits`
Convolve the image with another one (PSF.fits), and add Poisson noise and read-out noise (Gaussian noise).

17.3 Multi-Temporal Images

17.3.1 Introduction

An observation is often repeated several times in the same configuration (same wavelength, same field, etc.). Hence, we have a 3D data set, where each frame contains the same field of view. However, the data can generally not be coadded because a small offset exists between the frames, due to the observation conditions. The problem is then to find an image $I(x, y)$ from the data set $D(x, y, k)$, $k = 1, \dots, N_f$, where N_f is the number of frames. In the case where the point spread function of the instrument is undersampled (i.e., the sampling theorem is not respected), these offsets may be an advantage because we may recover information at a higher resolution than the resolution of a single frame.

17.3.2 Image Coaddition: `im3d_coadd`

Program *im3d_coadd* coadds a set of images, taking into account the offsets between the successive frames. The offsets are determined by cross-correlation in a given surface area. The “-r” option allows the user to coadd the data on a finer grid. The resampled and registered cube can be saved with the “-W” option. If the offsets are already known, *im3d_coadd* can use this information when the option “-o” is selected. Denoting $D_r(x, y, k)$ as the registered cube, the output image I is a simple average of the frames:

$$I(x, y) = \frac{1}{N_f} \sum_{i=1}^{N_f} D_r(x, y, i)$$

If the “-M” option is used, the median image is calculated instead of the mean image:

$$I(x, y) = \text{median}(D_r(x, y, i)_{i=1, \dots, N_f})$$

When the “-R” option is set, the root mean square is calculated by:

$$R(x, y) = \sqrt{\frac{1}{N_f} \sum_{i=1}^{N_f} (D_r(x, y, i) - I(x, y))^2}$$

where I is either the mean or the median image. The command line is:

USAGE: im3d_coadd options cube_in image_out

where options are:

- **[-f type_of_interpolation]**

1. Sinc interpolation: $\phi(x) = \text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$.

2. Lanczos interpolation: $\phi(x) = \begin{cases} \text{sinc}(x)\text{sinc}(\frac{x}{2}) & \text{if } |x| < 2 \\ 0 & \text{otherwise} \end{cases}$
3. Hamming interpolation: $\phi(x) = \begin{cases} \alpha + (1 - \alpha) \cos(\frac{\sin(2\pi x)}{N}) & \text{if } |x| < \frac{N-1}{2} \\ 0 & \text{otherwise} \end{cases}$ where N is the number of samples in the windowing function, and $\alpha = 0.54$.
4. Hann: same as Hamming interpolation but $\alpha = 0.5$.
5. Hyperbolic tangent: this is defined in Fourier space by

$$H_k(\nu) = \left(\frac{\tanh(k(\nu + 0.5)) + 1}{2} \right) \left(\frac{\tanh(k(-\nu + 0.5)) + 1}{2} \right)$$

Default is 5.

- **[-r ZoomFactor]**
Rebin the reconstructed image.
- **[-d MaxDist]**
Maximum offset between two frames. Default is 4 pixels.
- **[-a Surface]**
Surface area used for the cross-correlation calculation. Default is 10 (i.e., 10×10 pixels).
- **[-x XPos]**
X position for the search area. Default is image center.
- **[-y YPos]**
Y position for the search area. Default is image center.
- **[-m]**
Subtract from each frame its mean value before calculating the offsets.
- **[-o InputOffsetFileName]**
Read the offsets from a FITS file. If this option is set, the offsets are not calculated. The offset file contains an array $(2, N_f)$.

```
Array(0,k) = x offset from frame k to first frame.
Array(1,k) = y offset from frame k to first frame.
```

- **[-w OutputOffsetFileName]**
Store in a FITS file the calculated offsets. The offset file contains an array $(2, N_f)$.

```
Array(0,k) = x offset from frame k to first frame.
Array(1,k) = y offset from frame k to first frame.
```

- **[-W RegisterCubeFileName]**
Write to disk the registered cube.
- **[-M]**
Take the median instead of the mean when coadding the frame.
- **[-R OutputRMSFileName]**
Write to disk the Root Mean Square Map.
- **[-N]**
No offset.

Examples:

- `im3d_coadd cube.fits ima.fits`
Coadd the cube, using all default options.
- `im3d_coadd -r 4 cube.fits ima.fits`
Ditto, but rebin by 4 the frames before the coadding. The output image is therefore 4 times larger in both directions.

17.3.3 Deconvolution: im3d_deconv**Introduction**

Some observations are made with an undersampled PSF. When the observation is repeated several times with a small shift between two measurements, we can reconstruct a deconvolved image on a smaller grid. We denote $D(i, j, k)$ the k th observation ($k = 1..n$), $\Delta_{i,k}$, $\Delta_{j,k}$ the shift in both directions relative to the first frame, \mathcal{L}_\uparrow the operator which coadds all the frames on a smaller grid, and $\mathcal{L}_\downarrow^{-1}$ the operator which estimates D from $\mathcal{L}_\uparrow D$ using shifting and averaging operations. The $\Delta_{i,k}$, $\Delta_{j,k}$ shifts are generally derived from the observations using correlation methods, or PSF fitting (e.g., in the case of a star is an astronomical field), but can also be the camera jitter information in other cases (e.g., space-borne detectors). Note also that $\mathcal{L}_\downarrow^{-1} \mathcal{L}_\uparrow D \neq D$. The point spread function P can generally be defined on a finer grid using a set of observations of a star, or using optical modeling of the instrument. The Landweber deconvolution iteration becomes:

$$O^{n+1} = O^n + \alpha P^* [\mathcal{L}_\uparrow(D - \mathcal{L}_\downarrow^{-1}(P * O^n))] \quad (17.1)$$

and the positivity and spatial constraints can also be used:

$$O^{n+1} = \mathcal{P}_{C_s}^+ [O^n + \alpha P^* [\mathcal{L}_\uparrow(D - \mathcal{L}_\downarrow^{-1}(P * O^n))]] \quad (17.2)$$

The coaddition operator \mathcal{L}_\uparrow can be implemented in different ways. All frames can first be interpolated to the finer grid size, shifted using an interpolation function, and then coadded. In [112], another method has been proposed which eliminates aliasing effects.

If noise is present in the data, it may be amplified during the iterations, and wavelet regularization then becomes helpful.

Deconvolution Program

Program *im3d_deconv* coadds and deconvolves at the same time a set of frames, taking into account the offsets between the successive frames. The offsets are determined by cross-correlation. Offset can be either calculated or given using the “-o option”. For example, offsets can be calculated using the *im3d_coadd* program. If the “-W” option is set, the wavelet transform is used for the regularization. The command line is:

USAGE: im3d_deconv option cube_in cube_out]

where options are:

- **[-d type_of_deconvolution]**

1. Van-Citter iteration.
2. Landweber iteration.
3. Lucy iteration.
4. MAP iteration.

- **[-f type_of_interpolation]**

1. Sinc.
2. Lanczos.
3. Hamming.
4. Hann.
5. Tanh.

Default is 5.

- **[-r ZoomFactor]**

Rebin the reconstructed image.

- **[-o InputOffsetFileName]**

Read the offsets from a FITS file. If this option is set, the offsets are not calculated. The offset file contains an array $(2, N_f)$.

```
Array(0,k) = x offset from frame k to first frame.
Array(1,k) = y offset from frame k to first frame.
```

- **[-i MaxIter]**

Number of iterations. Default is 50.

- **[-p]**

Suppress the positivity constraint.

- **[-N]**

No offset.

- **[-W]**

Regularization by the wavelet transform.

- **[-t type_of_multiresolution_transform]**

1. linear wavelet transform: à trous algorithm
2. B-spline wavelet transform: à trous algorithm
3. wavelet transform in Fourier space
4. morphological median transform
5. morphological minmax transform
6. pyramidal linear wavelet transform
7. pyramidal B-spline wavelet transform
8. pyramidal wavelet transform in Fourier space: wavelet = between two resolutions
9. pyramidal wavelet transform in Fourier space: wavelet = difference between the square of two resolutions
10. pyramidal median transform
11. pyramidal Laplacian
12. morphological pyramidal minmax transform
13. decomposition on scaling function
14. (bi-) orthogonal wavelet transform.
Antonini 7/9 filters [7] are used by default, with an L_1 normalization. The filters can be changed using the “-T” option, and an L_2 normalization is obtained by “-L” option.
15. Feauveau wavelet transform

16. Feauveau wavelet transform without undersampling
17. G transform (non-redundant morphological min-max algorithm)
18. Haar wavelet transform (L2 normalization).
19. Half-pyramidal wavelet transform (HPWT)
20. Mixed HPWT and Median method
21. dyadic wavelet transform
22. Mixed WT and PMT method (WT-PMT)
23. Undecimated Haar transform: à trous algorithm
24. Undecimated (bi-) orthogonal wavelet transform.
Antonini 7/9 filters [7] are used by default, with an L_1 normalization. The filters can be changed using the “-T” option, and an L_2 normalization is obtained by “-L” option.

Default is 2.

- **`[-n number_of_scales]`**

Number of scales used in the multiresolution transform. Default is 4.

- **`[-s NSigma]`**

The detection level at each scale is determined by the product of the standard deviation of the noise by the $NSigma$. $NSigma$ fixes the confidence interval we want. By default, $NSigma$ is equal to 3.

Examples:

- `im3d_deconv cube.fits ima.fits`
Coadd the cube, using all default options.
- `im3d_deconv -r 4 cube.fits ima.fits`
Ditto, but rebin by 4 the frames before the coadding. The output image is therefore 4 times larger in both directions.
- `im3d_deconv -r 4 -W cube.fits ima.fits`
Ditto, but use also the wavelet transform for the regularization.

17.4 Color Images

17.4.1 Introduction

Color images are a special case of multi-channel images. The information included in the three R,G,B channels is in general very redundant, and a good filtering or compression method should use this redundancy. A Karhunen-Loève transform could be used to do this job, as discussed in the next chapter, but a more efficient approach is to use a pre-defined transformation which is known to be near optimal for decorrelating the information. The YUV transformation consists of performing the following operations:

$$\begin{aligned} Y &= (0.257 * R) + (0.504 * G) + (0.098 * B) + 16 \\ C_r &= (0.439 * R) - (0.368 * G) - (0.071 * B) + 128 \\ C_b &= -(0.148 * R) - (0.291 * G) + (0.439 * B) + 128 \end{aligned} \quad (17.3)$$

where Y is called the luminance map, and C_r and C_b the chromaticity maps. Each pixel of the original map can therefore be decomposed, independently of the others. The transformation does not require much computation time, and no extra memory.

17.4.2 Color Image Filtering: col_filter

Program *col_filter* filters a color image by the undecimated bi-orthogonal wavelet transforms (7/9 filters). The data are first transformed into the YUV coordinate system. Each L, C_r, C_b map is wavelet transformed, thresholded, and reconstructed. The inverse transformation YUV to RGB furnishes the filtered image.

The undecimated wavelet transform is very redundant. The number of pixels after transformation is $3(N - 1) + 1$ where N is the number of pixels in the input channel. For this reason, it requires much more computation time than the decimated wavelet transform, but the quality is also much better. The “-u” option allows the user to keep the first scales undecimated (highest frequency bands) while decimating the others. So this option defines a “partially” decimated wavelet transform. If “-u 0” is set, a decimated wavelet transform is selected, and if “-u NumberOfScales” is set, an undecimated wavelet transform is selected. Experiments have shown that a good trade-off between quality and redundancy is obtained with only one undecimated scale (“-u 1”).

USAGE: col_filter option image_in image_out

where options are:

- **[-n number_of_scales]**
Number of scales used in the multiresolution transform. Default is 6.
- **[-u number_of_undecimated_scales]**
Number of undecimated scales used in the wavelet transform. Default is all.
- **[-s NSigma]**
Thresholding at $NSigma * SigmaNoise$. Default is 3.
- **[-S]**
Use soft thresholding instead of hard thresholding. Default is False.
- **[-g SigmaNoise]**
 $SigmaNoise = \text{Gaussian noise standard deviation}$. Default is automatically estimated.
- **[-C]**
Correlated noise.

Examples:

- `col_filter image.fits output.fits`
Filter the image with all default options.
- `col_filter -u 1 -n5 image.tiff output.tiff`
Filter the image using 5 scales, and only one undecimated scale.
- `col_filter -u 1 -n5 -s5 image.tiff output.tiff`
Ditto, but filter more structures.
- `col_filter -C image.tiff output.tiff`
Consider correlated noise.

17.4.3 Color Image Compression: col_comp

Program *col_comp* compresses an image by the bi-orthogonal wavelet wavelet transform (7/9 filters). The quantization is not uniform, and uses the quantization levels given in [211], which depend on the orientation (i.e., horizontal, diagonal, or vertical band) and on the scale. These levels have been derived from properties of the human visual system.

The output file has the “.MRC” suffix. If the output file is not specified, the output file will be the input file name with the extension “.MRC”. When the “-B” option is set, the image is first separated into independent blocks, and each block is wavelet compressed. This allows the user to have fast access to a part of the image during the decompression.

USAGE: col_comp option image_in [compressed_file_out]

where options are:

- **[-g QuantifParam]**
Quantization parameter of noise level. Default is 7.
- **[-p ReadPseudo]**
Read the input data as a pseudo-image (for JPEG format only).
- **[-n number_of_scales]**
Number of scales used in the multiresolution transform. Default is 6.
- **[-B BlockSize]**
Compress by block. BlockSize = size of each block. Default is not to use blocks.

Examples:

- col_comp image.tiff
Compress the image, and store the result in “image.tiff.MRC”
- col_comp -g 10 image.jpg test.MRC
Compress the image, and store the result “test.MRC”. The quantization parameter has been increased in order to increase the compression ratio.
- col_comp -B 128 lena.jpg lena.MRC
The 512×512 input image is first decomposed into 4 blocks, and each of them is wavelet compressed.

17.4.4 Color Image Decompression: col_decomp

Program *col_decomp* decompresses a file compressed with *col_comp*. We are not always interested in decompressing the image at full resolution. The option “-r” allows an image at lower resolution to be extracted from the compressed file, and produces in this way a smaller image than the original. Note that for lower resolution decompression, only the necessary part of the file is read and decompressed (and so the decompression is particularly efficient).

USAGE: col_decomp option CompressedFileName OutputFileName

where options are:

- **[-B BlockNbr]**
Decompress only one block. *BlockNbr* is the block number to decompress. Default is to take the entire image.
- **[-r resolution]**
resolution must be ≥ 0 and $<$ number of scales of the transform. By default, the image is reconstructed at full resolution with its noise if this exists in the compressed file.

Examples:

- col_comp image.tiff
Compress the image, and store the result “image.tiff.MRC”
- col_decomp image.tiff.MRC decima.tiff
Decompress the file, and store the result “decima.tiff”.
- col_decomp -r 1 image.tiff.MRC decima.tiff
Decompress the file at a lower resolution. The decompressed image has size 128×128 when the original one had a size of 256×256 .

- `col_decomp -r 3 image.tif MRC decima.tif`
Same as before, but the decompressed image has size 32×32 .
- `col_comp -B 128 lena.jpg lena.MRC`
The 512×512 input image is first decomposed into 4 blocks, and each of them is wavelet compressed.
- `col_decomp -B 2 -r 2 lena.MRC decima.tif`
Decompress only the second block, and at a lower resolution.

17.4.5 Color Image Enhancement: col_contrast

Program *col_contrast* improves the contrast in a color image.

Color saturation

A RGB color image can be saturated by the following procedure:

$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \frac{C_1}{C_1 - C_2} \begin{pmatrix} R - C_2 \\ G - C_2 \\ B - C_2 \end{pmatrix} \quad (17.4)$$

with $C_1 = \max\{R, G, B\}$ and $C_2 = \min\{R, G, B\}$.

As objects can exhibit variations in color saturation with little or no corresponding in luminance variation, several multiscale methods have been proposed in the past for color image enhancement [205].

Color saturation with sigma clipping

The sigma clipping method consists in estimating the mean and the data set standard deviation, but without considering the outlier values.

1. Set $i = 1$, and L^i = list of all pixels.
2. Calculate m^i and σ^i , the mean and the standard deviation of L^i .
3. while $i \leq k$ do
 - Remove from L^i all pixels such that $|L^i(l) - m^i| > 3\sigma^i$. We get L^{i+1} .
 - Calculate m^{i+1} and σ^{i+1} , the mean and the standard deviation of L^{i+1} .
4. $i = i + 1$ and goto 3
5. Set $m_k = m^i$ and $\sigma_k = \sigma^i$.

We take generally $k = 3$.

The saturation is then performed by:

$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \frac{C_1}{C_1 - C_2} \begin{pmatrix} S(R) - C_2 \\ S(G) - C_2 \\ S(B) - C_2 \end{pmatrix} \quad (17.5)$$

with $C_1 = m_k - K\sigma_k$, $C_2 = m_k + K\sigma_k$, and S is the function defined by:

$$S(x) = \begin{cases} x & \text{if } x \in [C_1, C_2] \\ C_1 & \text{if } x < C_1 \\ C_2 & \text{if } x > C_1 \end{cases} \quad (17.6)$$

Histogram equalization

The luminance map L can be modified by an histogram equalization. Let \tilde{L} represent the enhanced version of $L(x, y)$. We modify the RGB components by:

$$\begin{aligned} \tilde{R}(x, y) &= K(x, y)R(x, y) \\ \tilde{G}(x, y) &= K(x, y)G(x, y) \\ \tilde{B}(x, y) &= K(x, y)B(x, y) \end{aligned} \quad (17.7)$$

where $K(x, y) = \frac{\tilde{L}(x, y)}{L(x, y)}$. Thus no inverse color transformation is required.

The Retinex

The retinex concept has been introduced by Land [109] as a model for human color constancy. The single scale retinex (SSR) method [101] consists in applying the following transform to each band i of the color image:

$$R_i(x, y) = \log(I_i(x, y)) - \log(F(x, y) * I_i(x, y)) \quad (17.8)$$

Where $R_i(x, y)$ is the retinex output, $I_i(x, y)$ is the image distribution in the i th spectral band, and F a Gaussian function. A gain/offset is applied to the retinex ouput which clips the highest and lowest signal excursions. This can be done by a k-sigma clipping. The retinex method is efficient for dynamic range compression, but does not provide good tonal rendition [157].

Multiscale Retinex

The multiscale Retinex (MSR) combines several SSR outputs to produce a single output image which has both good dynamic range compression and color constancy, and good tonal rendition [100]. The MSR can be defined by:

$$R_{MSR_i} = \sum_{j=1}^N w_j R_{i,j} \quad (17.9)$$

with

$$R_{i,j}(x, y) = \log(I_i(x, y)) - \log(F_j(x, y) * I_i(x, y)) \quad (17.10)$$

N is the number of scales, $R_{i,j}$ is the i th spectral component of the MSR output, and w_j is the weight associated with the scale j . The Gaussian F_j is given by:

$$F_j(x, y) = K \exp -\frac{r^2}{c_j^2} \quad (17.11)$$

c_j defines the width of the Gaussian. In [100], three scales were recommended with c_j values equal respectively to 15,80,250, and all weights w_j fixed to $\frac{1}{N}$.

Multiscale Edge Enhancement

Velde has proposed the following algorithm [208]:

1. The RGB image is transformed in Luv image.
2. The L component is mapped nonlinearly according to:

$$L(i) \rightarrow L(i)^{1-q} 100^q \quad (17.12)$$

3. The L,u,v components are each decomposed into a multiscale gradient pyramid by using the diadic wavelet transform (two directions per scale). The gradient at the scale j , the pixel position i and for the component C ($X \in \{L, u, v\}$) is calculated by: $G_j^C(i) = \sqrt{(w_j^{(h)}(i))^2 + (w_j^{(v)}(i))^2}$ where $w_j^{(h)}$ and $w_j^{(v)}$ are the wavelet coefficients in both directions at pixel position i . The norm of the color gradient at resolution level j and pixel i is therefore:

$$\Gamma_j(i) = \sqrt{\| G_j^L(i) \|^2 + \| G_j^u(i) \|^2 + \| G_j^v(i) \|^2} \quad (17.13)$$

4. All wavelet coefficients at scale j and at position i are multiplied by $y(\Gamma_j(i))$, where y is defined by:

$$\begin{aligned} y(x) &= \left(\frac{m}{c}\right)^p \text{ if } |x| < c \\ y(x) &= \left(\frac{m}{|x|}\right)^p \text{ if } c \leq |x| < m \\ y(x) &= 1 \text{ if } |x| \geq m \end{aligned} \quad (17.14)$$

5. The $\tilde{L}, \tilde{u}, \tilde{v}$ components are reconstructed from the modified wavelet coefficients.

6. The components $\tilde{L}, \tilde{u}, \tilde{v}$ are linearly rescaled to the original range of L, u, v .

7. the \tilde{L} components is mapped nonlinearly according to:

$$\tilde{L}(i) \rightarrow \tilde{L}(i)^{\frac{1}{1-q}} 100^{-\frac{q}{1-q}} \quad (17.15)$$

8. The component \tilde{u}, \tilde{v} are shifted in order to have the same mean as components u and v .

9. The $(\tilde{L}, \tilde{u}, \tilde{v})$ image is transformed into an RGB image.

Four parameters are needed p, q, m, c . p determines the degree of non-linearity in the nonlinear rescaling of the luminance, and must be in $]0, 1[$. q must be in $[-0.5, 0.5]$. When $q > 0$, then darker parts are less enhanced than the lighter parts. When $q < 0$, then the dark parts are more enhanced than lighter parts. Coefficients larger than m are not modified by the algorithm. The c parameter corresponds to the noise level.

The command line is:

USAGE: col_contrast option in_image out_image

where options are:

- **[-f]**
Performs a filtering before the enhancement. Default is no.
- **[-n number_of_scales]**
Number of scales used in the wavelet transform. Default is 4.
- **[-s nsigma]**
Only used when filtering is performed: HardThres = nsigma * SigmaNoise. Default is 3.
- **[-g sigma]**
Noise standard deviation. Only used when filtering is performed. Default is automatically estimated.
- **[-S]**
By default a color saturation is performed. When this option, no color saturation is performed.
- **[-c]**
By default a sigma clipping is performed. When this option, no sigma clipping is performed.
- **[-h]**
Histogram equalization of the L component. Default is no.
- **[-e]**
Multiscale Edge enhancement. Default is no.

- **[-r]**
Retinex method.
- **[-R]**
Multiscale Retinex Method.
- **[-A]**
Logarithm transformation of the wavelet coefficients of the luminance map.
- **[-m M_parameter]**
M Parameter. Only used if “-e” option is set. Default is 100.
- **[-P P_parameter]**
P Parameter. Only used if “-e” option is set. Default is 0.5.
- **[-Q Q_parameter]**
Q Parameter. Only used if “-e” option is set. Default is 0.
- **[-C C_parameter]**
C Parameter. Only used if “-e” option is set. Default is 0.
- **[-K ClippingValue]**
Clipping value. Default is 3.

Examples:

- col_contrast image.tiff image_out.tiff
Enhance the contrast by sigma clipping and color saturation.
- col_contrast -r image.tiff image_out.tiff
Apply the retinex method.
- col_contrast -R image.tiff image_out.tiff
Apply the multiscale retinex method.
- col_contrast -e image.tiff image_out.tiff
Apply the multiscale edge enhancement method.

Chapter 18

MR/3 Three Dimensional Data Set

18.1 MR/3 Multiresolution

18.1.1 Introduction

The 3D à trous Wavelet Transform

The 3D à trous WT of a cube produces J bands, each one being a cube with the same dimension as the input cube. The 3D WT of a cube $C(*, *, *)$ is therefore a 4D data set $W(*, *, *, 0 : J-1)$, where J is the number of resolutions used in the decomposition.

The 3D bi-orthogonal Wavelet Transform

A one-level 3D sub-band decomposition transforms a cube of N^3 pixels into eight sub-cubes, also called bands, of $(\frac{N}{2})^3$ pixels. One of these bands corresponds to the original cube at a lower resolution (smoothed cube), while the seven other bands are the detail information, i.e., information lost between the two resolution levels, original and smoothed data resolution level. A reconstruction of the original input cube can be performed from the eight sub-bands. The 3D wavelet transform (WT) iterates the decomposition process on the smoothed cube. For a 3D WT with P scales, the final number of bands is equal to $7(P-1)+1$, i.e., seven detail bands for the first $P-1$ scales, plus the last smoothed array. For an N^3 pixels cube $D(0..N-1, 0..N-1, 0..N-1)$, the eight bands produced by one-level decomposition (two scales) are stored in the following way:

1. Band 1: $B_1(\frac{N}{2} : N-1, 0 : \frac{N}{2}-1, 0 : \frac{N}{2}-1)$ is obtained by 1D convolution with filters g_x, h_y, h_z .
2. Band 2: $B_2(0 : \frac{N}{2}-1, \frac{N}{2} : N-1, 0 : \frac{N}{2}-1)$ is obtained by 1D convolution with filters h_x, g_y, h_z .
3. Band 3: $B_3(\frac{N}{2} : N-1, \frac{N}{2} : N-1, 0 : \frac{N}{2}-1)$ is obtained by 1D convolution with filters g_x, g_y, h_z .
4. Band 4: $B_4(\frac{N}{2} : N-1, 0 : \frac{N}{2}-1, \frac{N}{2} : N-1)$ is obtained by 1D convolution with filters g_x, h_y, g_z .
5. Band 5: $B_5(0 : \frac{N}{2}-1, \frac{N}{2} : N-1, \frac{N}{2} : N-1)$ is obtained by 1D convolution with filters h_x, g_y, g_z .
6. Band 6: $B_6(\frac{N}{2} : N-1, \frac{N}{2} : N-1, \frac{N}{2} : N-1)$ is obtained by 1D convolution with filters g_x, g_y, g_z .

7. Band 7: $B_7(0 : \frac{N}{2} - 1, 0 : \frac{N}{2} - 1, \frac{N}{2} : N - 1)$ is obtained by 1D convolution with filters $h_x h_y g_z$.
8. Band 8: $B_8(0 : \frac{N}{2} - 1, 0 : \frac{N}{2} - 1, 0 : \frac{N}{2} - 1)$ is obtained by 1D convolution with filters h_x, h_y, h_z .

For the three-scale decomposition, B_8 pixels of the 2-scale decomposition are replaced by bands 8 to 15, each having a size of $(\frac{N}{4})^3$ pixels.

18.1.2 Multiresolution transform of cube: mr3d_trans

The program *mr3d_trans* computes the multiresolution transform of a cube by the à trous WT or a non-redundant transform. The output file which contains the transformation has a suffix, .mr. If the output file name given by the user does not contain this suffix, it is automatically added. The “.mr” file is a FITS format file, and can be manipulated by any package dealing with FITS format, or using the *mr3d_extract* program.

USAGE: **mr3d_trans option cube_in multiresolution_transform_out**

where options are:

- **[-t type_of_multiresolution_transform]**

1. (bi-) orthogonal wavelet transform. Antonini 7/9 filters [7] are used by default, with an L_1 normalization. The filters can be changed using the “-T” option, and an L_2 normalization is obtained by the “-L” option.
2. Wavelet transform via lifting scheme
3. A trous wavelet transform

Default is 1.

- **[-n number_of_scales]**

Number of scales used in the multiresolution transform.

Default is 4.

- **[-T type_of_filters]**

1. Antonini 7/9 filters.
2. Daubechies filter 4.
3. Biorthogonal 2/6 Haar filters.
4. Biorthogonal 2/10 Haar filters.
5. Odegard 7/9 filters.
6. User’s filters.

Default is Antonini 7/9 filters.

- **[-L]**

Use an L_2 normalization. Default is L_1 .

- **[-l type_of_lifting_transform]**

1. Lifting scheme: CDF WT.
2. Lifting scheme: median prediction.
3. Lifting scheme: integer Haar WT.
4. Lifting scheme: integer CDF WT.
5. Lifting scheme: integer (4,2) interpolating transform.

6. Lifting scheme: Antonini 7/9 filters.
7. Lifting scheme: integer Antonini 7/9 filters.

Default is Lifting scheme: integer Haar WT.

- **[i]**

Print statistical information about each band.

The result is stored in a file (suffix “.mr”), and cube (or scales) of the transformation can be extracted by using the *mr3d_extract* program.

Examples:

- mr3d_trans cube.fits cube_trans.mr
Apply the wavelet transform transform to a cube, and store the result in cube_trans.mr.
- mr3d_trans -T 3 -i cube.fits cube_trans.mr
Same as before, but use Haar filters, and print statistical information.
- mr3d_trans -t 2 -l 1 cube.fits cube_trans.mr
Apply a wavelet transform via the lifting scheme.

18.1.3 Extraction of a scale: *mr3d_extract*

The program *mr3d_extract* allows the user to extract a band from a multiresolution transform file (suffix “.mr”).

USAGE: mr3d_extract options multiresolution_file output_cube

where options are:

- **[-b band_number]**
Band number to extract. Default is 1.

Example:

- mr3d_extract -b 2 mr_file.mr cube_band_2.fits
Extract the second band of the wavelet transform, and write as a .fits file of name “cube_band_2.d”.

18.1.4 Insertion of an image: *mr3d_insert*

The program *mr3d_insert* replaces a band by some cube, by inserting it in the multiresolution transform file. The band and the cube must have the same size.

USAGE: mr3d_insert options multiresolution_file input_cube

where options are:

- **[-b band_number]**
Band number to insert. Default is 1.

multiresolution_file is the file (.mr) which contains the multiresolution transformation, *input_cube* is the cube which must replace the band cube. *band_number* specifies the band number to be replaced. Default is 1.

The multiresolution transform is updated.

Example:

- `mr3d_insert -b 3 mr_file.mr cube.fits`
Insert a cube at the third band of the multiresolution file.

18.1.5 Reconstruction: mr3d_recons

The program *mr3d_recons* reconstructs a cube from its multiresolution transform.

USAGE: mr3d_recons multiresolution_file cube_out

multiresolution_file is the file (.mr) which contains the multiresolution transformation, *output_cube* is the output reconstructed cube.

18.2 MR/3 Denoising: mr3d_filter**18.2.1 Gaussian noise filtering: mr3d_filter**

Program *mr_filter* filters a cube. Only Gaussian noise is considered.

USAGE: mr3d_filter option cube_in cube_out

where options are

- **[-t type_of_multiresolution_transform]**
1. (bi-) orthogonal wavelet transform. Antonini 7/9 filters [7] are used by default, with an L_1 normalization. The filters can be changed using the “-T” option, and an L_2 normalization is obtained by the “-L” option.
2. A trous wavelet transform
- **[-n number_of_scales]**
Number of scales used in the multiresolution transform. Default is 4.
- **[-T type_of_filters]**
see 18.1.2.
- **[-g sigma]**
The image contains Gaussian noise, and the standard deviation is given by *sigma*. The option should be set only if the user knows the standard deviation of the noise.
- **[-s NSigma]**
The detection level at each scale is determined by the product of the standard deviation of the noise by the *NSigma*. *NSigma* fixes the confidence interval we want. By default, *NSigma* is equal to 3.
- **[-C]**
Correlated noise.

Examples:

- `mr3d_filter cube_in.fits cube_out.fits`
Filters a cube by multiresolution thresholding, assuming Gaussian noise (its standard deviation is automatically estimated).
- `mr3d_filter -s 4 -g 10. cube_in.fits cube_out.fits`
Same as before, but the noise standard deviation is fixed to 10, and a 4σ thresholding is performed.
- `mr3d_filter -C cube_in.fits cube_out.fits`
Consider correlated noise instead of Gaussian noise. The noise standard deviation is estimated at each scale using the MAD (median absolute deviation) estimator.

18.3 3D Point Clouds Analysis

We will call *Catalog* a set of points in the 3D space, each point being represented by its three coordinates X, Y, Z .

ASCII Catalog

The catalog format is the following:

Catalogue format

- the first line must contain the number of points N , the dimension D (i.e. 3), and the coordinate system S ($S = 1$ or 2). The recognized coordinate systems are:
 1. X and/or Y and/or Z Euclidian system.
 2. Angular coordinate system (longitude, latitude and/or distance; that could be the equatorial system – right ascension α , declination δ , the galactic coordinate system with galactic longitude l , galactic latitude b , the supergalactic coordinate system with SL and SB).
- the D following lines must contains three values: the range of variation of the $i - th$ coordinate (min,max) and a flag indicating the generation model for the corresponding coordinate: 0 – uniform between [min,max], 1 – bootstrapping the coordinate and 2 – uniform on sphere between [min,max] in degrees. When the user have supplied its own generated random catalogs, then those three lines are ignored. This last value is used only by programs which need to simulate data from the input data, and can generally be set to zero.
- the N following lines contains the coordinates of the points.

An example of a 3D catalogue, with 10 points in the Euclidian system, each coordinate being defined in the interval $[0, 100[$ and random catalog coordinates uniform in $[0, 100[$ for each axis, is:

10	3	1
0	100.000	0
0	100.000	0
0	100.000	0
42.1782	13.4610	73.7444
41.6855	9.82727	75.3605
42.3580	14.7867	73.1548
42.0255	12.3347	74.2453
42.7474	17.6581	71.8777
41.9410	11.7113	74.5226
65.5637	9.84036	71.3585
65.7140	12.6019	70.5645
65.5843	10.2196	71.2495
65.4521	7.79074	71.9479

The file name must have the suffixe “cat”.

18.3.1 ASCII Catalog to 3D Cube in FITS Format: `mr3d_cat2fits`

The program *mr3d_cat2fits* allows the user to convert a catalog into a 3D cube in the FITS format using a B_3 spline scaling function. When using the à trous Wavelet Transform, this step corresponds to the projection in the V_0 space, and garanties that the obtained wavelet coefficients are exactly the scalar products of the input irregularly spaced data with the wavelet function.

Assuming Poisson noise, very robust threshold levels can be derived in the wavelet space (program *mr_phisto*), which can be used for the detection of clusters (program *mr_psupport*) or for the filtering (program *mr_pfilter*). See [190] for more details.

USAGE: mr3d_cat2fits options catalog cube_out

where options are:

- **[-B Bin]**
Bin gives the resolution. Default is 1.
- **[-p]**
Do not interpolate the data. By default, a interpolation by a spline a degree 3 is performed. It corresponds to the projection in the space V_0 in the wavelet theory.

Example:

- mr3d_cat2fits data.cat cube_out.fits
Convert a catalog into a 3D fits file.
- mr3d_cat2fits -p -b5 data.cat cube_out.fits
Ditto, but do not use an interpolation, and take a pixel resolution of 5 (units are the same as in the input catalog).
- mr3d_transform -t3 cube_out.fits trans.mr
Wavelet transform of the catalog. The wavelet coefficients in *trans.mr* are exactly the scalar products of the irregularly spaced data with the wavelet function.

18.3.2 Wavelet Histogram Autoconvolution: mr3d_phisto

Program *mr3d_phisto* precomputes a set tables which is used by *mr3d_psupport* and *mr3d_pfilter*. The tables are saved in the FITS table format. They allows the estimation of the detection levels for a wavelet coefficient caculated from n evenements. To derive these thresholds, the wavelet funciton histogram must be autoconvolved n times. The filenames are:

- *_Aba_histo.fits*: 2D array $H[n, 2049]$, which contains n autoconvolutions.
- *_histobin.fits*: 2D array $B_1[n, 2]$, where $B_2[n, 0]$ is the bin for $H[n, *]$ and $B_1[n, 1]$ is the number of points where the $H[n, *]$ is different from zero.
- *_histobound.fits*: 2D array $B_2[n, 2]$, where $B_2[n, 0]$ and $B_2[n, 1]$ are respectively the min and the max of the nth histogram.
- *_histodistrib.fits*: 2D array $D[n, 2049]$, where $D[n, *]$ is the repartition function of the histogram distribution $H[n, *]$.
- *_param.fits*: 1D array $P[2]$, where $P[0]$ and $P[1]$ are respectively the r (rythm variable) and the n (number of calculated histograms) parameters,

USAGE: mr3d_phisto option

where options are

- **[-n number_of_convolution]**
Total number of calculated histograms. Default is 30 (the total of the cube must be lower than 2^n).
- **[-r rythm]**
Parameter relative to the rythm of autoconvolution (default=0). The program computes the 2^n autoconvolutions of the wavelet histogram. It computes also 2^r convolutions between 2^n and $2^{(n+1)}$.

- **[-d]**
Use all default options.
- **[-v]**
Verbose. Default is no.

Examples:

- `mr3d_phisto -d`
Estimates the autoconvolved histograms, and calculates the threshold levels.

18.3.3 Multiresolution Support Calculation: mr3d_psupport

Program *mr3d_psupport* applies a wavelet transform using the à trous algorithm to 3D data, assuming that the noise follows a Poisson distribution, and in the case where we have only few events per pixel [190]. Data can either be given by an ASCII table of coordinates in real values, or by a cube in the fits format.

USAGE: mr3d_psupport options in_cube out_file

where options are

- **[-F first_detection_scale]**
First scale used for the detection. Default is 1.
- **[-e minimum_of_events]**
Minimum number of events for a detection. Default is 5. If a wavelet coefficient has been calculated from fewer events than this minimum value, it is not considered as significant.
- **[-p]**
Detect only positive structure.
- **[-n number_of_scales]**
Number of scales used in the multiresolution transform. Default is 6. The maximum number of scales depends on the cube size. For a $32 \times 32 \times 32$ cube, three scales are allowed (four scale for a $64 \times 64 \times 64$ cube, five for a $128 \times 128 \times 128$ cube, ...).
- **[-E epsilon]**
Value of epsilon used for the threshold (default=1e-3).
- **[-k]**
Suppress isolated pixels in the support. Default is no.
- **[-R]**
Remove the structures near the borders.
- **[-B bin]**
If the input data is a catalogue, it gives the bin of it (default=1).

Examples:

- `mr3d_phisto -d`
Compute a set a table which will be used by *mr3d_psupport*.
- `mr3d_psupport -v simcub32.fits w1`
Multiresolution support estimation.
Creates the two files `w1_support_1.fits` and `w1_support_2.fits` corresponding the support of two wavelet scales.
- `mr3d_psupport -R -E1e-4 -v simcub32.fits w1`
Ditto, but remove isolated pixels, structures at the border, and increase the detection the level.

18.3.4 Data Cube Filtering: mr3d_pfilter

Program *mr3d_pfilter* filters an image (as does *mr3d_filter*) assuming that the noise follows a Poisson distribution and in the case where we have only few events per pixel. Data can either be given by an image or by an ASCII table of coordinates in real values. As for *mr3d_psupport*, the pre-computed table must exist (i.e. the program *mr3d_phisto* must be executed). If the “-c” option is used, the user gives a second input image file name. Then the multiresolution support is estimated from the first image (given by option “-a” or “-I”), and the second image is filtered using the multiresolution support of the first.

USAGE: **mr3d_pfilter** **option** **image_out**

where options are:

- **[-n number_of_scales]**
Number of scales used in the multiresolution transform. Default is 6.
- **[-F first_detection_scale]**
First scale used for the detection. Default is 1.
- **[-E epsilon]**
Value of epsilon used for the threshold (default=1e-3).
- **[-m minimum_of_events]**
Minimum number of events for a detection. Default is 5. If a wavelet coefficient has been calculated from fewer events than this minimum value, it is not considered as significant.
- **[-i number_of_iterations]**
Maximum number of iterations. Default is 0. If this option is set, then an iterative reconstruction method is used to restore the cube.
- **[-B bin]**
If the input data is a catalogue, it gives the bin of it (default=1).
- **[-R]**
Remove the structures near the borders.
- **[-C]**
Smoothness constraint.
- **[-p]**
Detect only positive structure.
- **[-w]**
Write the multiscale support to the disk. Each scale of the multiresolution support is written separately in a FITS file, with name “x_support_j.fits”, where “x” is the output file name and *j* is the scale number.
- **[-S]**
Read the multiresolution support already computed. When we want to apply the filtering several times with different options, the multiresolution can be calculated the first time and saved on the disk with the “-w” option, and for the other experiments can read the pre-computed multiresolution support and do not have to recalculate it.
- **[-k]**
Suppress isolated pixels in the support. Default is no
- **[-K]**
Suppress the last scale.

Examples:

- `mr3d_phisto -d`
Compute a set a table which will be used by *mr3d_pfilter*.
- `mr3d_pfilter cube_in cube_out`
Filtering with all default options.
- `mr3d_pfilter -R -k -E1e-4 cube_in cube_out`
Ditto, but remove isolated pixels and in structures at the border the multiresolution support, and increase the detection the level.

Chapter 19

MR/3 Multi-Channel Data Set

19.1 Introduction

Modern image processing applications often involve multispectral data. This can be color images in R, G and B coordinates, observations of the same area but at different times, etc. Much effort in the compression application domain has been expended in recent years in order to compress efficiently such data sets. The challenge is to have a data representation which takes into account at the same time both the spatial and the spectral (or temporal) correlation. A three-dimensional transform-based coding technique has been proposed in [162], consisting of a one-dimensional spectral Karhunen-Loëve transform [102] (KLT) and a two-dimensional spatial discrete cosine transform (DCT). The KLT is used to decorrelate the spectral domain and the DCT is used to decorrelate the spatial domain. All images are first decomposed into blocks, and each block uses its own Karhunen-Loëve transform instead of one single transform for the whole image. Lee [114] has improved on this approach by introducing a varying block size. The block size is defined using a quadtree, followed by bit allocation for each block. The DCT transform can also be replaced by a wavelet transform [70, 206] (WT).

We present in this chapter how a Wavelet-KLT transform can be used for noise removal. Decorrelating first the data in the spatial domain using the WT and following that in the spectral domain, by the KLT, has the advantage of providing us with robust noise modeling in the WT-KLT space, and hence to be able to filter the transformed data in an efficient and effective way. We show also that the correlation matrix can be computed by different methods based on the noise modeling. These methods are evaluated with a set of different images.

We first present the Wavelet-KLT transform and how to model the noise in this space. We then describe three different methods to filter the coefficients, and finally illustrate these approaches with a set of experiments.

19.2 The Wavelet-Karhunen-Loëve transform

19.2.1 Definition

The Karhunen-Loëve transform, also often referred to as eigenvector, Hotelling transform, or Principal Component Analysis (PCA) [102, 115, 94] allows us to transform discrete signals into a sequence of uncorrelated coefficients. Considering a vector $D = d_1, \dots, d_L$ of L signals or images of dimension N (i.e., N pixels per image), we denote $M = \{m_1, \dots, m_L\}$ the mean vector of the population (m_i is the mean of the i th signal d_i). The covariance matrix C of D is defined by $C = (D - M)(D - M)^t$, and is of order $L \times L$. Each element $c_{i,i}$ of C is the variance of d_i , and each element $c_{i,j}$ is the covariance between d_i and d_j . The KLT method consists of applying the following transform to all vectors $x_i = \{d_1(i), \dots, d_L(i)\}$ ($i = 1..N$):

$$y_i = \Lambda^{-\frac{1}{2}} A(x_i - M) \quad (19.1)$$

where Λ is the diagonal matrix of eigenvalues of the covariance matrix C , and A is a matrix whose rows are formed from the eigenvectors of C [85], ordered following decreasing order of eigenvalues.

Because the rows of A are orthonormal vectors, $A^{-1} = A^t$, and any vector x_i can be recovered from its corresponding y_i by:

$$x_i = \Lambda^{\frac{1}{2}} A^t y_i + M \quad (19.2)$$

The Λ matrix multiplication can be seen as a normalization. Building A from the correlation matrix instead of the covariance matrix leads to another kind of normalization, and the Λ matrix can be suppressed ($y_i = A(x_i - M)$ and $x_i = A^t y_i + M$). Then the norm of y will be equal to the norm of x .

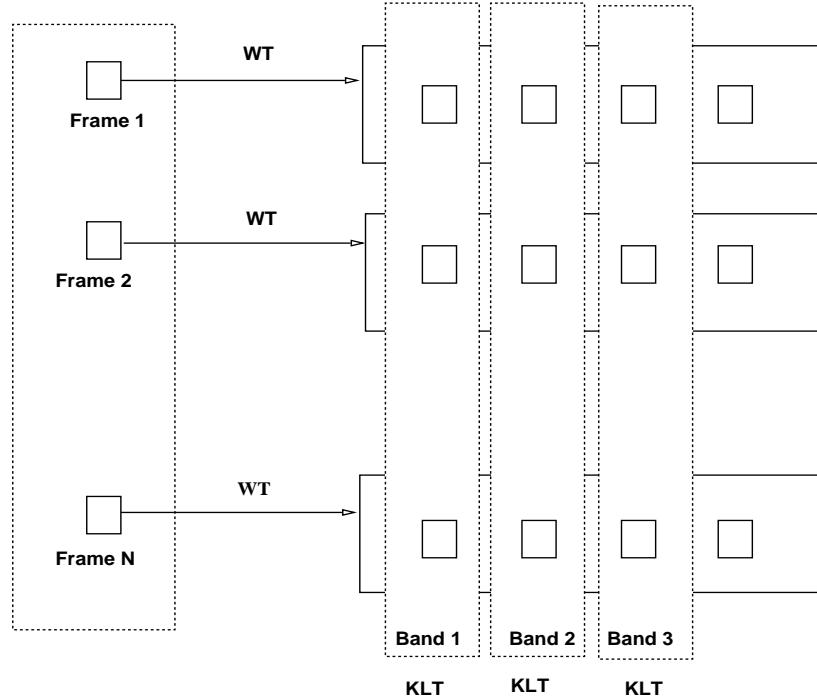


Figure 19.1: WT-KLT transform flowchart. Each frame of the input data set is first wavelet transformed, and a principal component analysis is applied at each resolution level.

We suppose now that we have L observations of the same view, e.g. at different wavelengths (or at different epochs, etc.), and denote as d_l one observation, $W^{(l)}$ its wavelet transform, and $w_{l,j,k}$ one wavelet coefficient at scale j and at position k . The standard approach would be to use an orthogonal wavelet transform, and to calculate the correlation matrix C from the wavelet coefficients instead of the pixel values:

$$C_{m,n} = \frac{\sum_{j=1}^J \sum_{k=1}^{N_j} w_{m,j,k} w_{n,j,k}}{\sqrt{\sum_{j=1}^J \sum_{k=1}^{N_j} w_{m,j,k}^2} \sqrt{\sum_{j=1}^J \sum_{k=1}^{N_j} w_{n,j,k}^2}} \quad (19.3)$$

where J is the number of bands, and N_j is the number of coefficients in the band j . In [114], a more complex approach has been proposed, which is to decompose the images into N_b blocks and apply a KLT to each block separately. We investigate here different approaches for data restoration.

19.2.2 Correlation matrix and noise modeling

We introduce a noise model into our calculation of the correlation matrix. Indeed, if the input sequence D contains noise, then the wavelet coefficient are noisy too. Eigenvalues at the high scales are computed with noisy WT coefficients and we may lose the true underlying correlation that exists between the input images d_l . The expression of the correlation matrix has to be modified in order to allow us to take into account the noise. We add a weighting term to each wavelet coefficient which depends on the signal to noise ratio. The correlation matrix is calculated by:

$$C_{m,n} = \frac{\sum_{j=1}^J \sum_{k=1}^{N_j} p_j(w_{m,j,k}) w_{m,j,k} p_j(w_{n,j,k}) w_{n,j,k}}{\sqrt{\sum_{j=1}^J \sum_{k=1}^{N_j} p_j^2(w_{m,j,k}) w_{m,j,k}^2} \sqrt{\sum_{j=1}^J \sum_{k=1}^{N_j} p_j^2(w_{n,j,k}) w_{n,j,k}^2}} \quad (19.4)$$

where p_j is a weighting function. The standard approach corresponds to the specific case where $p_j(w_m) = 1$ (no weighting). By considering that only wavelet coefficients with high signal to noise ratio should be used for the correlation matrix calculation, p_j can be defined by:

$$p_j(w) = \begin{cases} 1 & \text{if } w \text{ is significant} \\ 0 & \text{if } w \text{ is not significant} \end{cases} \quad (19.5)$$

and a wavelet coefficient w is said to be “significant” if its probability of being due to noise is smaller than a given ϵ value. In the case of Gaussian noise, it suffices to compare the wavelet coefficients w to a threshold level t_j . t_j is generally taken as $\lambda\sigma_j$, where σ_j is the noise standard deviation at scale j , and λ is chosen between 3 and 5. The value of $\lambda = 3$ corresponds to a probability of false detection of 0.27%, for Gaussian statistics.

This hard weighting scheme may lead to problems if only a few coefficient are significant, and can be replaced by a soft weighting one, by defining $p_j(w)$ by:

$$p_j(w) = 1 - \text{Prob}(W > |w|) \quad (19.6)$$

where $\text{Prob}(W > |w|)$ is the probability that a wavelet coefficient is larger than w due to the noise. For Gaussian noise, we have:

$$\begin{aligned} p_j(w) &= 1 - \frac{2}{\sqrt{2\pi}\sigma_j} \int_{|w|}^{+\infty} \exp(-W^2/2\sigma_j^2) dW \\ &= \text{erf}\left(\frac{|w|}{\sqrt{2}\sigma_j}\right) \end{aligned} \quad (19.7)$$

19.2.3 Scale and Karhunen-Loève transform

We can also analyze separately each band of the wavelet transform, and then apply one KLT per resolution level. This implies calculating a correlation matrix $C^{(j)}$ for each band j .

$$C_{m,n}^{(j)} = \frac{\sum_{k=1}^{N_j} p_j(w_{m,j,k}) w_{m,j,k} p_j(w_{n,j,k}) w_{n,j,k}}{\sqrt{\sum_{k=1}^{N_j} p_j^2(w_{m,j,k}) w_{m,j,k}^2} \sqrt{\sum_{k=1}^{N_j} p_j^2(w_{n,j,k}) w_{n,j,k}^2}} \quad (19.8)$$

This has the advantage to take into account more complex behavior of the signal. Indeed, structures of different sizes may have a different spectral behavior (for example, stars and galaxies in astronomical images), and a band-by-band independent analysis allows us to better represent this kind of data.

19.2.4 The WT-KLT transform

The final WT-KLT algorithm has these steps:

1. Estimate the noise standard deviation $\sigma^{(l)}$ of each input data set d_l .
2. Calculate the wavelet transform $W^{(l)}$ of its each input data set d_l .
3. For each band j of the wavelet transform, calculate the correlation matrix $C^{(j)}$ relative to the vector $x_j = \{W_j^{(1)}, W_j^{(2)}, \dots, W_j^{(L)}\}$, where $W_j^{(l)}$ represents the band j of the wavelet transform $W^{(l)}$ of d_l .
4. For each band j , we diagonalize the matrix $C^{(j)}$ and build the transform matrix A_j from the eigenvectors of $C^{(j)}$.
5. For each band j and each position k , we apply the matrix A_j to the vector $x_{j,k} = \{w_{1,j,k}, w_{2,j,k}, \dots, w_{L,j,k}\}$:

$$y_{j,k} = A_j x_{j,k} \quad (19.9)$$

6. The WT-KLT coefficients $c_{l,j,k}$ are derived from $y_{j,k}$ by $c_{l,j,k} = y_{j,k}(l)$. The l index in the transformed coefficients no longer represent the observation number, but instead the eigenvector number. $l = 1$ indicates the main eigenvector while $l = L$ indicates the last one.

The mean vector M disappears in this algorithm because the wavelet coefficients are zero mean.

Figre 19.1 shows the flowchart of the WT-KLT transform.

19.2.5 The WT-KLT reconstruction algorithms

The reconstruction algorithm has these steps:

1. For each band j and each position k , we apply the matrix A_j^t to the vector $y_{j,k} = \{c_{1,j,k}, c_{2,j,k}, \dots, c_{L,j,k}\}$

$$x_{j,k} = A_j^t y_{j,k} \quad (19.10)$$

2. The wavelet coefficients $w_{l,j,k}$ are derived from $x_{j,k}$ by $w_{l,j,k} = x_{j,k}(l)$.
3. An inverse wavelet transform of $W^{(l)}$ furnishes d_l .

19.2.6 WT-KLT Transform of 1D Multichannel Data: `wk1d_trans`

The program `wk1d_trans` computes the WT-KLT transform of a multichannel 1D signal (2D data set). The wavelet transform (à trous algorithm) is applied on each individual 1D signal, and the KL transform is applied for each resolution level, except the last (smooth) scale. So we have a set of eigenvectors for each resolution level $j \{e_{j,1}, \dots, e_{j,N}\}$. The output is a set N multiresolution transform files in FITS format (suffix ".fits"). Eigenvectors are stored by increasing importance. The i th multiresolution file ($i = 1 \dots N$) contains the eigenvectors $\{e_{1,i}, \dots, e_{J-1,i}\}$, where J is the maximum resolution level (last smooth array).

USAGE: `wk1d_trans option data_in prefix_out`

where options are:

- **[-n number_of_scales]**

Number of scales used in the multiresolution transform. Default is 4.

- **[-C]**

Write the correlation matrix to disk. The file name is “Correl_Matrix”. Default is not to do this.

- **[-x CorrelMat_Method]**

- 0: One correlation matrix per band.
- 1: One correlation matrix for all bands.
- 2: One correlation matrix for all band, except the last scale.
- 3: Import correlation matrix.

Default is one correlation matrix per band.

- **[-O Input_CorrelMat_FileName]**

Input correlation matrix file name. Only used when option “-x 3” is set. Default is not to avail of this.

Examples

- `wk1d_trans input.fits wk`

Calculate the WT-KLT of the input data and store the eigenvectors in the files “wk_ev_1.fits, ..., wk_ev_N.fits”. Each file number i contains a 2D data set. A given row j represents eigenvector number i at resolution level j .

- `wk1d_trans -C input.fits wk`

Ditto, but write the calculated correlation matrix to disk.

19.2.7 WT-KLT Reconstruction of 1D Multichannel Data: `wk1d_trec`

The program `wk1d_trec` computes the WT-KLT transform of a multichannel 1D signal (2D data set), and reconstructs the data from a subset of eigenvectors. By default, only the first eigenvector is used at each resolution level.

USAGE: `wk1d_trec option data_in data_out`

where options are:

- **[-n number_of_scales]**

Number of scales used in the multiresolution transform. Default is 4.

- **[-C]**

Write the correlation matrix to disk. The file name is “Correl_Matrix”. Default is not to do this.

- **[-x CorrelMat_Method]**

- 0: One correlation matrix per band.
- 1: One correlation matrix for all bands.
- 2: One correlation matrix for all band, except the last scale.
- 3: Import correlation matrix.

Default is one correlation matrix per band.

- **[-O Input_CorrelMat_FileName]**

Input correlation matrix file name. Only used when option “-x 3” is set. Default is not to use this.

- **[-F NbrEigenVect]**

Number of eigenvectors used for the reconstruction. Default is set to the number of images.

- **[-K EigenVect_Number]**

Eigenvector number which will not be used for the reconstruction.

Examples

- `wk1d_trec input.fits output.fits`
Computes the WT-KLT transform and reconstructs the data from the first eigenvector of each resolution level.
- `wk1d_trec -F 3 input.fits output.fits`
Use the three first eigenvectors
- `wk1d_trec -F 7 -K 2 -K 3 -K 4 trans.fits output.fits`
Use four eigenvectors (i.e., eigenvectors 1,5,6,7).

19.2.8 WT-KLT Transform of 2D Multichannel Data: `wk_trans`

The program `wk_trans` computes the WT-KLT transform of a multichannel image (3D data set) containing N frames ($N < 500$). The wavelet transform is applied to each frame, and the KL transform is applied to each resolution level, except the last (smooth) scale. So we have a set of eigenvectors for each resolution level j $\{e_{j,1}, \dots, e_{j,N}\}$. The output is a set N multiresolution transform files (suffix ".mr"). Eigenvectors are stored by increasing importance. The i th multiresolution file ($i=1..N$) contains the eigenvectors $\{e_{1,i}, \dots, e_{J-1,i}\}$, where J is the maximum resolution level (last smooth array). If the output file name given by the user does not contain the ".mr" suffix, it is automatically added. The ".mr" file is a FITS format file, and can be manipulated by any package dealing with FITS format.

USAGE: `wk_trans option image_in wtklt_transform_out`

where options are:

- **[`-t type_of_multiresolution_transform`]**
 1. linear wavelet transform: à trous algorithm
 2. B-spline wavelet transform: à trous algorithm
 3. wavelet transform in Fourier space
 4. morphological median transform
 5. morphological minmax transform
 6. pyramidal linear wavelet transform
 7. pyramidal B-spline wavelet transform
 8. pyramidal wavelet transform in Fourier space: wavelet = between two resolutions
 9. pyramidal wavelet transform in Fourier space: wavelet = difference between the square of two resolutions
 10. pyramidal median transform
 11. pyramidal Laplacian
 12. morphological pyramidal minmax transform
 13. decomposition on scaling function
 14. (bi-) orthogonal wavelet transform.
Antonini 7/9 filters [7] are used by default, with an L_1 normalization. The filters can be changed using the "-T" option, and an L_2 normalization is obtained by "-L" option.
 15. Feauveau wavelet transform
 16. Feauveau wavelet transform without undersampling
 17. G transform (non-redundant morphological min-max algorithm)
 18. Haar wavelet transform (L_2 normalization).
 19. Half-pyramidal wavelet transform (HPWT)

20. Mixed HPWT and Median method
21. dyadic wavelet transform
22. Mixed WT and PMT method (WT-PMT)
23. Undecimated Haar transform: à trous algorithm
24. Undecimated (bi-) orthogonal wavelet transform.
Antonini 7/9 filters [7] are used by default, with an L_1 normalization. The filters can be changed using the “-T” option, and an L_2 normalization is obtained by “-L” option.

Default is 2.

- **[-T type_of_filters]**

1. Antonini 7/9 filters.
2. Daubechies filter 4.
3. Biorthogonal 2/6 Haar filters.
4. Biorthogonal 2/10 Haar filters.
5. Odegard 7/9 filters.
6. User's filters.

Default is Antonini 7/9 filters.

This option is only available if the chosen transform method is the (bi-) orthogonal transform (-t 14 or -t 24).

- **[-L]**

Use an L_2 normalization. Default is L_1 .

- **[-n number_of_scales]**

Number of scales used in the multiresolution transform. Default is 4.

- **[-C]**

Write the correlation matrix to disk. The file name is “Correl_Matrix”. Default is not to do this.

- **[-x CorrelMat_Method]**

- 0: One correlation matrix per band.
- 1: One correlation matrix for all bands.
- 2: One correlation matrix for all band, except the last scale.
- 3: Import correlation matrix.

Default is one correlation matrix per band.

- **[-O Input_CorrelMat_FileName]**

Input correlation matrix file name. Only used when option “-x 3” is set. Default is not to do this.

Examples

- **wk_trans cube.fits wk**
Calculate the WK-KLT of a 3D data set. If the cube contains N frames, N files will be created (wk_ev_1.mr, ..., wk_ev_N.mr).
- **mr_extract -b 2 wk_ev_1.mr b2_1**
Extract the second band from the multiresolution file “wk_ev_1.mr”. The file “b2_1.fits” contains an image which corresponds to the first eigenvector at the second resolution level.

19.2.9 WT-KLT Reconstruction of 2D Multichannel Data: `wk_trec`

The program `wk_trec` computes the WT-KLT transform of a multichannel image (3D data set), and reconstructs the data from a subset of eigenvectors. The output file contains the reconstruction. Most of the options are the same as those described in section 19.2.8.

USAGE: `wk_trans option image_in wtklt_transform_out`

where options are:

- **[-t type_of_multiresolution_transform]**
see section 19.2.8.
- **[-T type_of_filters]** see section 19.2.8.
- **[-L]** see section 19.2.8.
- **[-n number_of_scales]**
- **[-w]**
Write to disk the eigenvectors multiresolution files. File names are: “`wk_ev_x.mr`”, where `x` is the eigenvector number. Default is not to do this.
- **[-C]** Write the correlation matrix to disk. The file name is “`Correl_Matrix`”. Default is not to do this.
- **[-x]**
 - 0: One correlation matrix per band.
 - 1: One correlation matrix for all bands.
 - 2: One correlation matrix for all band, except the last scale.
 - 3: Import correlation matrix.

Default is one correlation matrix per band.

- **[-O Input_CorrelMat_FileName]**
Input correlation matrix file name. Only used when option “`-x 3`” is set. Default is not to do this.
- **[-F NbrEigenVect]**
Number of eigenvectors used for the reconstruction. Default is set to the number of images.
- **[-K EigenVect_Number]**
Eigenvector number which will not be used for the reconstruction.

Examples

- `wk_trec in_cube.fits out_cube`
Calculate the WT-KLT transform, and reconstruct only from the first eigenvector of each scale.
- `wk_trec -F 5 in_cube.fits out_cube`
Calculate the WT-KLT transform, and reconstruct from the first five eigenvectors of each scale.
- `wk_trec -F 5 -K 2 in_cube.fits out_cube`
Ditto, but do not use eigenvector number 2. Only 4 eigenvectors are used.
- `wk_trec -F 5 -K 2 -t24 trans.fits output.fits`
Ditto, but use the undecimated wavelet transform instead of the à trous algorithm.
- `wk_trec -F 5 -K 1 -K 3 -t24 trans.fits output.fits`
Only three eigenvectors (i.e., eigenvectors numbered 2,4,5) are used.

19.3 Noise Modeling in the WT-KLT Space

Since a WT-KLT coefficient c is obtained by two successive linear transforms, robust noise modeling can be derived in order to know the noise standard deviation associated with the c value.

19.3.1 Non-Gaussian noise

If the noise in the data D is Poisson, the Anscombe transformation [6]

$$t(D) = 2\sqrt{D + \frac{3}{8}} \quad (19.11)$$

acts as if the data arose from a Gaussian white noise model, with $\sigma = 1$, under the assumption that the mean value of I is sufficiently large. The arrival of photons, and their expression by electron counts, on CCD detectors may be modeled by a Poisson distribution. In addition, there is additive Gaussian read-out noise. The Anscombe transformation has been extended to take this combined noise into account. The generalization of the variance stabilizing Anscombe formula is derived as [191]:

$$t(D) = \frac{2}{g}\sqrt{gD + \frac{3}{8}g^2 + \sigma^2 - gm} \quad (19.12)$$

where g is the electronic gain of the detector, σ and m the standard deviation and the mean of the read-out noise.

This implies that for the filtering of an image with Poisson noise or a mixture of Poisson and Gaussian noise, we will first pre-transform the data D into another one $t(D)$ with Gaussian noise. Then $t(D)$ will be filtered, and the filtered data will be inverse-transformed.

For other kinds of noise, modeling must be performed in order to define the noise probability distribution of the wavelet coefficients [191]. In the following, we will consider only stationary Gaussian noise.

19.3.2 Noise level on WT-KLT coefficients

Assuming a Gaussian noise standard deviation σ_l for each signal or image d_l , the noise in the wavelet space follows a Gaussian distribution $\sigma_{l,j}$, j being the scale index. For a bi-orthogonal wavelet transform with a L^2 normalization, $\sigma_{l,j} = \sigma_l$ for all j . Since the WT-KLT coefficients are obtained from a linear transform, we can easily derive the noise standard deviation relative to a WT-KLT coefficient from the noise standard deviation relative to the wavelet coefficients. Considering the noise standard deviation vector $s = \{\sigma_1, \dots, \sigma_L\}$, we apply the following transformation:

$$y_j = A_j^2 s^2 \quad (19.13)$$

and the noise standard deviation relative to a WT-KLT coefficient $c_l(j, k)$ is $\sqrt{y_j(l)}$.

19.4 Multichannel Data Filtering

19.4.1 Introduction

KLT based filtering methods have been proposed in the past [5, 113, 106] for single images. The proposed idea was to decompose the image I of $M \times N$ pixels into non-overlapping blocks B_s of size $N_b \times N_b$. Typically, N_b takes values from 4 to 16. Let $\lambda_1, \lambda_2, \dots, \lambda_n$ be the singular values of the matrix I in decreasing order, and assuming that the matrix I is noisy, the rank r of I has been defined as [107]

$$\lambda_r \geq \epsilon_1 > \lambda_{r+1} \quad (19.14)$$

where ϵ_1 is the norm of the noise. In case of a Gaussian distribution of zero mean, an upper bound is $\sqrt{MN}\sigma$ [106]. A filtered version of I can be obtained by reconstructing each block only from its r first eigenvalues. An original approach has been developed in [147, 106] in order to find the optimal ϵ value, based on the ϵ -compression ratio curve, using a lossless compression method like JPEG. It has been found that the maximum of the second derivative of the curve furnishes the optimal ϵ value [147].

In the case of multichannel data filtering, several different approaches may be considered based on noise modeling. They are presented in this section, and evaluated in the next one.

19.4.2 Reconstruction from a subset of eigenvectors

The WT-KLT transform of a data set $D = \{d_1, \dots, d_L\}$ consists of applying a KLT on the wavelet scales. Hence the vector $W_j = \{W_j^{(1)}, \dots, W_j^{(L)}\}$ of the scales j of the wavelet transforms W^l can be decomposed uniquely as:

$$W_j = U_j \Lambda_j^{\frac{1}{2}} V_j^{-1} = \sum_{i=1}^L \sqrt{\lambda_{j,i}} u_{j,i} v_{j,i}^t \quad (19.15)$$

where Λ_j is the diagonal matrix of eigenvalues of the correlation matrix C_j , U_j and V_j are orthogonal matrices with column vectors $u_{j,i}$ and $v_{j,i}$ which are respectively the eigenvectors of $W_j W_j^t$ and $W_j^t W_j$.

The filtered wavelet coefficients of the band j can be obtained by:

$$\tilde{W}_j = \sum_{i=1}^r \sqrt{\lambda_{j,i}} u_{j,i} v_{j,i}^t \quad (19.16)$$

where r is the rank of the matrix.

19.4.3 WT-KLT Coefficient Thresholding

Hard thresholding can be applied to the WT-KLT coefficients in a fashion analogous to the thresholding of wavelet coefficients.

Example: astronomical source detection

Figure 19.2 shows a simulation. We created a dataset of 18 frames, each of them containing a source (i.e., a point source, or idealized star) at the same position, but at different intensity levels. The source is a small Gaussian, and the source SNR is defined as the ratio between the maximum of the source and the noise standard deviation. Figure 19.2 (top) shows the evolution of the SNR in the 18 frames. Frames two and ten are shown in Figure 19.2, middle left and right. The source SNR ratio is respectively three and one. Figure 19.2, bottom left and right, shows respectively frame ten filtered by the wavelet transform and the WT-KLT. The WT detects only noise, while the WT-KLT clearly identifies the source.

19.4.4 Multiscale Entropy

The multiscale entropy relative to a set of observations $D(1..M)$ can be written as:

$$H(D) = \sum_{l=1}^L \sum_{j=1}^J \sum_{k=1}^{N_j} h(c_{l,j,k}) \quad (19.17)$$

where J is the number of scales used in the wavelet transform decomposition, L the number of observations, k a pixel position, c WT-PCA coefficients, and l denotes the eigenvector number.

The last scale of the wavelet transform is not used, as previously, so this entropy measurement is background independent, which is really important because the background can vary from one wavelength to another.

As for a wavelet coefficients in the case of mono-channel data, we know the noise standard deviation relative to a coefficient, and coefficients are of zero mean. Therefore, we can apply the same filtering method. The filtered WT-PCA coefficients are found by minimizing for each $c_{l,j,k}$:

$$j(\tilde{c}_{l,j,k}) = h_s(c_{l,j,k} - \tilde{c}_{l,j,k}) + \alpha h_n(\tilde{c}_{l,j,k}) \quad (19.18)$$

Example

Figure 19.3 relates to a simulation using the same dataset as in the previous example (i.e., a dataset of 18 frames). Additive noise was used, and the data were filtered. We calculated the Root Mean Square Error (RMSE) on each individual frame on a 5×5 square centered on the source. Hence, the RMSE reflects well the photometric errors, and the addition of the 18 RMSE values, which we call IRMSE (Integrated RMSE), furnishes us a reliable measurement of the filtering quality. The simulation was repeated with 12 noise levels, and four different filtering methods were compared. Figure 19.3 shows the IRMSE versus the noise standard deviation plot. The four methods are (i) multiscale entropy applied to the WT-KLT coefficients (diamond), (ii) reconstruction from a subset of eigenvectors of the KLT (triangle), (iii) multiscale entropy applied to each frame independently (square), and (iv) thresholding applied to the wavelet transform of each frame (star). This simulation shows clearly that the approach proposed here, multiscale entropy applied to the WT-KLT coefficients, outperforms all other methods.

The same experiments were performed using a simulated Planck data set. Planck is an upcoming European Space Agency mission to carry out measurements of the cosmic microwave background. The data set contains ten images, each one being a linear combination of 6 sky component images (CMB, SZ, free-free, etc.). As in the previous simulation, noise was added, and the data were filtered by the four methods. The only difference is that the RMSE is calculated on the full frames. Figure 19.4 shows IRMSE versus the noise standard deviation plot. Diamonds, triangles, squares and stars represent the same methods as before. Again, the multiscale entropy applied to the WT-KLT coefficients outperforms the other methods.

19.4.5 WT-KLT Filtering of 1D Multichannel Data: `wk1d_filter`

The program `wk1d_filter` filters a multichannel signal (2D data set).

USAGE: `wk1d_filter option data_in data_out`

where options are:

- **[-n number_of_scales]**
see section 19.2.8.
- **[-g sigma]**
The image contains Gaussian noise, and the standard deviation is given by *sigma*. This option should be set only if the user knows the standard deviation of the noise.
- **[-x]**
see section 19.2.8.
- **[-O Input_CorrelMat_FileName]**
see section 19.2.8.
- **[-y NoiseCorrelationType]**
 - 0: Compute the correlation matrix with all wavelet coefficients.
 - 1: Compute the correlation matrix only with significant wavelet coefficients.
 - 2: Compute the correlation matrix with weighted wavelet coefficients.

Default is 1.

- **[-s NSigma]**
Wavelet coefficients are significant when their absolute values are larger than $nsigma * SigmaNoise$. This option has an effect when the “-y” option is set to 1 or 2. Default $NSigma$ value is 3.
- **[-S NSigma]**
The WT-KLT coefficients are at $NSigma * SigmaNoise$. Default is 3.
- **[-P]**
Positivity constraint. Default is not to enforce this.
- **[-F NbrEigenVect]**
Number of eigenvectors used for the reconstruction. Default is set to the number of images.
- **[-K EigenVect_Number]**
Eigenvector number which will not be used for the reconstruction.
- **[-C]**
see section 19.2.8.

Examples

- `wk1d_filter input.fits output.fits`
Filter the 1D multi-channel data set using the WT-KLT transform.
- `wk1d_filter -S 5 input.fits output.fits`
Ditto, but smooth the WT-KLT at 5 sigma instead of 3.

19.4.6 WT-KLT Filtering of 2D Multichannel Data: `wk_filter`

The program `wk_filter` filters a multichannel image (3D data set).

USAGE: `wk_filter option data_in data_out`

where options are:

- **[-t type_of_multiresolution_transform]**
see section 19.2.8.
- **[-T type_of_filters]**
see section 19.2.8.
- **[-L]**
see section 19.2.8.
- **[-n number_of_scales]**
see section 19.2.8.
- **[-g sigma]**
Noise standard deviation. Default is automatically calculated.
- **[-x]**
see section 19.2.8.
- **[-O Input_CorrelMat_FileName]**
see section 19.2.8.
- **[-y NoiseCorrelationType]**
 - 0: Compute the correlation matrix with all wavelet coefficients.
 - 1: Compute the correlation matrix only with significant wavelet coefficients.

- 2: Compute the correlation matrix with weighted wavelet coefficients.

Default is 1.

- **[-s NSigma]**

Wavelet coefficients are significant when their absolute values are larger than $nsigma * SigmaNoise$. This option has an effect when the “-y” option is set to 1 or 2. Default *NSigma* value is 3.

- **[-S NSigma]**

The WT-KLT coefficients are at $NSigma * SigmaNoise$. Default is 3.

- **[-P]**

Positivity constraint. Default is not to enforce this.

- **[-b]**

Maximum image constraint at 255. Default is not to use this.

- **[-w]**

see section 19.2.8.

- **[-F NbrEigenVect]**

Number of eigenvectors used for the reconstruction. Default is set to the number of images.

- **[-K EigenVect_Number]**

Eigenvector number which will not be used for the reconstruction.

- **[-C]**

see section 19.2.8.

Examples

- `wk_filter in_cube.fits out_cube.fits`
Filter the cube using all default options.
- `wk_filter -S 5 in_cube.fits out_cube.fits`
Threshold the WT-KLT coefficients at 5 sigma, instead of 3 sigma.
- `wk_filter -S 5 -t24 in_cube.fits out_cube.fits`
Ditto, but use an undecimated wavelet transform.

19.4.7 WT-KLT Filtering of 2D Multichannel Data by the Multiscale Entropy Method: `wk_memfilter`

The program *wk_memfilter* filters a multichannel image (3D data set) by the multiscale entropy method.

USAGE: `wk_memfilter` option cube_in cube_out

where options are:

- **[-t type_of_multiresolution_transform]**
see section 19.2.8.

- **[-T type_of_filters]**
see section 19.2.8.

- **[-L]**
see section 19.2.8.

- **[-n number_of_scales]**
see section 19.2.8.

- **[-g sigma]**
see section 19.4.5.
- **[-x]**
see section 19.2.8.
- **[-O]**
see section 19.2.8.
- **[-y NoiseCorrelationType]**
see section 19.4.6.
- **[-s nsigma]**
see section 19.4.6.
- **[-P]**
see section 19.4.6.
- **[-b]**
see section 19.4.6.
- **[-w]**
see section 19.4.6.
- **[-F NbrEigenVect]**
see section 19.4.6.
- **[-K EigenVect_Number]**
see section 19.4.6.
- **[-C CvgParam]**
Convergence parameter. Default is $1e - 2$.
- **[-G RegulParam]** Regularization parameter. Default is 1.
- **[-U Type_of_Regularization]**

1. Use a fixed user Alpha value.
2. Estimate the optimal Alpha.
3. Estimate one Alpha value per band.

Default is 1.

- **[-D]**
Alpha is modified using the data SNR. Default is no.
- **[-i MaxIter]**
Maximum number of iterations. Default is 20.

Examples

- `wk_memfilter -U 3 in_cube.fits out_cube.fits`
Filtering using one regularization parameter per resolution level.
- `wk_memfilter -U 3 -D in_cube.fits out_cube.fits`
Ditto, but protect wavelet coefficients with high SNR from the regularization.
- `wk_memfilter -t24 -U 3 -D in_cube.fits out_cube.fits`
Ditto, but use the undecimated wavelet transform.

19.5 Filtering using the Haar-Multichannel Transform

19.5.1 Definition

We have seen in MR/1 that the Haar transform presents some advantages, especially when the data contains Poisson noise.

In order to decorrelate the information both spatially and in wavelength (or in time), a 2D-wavelet transform must first be performed on each frame of the date cube $D(x, y, z)$. We denote the result $w_{j,k_x,k_y,z}$, where j is the scale index ($j \in [1, J]$, and where J is the number of scales), k_x, k_y the spatial position in the scale ($k_x \in [0, N_x - 1]$, and $k_y \in [0, N_y - 1]$), and z is the frame number. This set must again be transformed in order to decorrelate the information in the third dimension. We apply a 1D-transform to each vector $w_{j,k_x,k_y,z}$, and we get a new set of data u_{j,j',k_x,k_y,k_z} , where j' and k_z are respectively the scale index and position in the third dimension.

Using the unnormalized Haar transform, a coefficient $u_{j+1,j'+1,k,l,t}$ can be written as:

$$u_{j+1,j'+1,k_x,k_y,k_z} = \sum_{i=2^{j'+1}k_z}^{2^{j'}k_z+2^{j'}-1} w_{j+1,k_x,k_y,i} - \sum_{i=2^{j'+1}k_z+2^{j'}}^{2^{j'}+1(k_z+1)-1} w_{j+1,k_x,k_y,i} \quad (19.19)$$

We assume now a constant background rate λ_i for each frame i of the data cube. Each Haar wavelet coefficient $w_{j+1,k_x,k_y,i}$ is the difference between two random variables X_i and Y_i , which follows a Poisson distribution of parameter $\lambda_{j,i}$, where $\lambda_{j,i}$ represents the rate per pixel over 2^{j+1} pixels of the i th frame, and is equal to $2^{2j}\lambda - i$. Then $u_{j+1,j'+1,k_x,k_y,k_z}$ is the difference of two variables, $X = \sum_i X_i$ and $Y = \sum_i Y_i$, and both follow a Poisson distribution of parameter $\sum_i \lambda_{j,i}$. The thresholding method described in MR/1 can therefore be used, using the correct λ value.

Filtering

The multichannel filtering algorithm is:

1. For each frame $D(*, *, z)$, apply the 2D-Haar transform: we obtain $w_{j,k,l}(z)$.
2. For each scale j , and at each position k_x, k_y , extract the 1D vector $w_{j,k_x,k_y,*}$ and compute its wavelet transform. We get a new data set.
3. Threshold the coefficients using the correct λ value.
4. Inverse 1D transform.
5. Inverse 2D transform.

For non-constant background, a coarse to fine approach, as explained in MR/1, can be used.

19.5.2 WT-KLT Filtering of 2D Multichannel Data by the Multiscale Haar Transform: `ww_filter`

The program `ww_filter` filters a multichannel image (3D data set) by applying a 2D WT following by a 1D WT in the third direction.

USAGE: `ww_filter option data_in data_out`

where options are:

- **[-t type_of_multiresolution_transform]**
see section 19.2.8.

- **[-T type_of_filters]**
see section 19.2.8.
- **[-L]**
see section 19.2.8.
- **[-n number_of_scales]**
see section 19.2.8.
- **[-g sigma]**
see section 19.4.5.
- **[-s nsigma]**
see section 19.4.6.
- **[-S nsigma]**
see section 19.4.6.
- **[-P]**
see section 19.4.6.

Examples

- `ww_filter -t 18 -T 3 input.fits output.fits`
Filtering using the Haar wavelet transform.
- `ww_filter -t 24 -T 3 input.fits output.fits`
Ditto, but use an undecimated wavelet transform.
- `ww_filter -t 24 -T 3 -S 5 input.fits output.fits`
Increase the thresholding level.

19.6 Independent Component Analysis

The idea of independent component analysis (ICA) originates from the problem of *blind source separation* (BSS), and consists of recovering unobserved signals or “sources” from several observed mixtures [35]. Assuming that n statistically independent signals $s_1(t), \dots, s_n(t)$ are mixed by an unknown $n \times n$ mixing matrix $A = [a_{ij}]$, we have:

$$X(t) = AS(t) \quad (19.20)$$

where $S(t) = [s_1(t), \dots, s_n(t)]^t$, and $X(t) = [x_1(t), \dots, x_n(t)]^t$ represents the observed signals, with $x_i(t) = \sum_{j=1}^n a_{ij}s_j(t)$. The challenge is to find how to achieve separation using as our only assumption that the source signals are statistically independent. The solution consists of finding an $n \times n$ separating matrix B , $Y(t) = BS(t)$, such that Y is an estimate of S . This is achieved by minimizing contrast functions Φ , which are defined in terms of the Kullback-Leibler divergence K :

$$\Phi(Y) = \int p_Y(u) \log \frac{p_Y(u)}{\prod p_{Y_i}(u_i)} du \quad (19.21)$$

The mutual information, expressed by the Kullback-Leibler divergence, vanishes if and only if the variables Y_i are mutually independent, and is strictly positive otherwise.

ICA has been used in astronomy to analyze multispectral images [149] of the galaxy 3C 120 and to separate the Cosmic Microwave Background from other sky components [10]. As for PCA, it has been shown [218, 219] that applying ICA to wavelet transformed signals leads to better quality results, especially in the presence of noise.

Example

Fig. 19.5 shows three simulated signals. Six observed data sets were derived from the three sources, by a linear combination (see Fig. 19.6). Without any knowledge of the mixing matrix, the JADE-ICA [35] method was applied and the three signals were reconstructed from the six observed data sets (see Fig. 19.7).

19.6.1 JADE-ICA IDL Programs

JADE

Apply the Independant Componant Method to a set of vectors using the JADE method.

USAGE: `jade, ObservSig, NbSource, DeMixingMat, Process, Verbose=Verbose`

where

- *ObservSig*: Input Data (*ObservSig* = A # original signal). *ObservSig*($i, *$) = i th vector.
- *NbSource*: Input number of sources in the input signal.
- *DeMixingMat*: Output demixing matrix.
- *Process*: Output Reconstructed process = *DeMixingMat* # *ObservSig*. *Process*($i, *$) = i th Process, with $i = 0..NbSource - 1$.

JADE1D

Apply the Independant Componant Method to a set of vectors using the JADE method. The routine is identical to *jade* routine, except that the input-output vectors are ordered differently ($V(*, i)$ for the i th vector, instead of $V(i, *)$).

USAGE: `jade1d, ObservSig, NbSource, DeMixingMat, Process, Verbose=Verbose`

where

- *ObservSig*: Input Data ($\text{ObservSig} = A \#$ original signal). ; $\text{ObservSig}(*, i) =$ *i*th vector.
- *NbSource*: Input number of sources in the input signal.
- *DeMixingMat*: Output demixing matrix.
- *Process*: Output Reconstructed process = *DeMixingMat* # *ObservSig*. $\text{Process}(*, i) =$ *i*th Process, with $i = 0..NbSource - 1$.

JADE2D

Apply the Independant Componant Method to a set of images using the JADE method. The routine is an extension to 2d of jade1d routine. The input set of images must be given with the following syntax: $\text{ObservSig}(*, *, i) =$ *i*th image. If the “/wave” keyword is set, a 2D wavelet transform is first perform on each image, JADE is applied on the wavelet transformed images, and an inverse wavelet transform is applied on the reconstructed process. The wavelet transform is performed by calling the program *mr_transform*.

USAGE: **jade2d, ObservSig, NbSource, DeMixingMat, Process, wave=wave,**
optw=optw, Verbose=Verbose

where

- *ObservSig*: Input Data ($\text{ObservSig} = A \#$ original signal). $\text{ObservSig}(*, *, i) =$ *i*th vector.
- *NbSource*: Input Number of sources in the input signal.
- *DeMixingMat*: Output demixing matrix.
- *Process*: Output Reconstructed process = *DeMixingMat* # *ObservSig*. $\text{Process}(*, *, i) =$ *i*th Process, with $i = 0..NbSource - 1$.
- *wave*:: if set, an bi-orthogonal wavelet transform is applied on each image, and the component separation is performed on the wavelet transformed images. The inverse wavelet transform is then applied on the reconstructed process.
- *optw*: string = options for the wavelet transformation

Examples:

- **jade2d, ObservIma, NbSource, DeMixingMat, RecIma**
 Apply the JADE method to *ObservIma*.
- **jade2d, ObservIma, NbSource, DeMixingMat, RecIma, /wave, optw='n5'**
 Ditto, but applied a wavelet transform with five scales on each image.

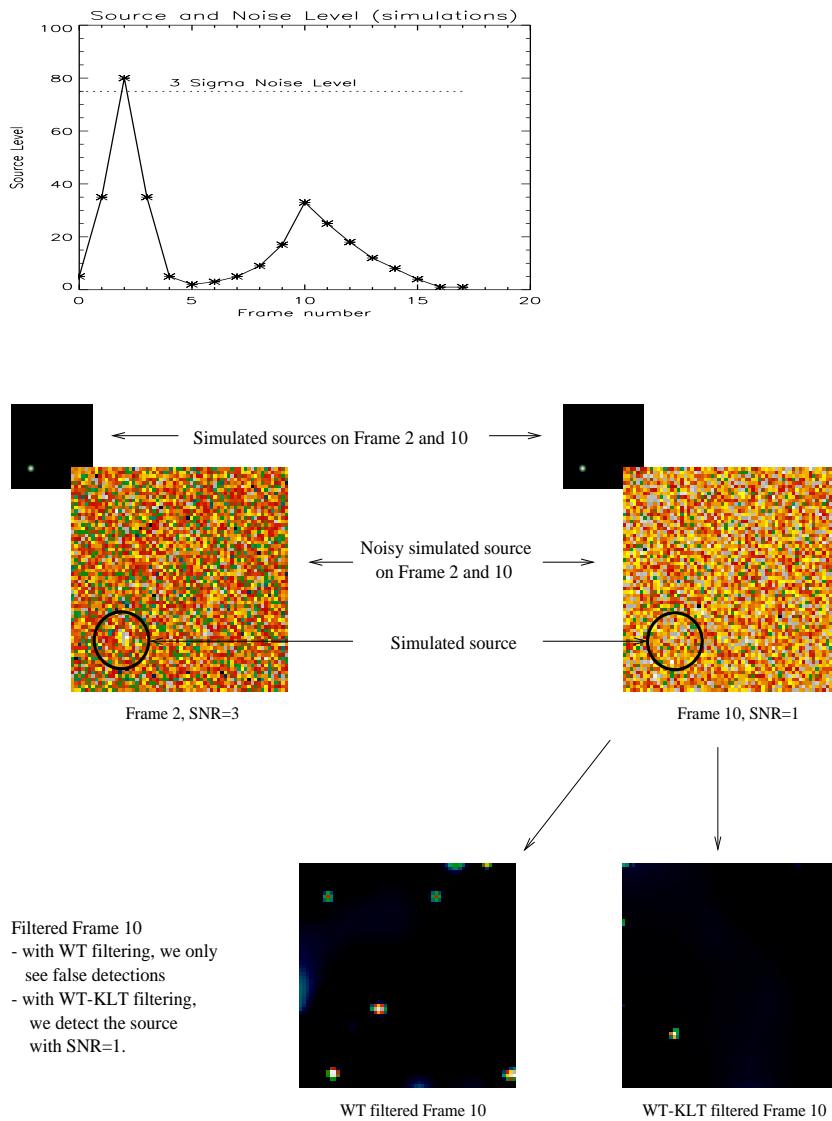


Figure 19.2: Simulation: the dataset is composed of 18 frames. Each of them contains a source (small Gaussian) at the same position, but at different intensity levels. Top, plot of the source maximum value versus the frame number. Middle, frames 2 and 10, and bottom, filtered version of the frame 10 by the wavelet transform and wavelet Karhunen-Loève transform.

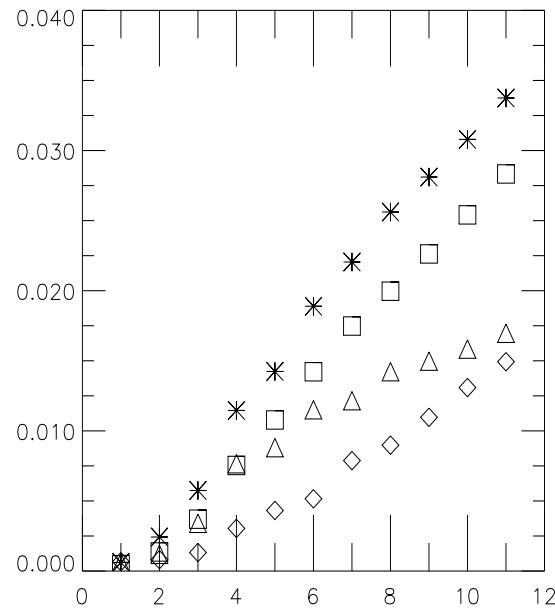


Figure 19.3: Simulation: Root Mean Square Error versus the noise standard deviation. See text.

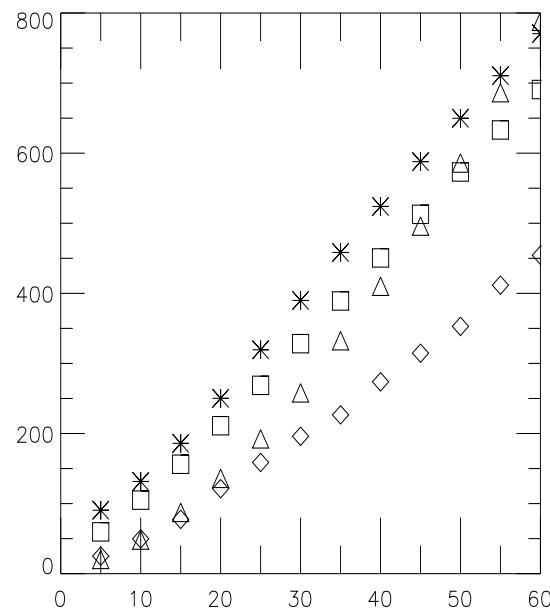


Figure 19.4: Planck Simulation: Root Mean Square Error versus the noise standard deviation. See text.

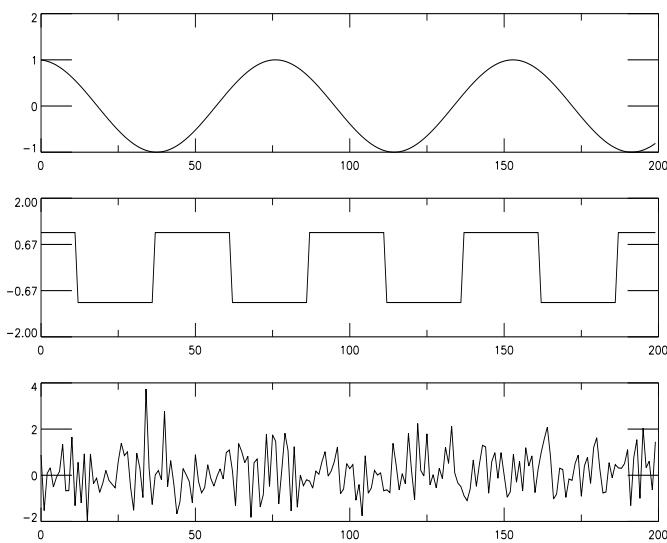


Figure 19.5: Example of three simulated sources.

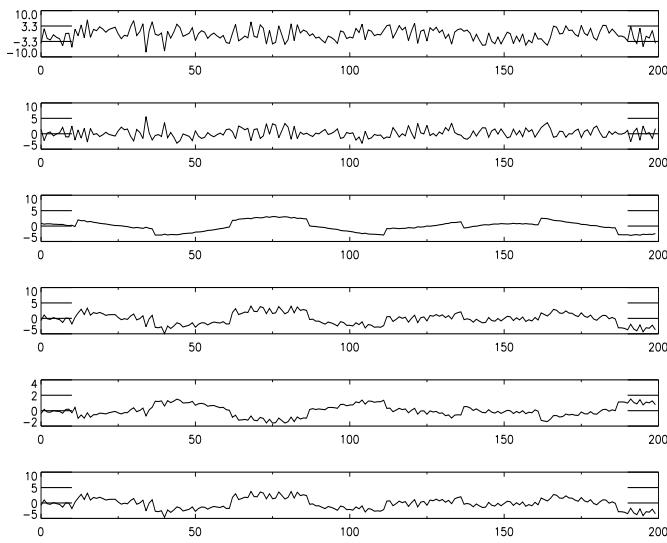


Figure 19.6: Mixed sources. Each of these six signals is a linear combination of the three simulated sources.

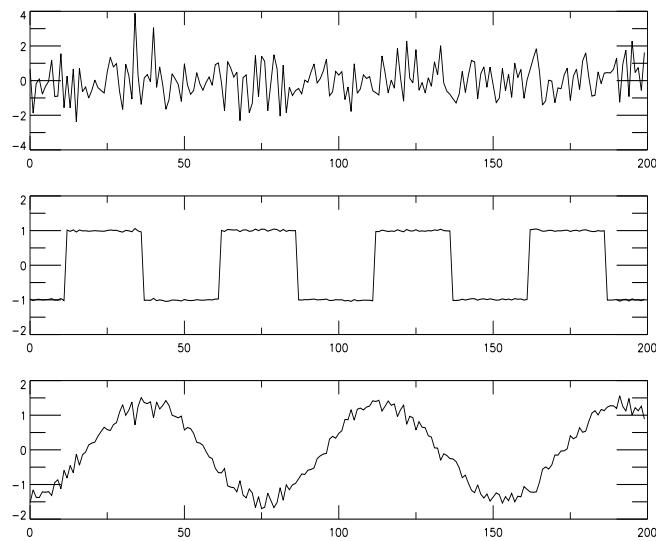


Figure 19.7: Reconstructed sources from the mixed signals.

Part IV

MR/4 : Ridgelets and Curvelets

Chapter 20

The Curvelet Transform

20.1 Introduction

Wavelets and related multiscale representations pervade all areas of signal processing. The recent inclusion of wavelet algorithms in JPEG 2000, the new still-picture compression standard, testifies to this lasting and significant impact. The most used wavelet transform algorithm is the decimated bi-orthogonal wavelet transform (OWT). Using the OWT, a signal s can be decomposed by:

$$s(l) = \sum_k c_{J,k} \phi_{J,l}(k) + \sum_k \sum_{j=1}^J \psi_{j,l}(k) w_{j,k} \quad (20.1)$$

with $\phi_{j,l}(x) = 2^{-j}\phi(2^{-j}x - l)$ and $\psi_{j,l}(x) = 2^{-j}\psi(2^{-j}x - l)$, where ϕ and ψ are respectively the scaling function and the wavelet function. J is the number of resolutions used in the decomposition, w_j the wavelet (or detail) coefficients at scale j , and c_J is a coarse or smooth version of the original signal s . Thus, the algorithm outputs $J+1$ subband arrays. The indexing is such that, here, $j = 1$ corresponds to the finest scale (high frequencies).

The application of the OWT to image compression, using the 7-9 filters [7] and the zerotree coding [167, 163] has lead to impressive results, compared to previous methods like JPEG.

A series of recent papers [32, 31], however, argued that wavelets and related classical multiresolution ideas are playing with a limited dictionary made up of roughly isotropic elements occurring at all scales and locations. We view as a limitation the facts that those dictionaries do not exhibit highly anisotropic elements and that there is only a fixed number of directional elements, independent of scale. Despite the success of the classical wavelet viewpoint, there are objects, e.g. images, that do not exhibit isotropic scaling and thus call for other kinds of multiscale representation. In short, the theme of this line of research is to show that classical multiresolution ideas only address a portion of the whole range of interesting multiscale phenomena and that there is an opportunity to develop a whole new range of multiscale transforms.

Following on this theme, Candès and Donoho introduced new multiscale systems like curvelets [31] and ridgelets [30] which are very different from wavelet-like systems. Curvelets and ridgelets take the form of basis elements which exhibit very high directional sensitivity and are highly anisotropic. In two-dimensions, for instance, curvelets are localized along curves, in three dimensions along sheets, etc. Continuing at this informal level of discussion we will rely on an example to illustrate the fundamental difference between the wavelet and ridgelet approaches, postponing the mathematical description of these new systems.

We investigate in this document the best way to implement the ridgelet and the curvelet transform for the purpose of image restoration. The second and third sections describe respectively the ridgelet transform and the curvelet transform. Section four shows how the ridgelet and the wavelet coefficients can be thresholded in order to filter an image. Comparisons with other methods are presented. Instructions for using the programs are given in the last section.

Consider an image which contains a vertical band embedded in white noise with relatively large amplitude. Figure 20.1 (top left) represents such an image. The parameters are as follows:

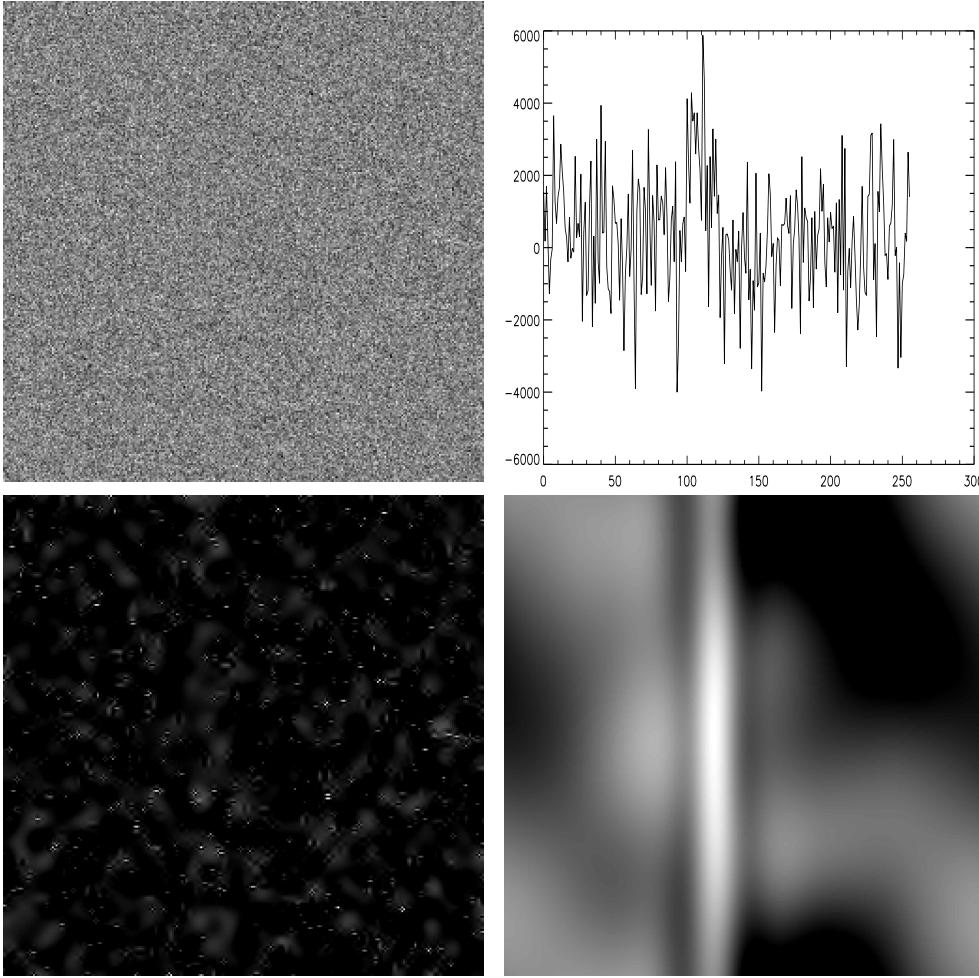


Figure 20.1: Top left, original image containing a vertical band embedded in white noise with relatively large amplitude. Top right, row sums illustrating the hardly perceptable band. Bottom left, reconstructed image using undecimated wavelet coefficients. Bottom right, reconstructed image using ridgelet coefficients.

the pixel width of the band is 20 and the SNR (signal-to-noise ratio) is set to be 0.1. Note that it is not possible to distinguish the band by eye. The wavelet transform (undecimated wavelet transform) is also incapable of detecting the presence of this object; roughly speaking, wavelet coefficients correspond to averages over approximately isotropic neighborhoods (at different scales) and those wavelets clearly do not correlate very well with the very elongated structure (pattern) of the object to be detected.

We now turn our attention towards procedures of a very different nature which are based on line measurements. To be more specific, consider an *ideal* procedure which consists of integrating the image intensity over columns; that is, along the orientation of our object. We use the adjective “ideal” to emphasize the important fact that this method of integration requires *a priori* knowledge about the structure of our object. This method of analysis gives of course an improved signal-to-noise ratio for our linear functional better which is better correlated with the object in question: see the top right panel of Figure 20.1.

This example will make our point. Unlike wavelet transforms, the ridgelet transform processes data by first computing integrals over lines with all kinds of orientations and locations. We will explain in the next section how the ridgelet transform further processes those line integrals. For now, we apply naive thresholding of the ridgelet coefficients and “invert” the

ridgelet transform; the bottom right panel of Figure 20.1 shows the reconstructed image. The qualitative difference with the wavelet approach is striking. We observe that this method allows the detection of our object even in situations where the noise level (standard deviation of the white noise) is five times superior to the object intensity.

20.2 Continuous Ridgelet Transform

The two-dimensional continuous ridgelet transform in \mathbf{R}^2 can be defined as follows [30]. We pick a smooth univariate function $\psi : \mathbf{R} \rightarrow \mathbf{R}$ with sufficient decay and satisfying the admissibility condition

$$\int |\hat{\psi}(\xi)|^2 / |\xi|^2 d\xi < \infty, \quad (20.2)$$

which holds if, say, ψ has a vanishing mean $\int \psi(t) dt = 0$. We will suppose that ψ is normalized so that $\int |\hat{\psi}(\xi)|^2 \xi^{-2} d\xi = 1$.

For each $a > 0$, each $b \in \mathbf{R}$ and each $\theta \in [0, 2\pi)$, we define the bivariate *ridgelet* $\psi_{a,b,\theta} : \mathbf{R}^2 \rightarrow \mathbf{R}^2$ by

$$\psi_{a,b,\theta}(x) = a^{-1/2} \cdot \psi((x_1 \cos \theta + x_2 \sin \theta - b)/a); \quad (20.3)$$

this function is constant along lines $x_1 \cos \theta + x_2 \sin \theta = const$. Transverse to these ridges it is a wavelet.

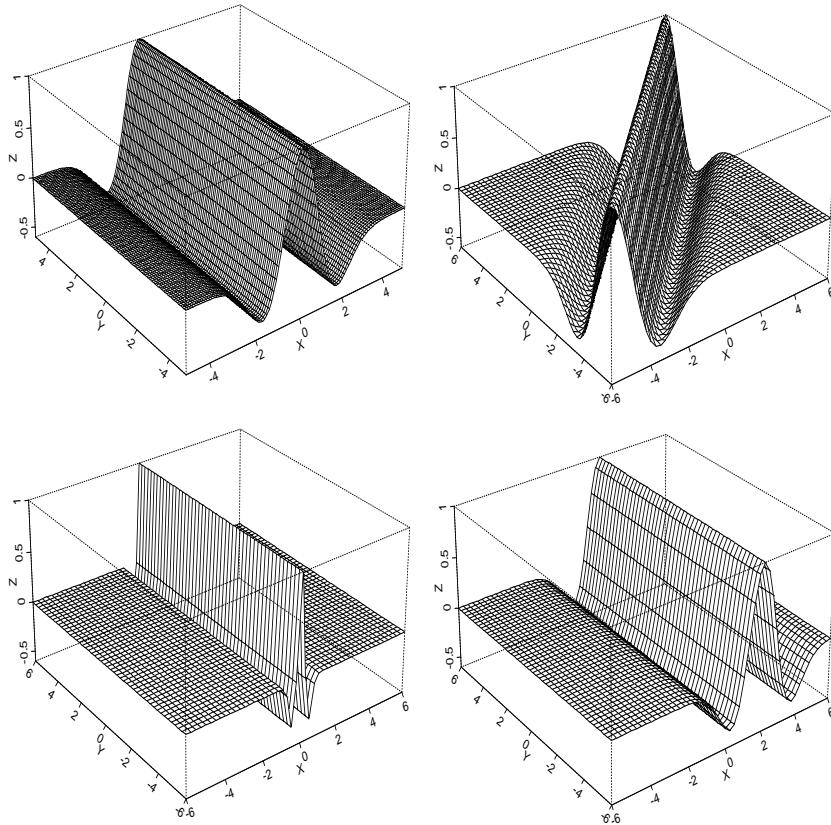


Figure 20.2: A few ridgelets.

Figure 20.2 graphs a few ridgelets with different parameter values. The top right, bottom left and right panels are obtained after simple geometric manipulations of the upper left ridgelet, namely rotation, rescaling, and shifting.

Given an integrable bivariate function $f(x)$, we define its ridgelet coefficients by

$$\mathcal{R}_f(a, b, \theta) = \int \psi_{a,b,\theta}(x) f(x) dx.$$

We have the exact reconstruction formula

$$f(x) = \int_0^{2\pi} \int_{-\infty}^{\infty} \int_0^{\infty} \mathcal{R}_f(a, b, \theta) \psi_{a,b,\theta}(x) \frac{da}{a^3} db \frac{d\theta}{4\pi} \quad (20.4)$$

valid a.e. (almost everywhere) for functions which are both integrable and square integrable. Furthermore, this formula is stable since we have a Parseval relation

$$\int |f(x)|^2 dx = \int_0^{2\pi} \int_{-\infty}^{\infty} \int_0^{\infty} |\mathcal{R}_f(a, b, \theta)|^2 \frac{da}{a^3} db \frac{d\theta}{4\pi}. \quad (20.5)$$

Hence, much like the wavelet or Fourier transforms, the identity (20.4) expresses the fact that one can represent any arbitrary function as a continuous superposition of ridgelets. Discrete analogs of (20.4)-(20.5) exist, see [30], or [64] for a slightly different approach.

20.2.1 The Radon Transform

A basic tool for calculating ridgelet coefficients is to view ridgelet analysis as a form of wavelet analysis in the Radon domain. We recall that the Radon transform of an object f is the collection of line integrals indexed by $(\theta, t) \in [0, 2\pi) \times \mathbf{R}$ given by

$$Rf(\theta, t) = \int f(x_1, x_2) \delta(x_1 \cos \theta + x_2 \sin \theta - t) dx_1 dx_2, \quad (20.6)$$

where δ is the Dirac distribution. The ridgelet coefficients $\mathcal{R}_f(a, b, \theta)$ of an object f are given by analysis of the Radon transform via

$$\mathcal{R}_f(a, b, \theta) = \int Rf(\theta, t) a^{-1/2} \psi((t - b)/a) dt.$$

Hence the ridgelet transform is precisely the application of a 1-dimensional wavelet transform to the slices of the Radon transform where the angular variable θ is constant and t is varying.

20.2.2 Ridgelet Pyramids

Let Q denote a dyadic square $Q = [k_1/2^s, (k_1 + 1)/2^s) \times [k_2/2^s, (k_2 + 1)/2^s)$ and let \mathcal{Q} be the collection of all such dyadic squares. We write \mathcal{Q}_s for the collection of all dyadic squares of scale s . Associated with the squares $Q \in \mathcal{Q}_s$ we construct a partition of energy as follows. With w a nice smooth window obeying $\sum_{k_1, k_2} w^2(x_1 - k_1, x_2 - k_2) = 1$, we dilate and transport w to all squares Q at scale s , producing a collection of windows (w_Q) such that the w_Q^2 's, $Q \in \mathcal{Q}_s$, make up a partition of unity. We also let T_Q denote the transport operator acting on functions g via

$$(T_Q g)(x_1, x_2) = 2^s g(2^s x_1 - k_1, 2^s x_2 - k_2).$$

With this notation, it is not hard to see that

$$fw_Q = \int \langle f, w_Q T_Q \psi_{a,b,\theta} \rangle T_Q \psi_{a,b,\theta} \frac{da}{a^3} db \frac{d\theta}{4\pi}$$

and, therefore, summing the above equality across squares at a given scale gives

$$f = \sum_{Q \in \mathcal{Q}_s} fw_Q^2 = \sum_Q \int \langle f, w_Q T_Q \psi_{a,b,\theta} \rangle w_Q T_Q \psi_{a,b,\theta} \frac{da}{a^3} db \frac{d\theta}{4\pi}. \quad (20.7)$$

The identity (20.7) expresses the fact that one can represent any function as a superposition of elements of the form $w_Q T_Q \psi_{a,b,\theta}$; that is, of ridgelet elements localized near the squares Q . Associated with the function $T_Q \psi_{a,b,\theta}$ is the ridgelet $\psi_{a_Q, \theta_Q, b_Q}$ (20.3) with parameters obeying

$$a_Q = 2^{-s}a, \quad \theta_Q = \theta, \quad b_Q = b + k_1 2^{-s} \cos \theta + k_2 2^{-s} \sin \theta$$

and thus $w_Q T_Q \psi_{a,b,\theta}$ is a windowed ridgelet, supported near the square Q , hence the name *local ridgelet transform*.

The previous paragraph discussed the construction of local ridgelets of fixed length, roughly 2^{-s} (s fixed). Letting the scale s vary defines the multiscale ridgelet dictionary $\{\psi_{a,b,\theta}^Q : s \geq s_0, Q \in \mathcal{Q}_s, a > 0, b \in \mathbf{R}, \theta \in [0, 2\pi)\}$ by

$$\psi_{a,b,\theta}^Q = w_Q T_Q \psi_{a,b,\theta};$$

that is, a whole pyramid of local ridgelets at various lengths and locations. This is, of course, a massively overcomplete representation system and no formula like (20.7) is available for this multiscale ridgelet pyramid, because it is highly overcomplete.

20.3 Digital Ridgelet Transform

So the basic strategy for calculating the continuous ridgelet transform is first to compute the Radon transform $Rf(t, \theta)$ and second, to apply a one-dimensional wavelet transform to the slices $Rf(\cdot, \theta)$.

Several digital ridgelet transforms have been proposed, and we will described three of them in this section, based on different implementations of the Radon transform.

20.3.1 The RectoPolar Ridgelet transform

In this section we develop a digital procedure which is inspired by this viewpoint, and is realizable on n by n numerical arrays.

A fundamental fact about the Radon transform is the projection-slice formula [53]:

$$\hat{f}(\lambda \cos \theta, \lambda \sin \theta) = \int Rf(t, \theta) e^{-i\lambda t} dt.$$

This says that the Radon transform can be obtained by applying the one-dimensional inverse Fourier transform to the two-dimensional Fourier transform restricted to radial lines going through the origin.

This of course suggests that approximate Radon transforms for digital data can be based on discrete fast Fourier transforms. This is a widely used approach, in the literature of medical imaging and synthetic aperture radar imaging, for which the key approximation errors and artifacts have been widely discussed. In outline, one simply does the following, for gridded data $(f(i_1, i_2))$, $0 \leq i_1, i_2 < n - 1$.

1. *2D-FFT*. Compute the two-dimensional FFT of f giving the array $(\hat{f}(k_1, k_2))$, $-n/2 \leq k_1, k_2 \leq n/2 - 1$.
2. *Cartesian to Polar Conversion*. Using an interpolation scheme, substitute the sampled values of the Fourier transform obtained on the square lattice with sampled values of \hat{f} on a polar lattice: that is, on a lattice where the points fall on lines going through the origin.
3. *1D-IFFT*. Compute the one-dimensional IFFT (inverse FFT) on each line, i.e. for each value of the angular parameter.

The use of this strategy in connection with ridgelet transforms has been discussed in the articles [62, 61, 185].

A Polar Sampling Scheme for Digital Data

For our implementation of the Cartesian-to-polar conversion, we have used a pseudo-polar grid, in which the pseudo-radial variable has level sets which are squares rather than circles. Starting with Oppenheim and Mersereau [131] this grid has often been called the *concentric squares* grid in the signal processing literature; in the medical tomography literature it is associated with the *linogram* [68, 69], while in [9] it is called the rectopolar grid; see this last reference for a complete bibliographic treatment. The geometry of the rectopolar grid is illustrated in Figure 20.3. We select $2n$ radial lines in the frequency plane obtained by connecting the origin to the vertices (k_1, k_2) lying on the boundary of the array (k_1, k_2) , i.e. such that k_1 or $k_2 \in \{-n/2, n/2\}$. The polar grid $\xi_{\ell,m}$ (ℓ serves to index a given radial line while the position of the point on that line is indexed by m) that we shall use is the intersection between the set of radial lines and that of Cartesian lines parallel to the axes. To be more specific, the sample points along a radial line \mathcal{L} whose angle with the vertical axis is less than or equal to $\pi/4$ are obtained by intersecting \mathcal{L} with the set of horizontal lines $\{x_2 = k_2, k_2 = -n/2, -n/2 + 1, \dots, n/2\}$. Similarly, the intersection with the vertical lines $\{x_1 = k_1, k_1 = -n/2, -n/2 + 1, \dots, n/2\}$ defines our sample points whenever the angle between \mathcal{L} and the horizontal axis is less than or equal to $\pi/4$. The cardinality of the rectopolar grid is equal to $2n^2$ as there are $2n$ radial lines and n sampled values on each of these lines. As a result, data structures associated with this grid will have a rectangular format. We observe that this choice corresponds to irregularly spaced values of the angular variable θ .

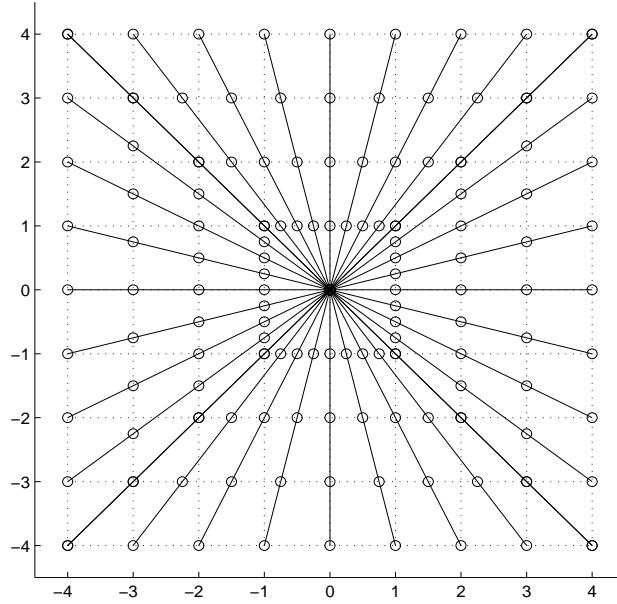


Figure 20.3: Illustration of the digital polar grid in the frequency domain for an n by n image ($n = 8$). The figure displays the set of radial lines joining pairs of symmetric points from the boundary of the square. The rectopolar grid is the set of points – marked with circles – at the intersection between those radial lines and those which are parallel to the axes.

Interpolation to Rectopolar Grid

To obtain samples on the rectopolar grid, we should, in general, interpolate from nearby samples at the Cartesian grid. In principle, cf. [9, 60], the interpolation of Fourier transforms is a very delicate matter because of the well-known fact that the Fourier transform of an image is highly oscillatory, and the phase contains crucial information about the image. In our approach, however, we use a crude interpolation method: we simply impute for $\hat{f}(\xi_{\ell,m})$ the value of the

Fourier transform taken at the point on the Cartesian grid nearest to $\xi_{\ell,m}$.

There are, of course, more sophisticated ways to realize the Cartesian-to-polar conversion; even simple bilinear interpolation would offer better theoretical accuracy. A very high accuracy approach used in [61] consists of viewing the data $(\hat{f}(k_1, k_2))$ as samples of the trigonometric polynomial F defined by

$$F(\omega_1, \omega_2) = \sum_{i_1=0}^{n-1} \sum_{i_2=0}^{n-1} f(i_1, i_2) \exp\{-i(\omega_1 i_1 + \omega_2 i_2)\} \quad (20.8)$$

on a square lattice; that is, with $\hat{f}(k_1, k_2) = F(\frac{2\pi k_1}{n}, \frac{2\pi k_2}{n})$ with $-n/2 \leq k_1, k_2 < n/2$. There turns out [61, 9] to be an exact algorithm for rapidly finding the values of F on the polar grid. The high-accuracy approach can be used in reverse, allowing for exact reconstruction of the original trigonometric polynomial from its rectopolar samples.

Our nearest-neighbor interpolation, although admittedly simple-minded, happens to give good results in our applications. In fact numerical experiments show that in overall system performance, it rivals the exact interpolation scheme. This is explainable as follows. Roughly speaking, the high-frequency terms in the trigonometric polynomial F are associated with pixels at the boundary of the underlying n by n grid. Our crude interpolation evidently will fail at reconstructing high-frequency terms. However, in the curvelet application – see below – we use a window function to downweight the contributions of our reconstruction near the boundary of the image array. So, inaccuracies in reconstruction caused by our crude interpolation can be expected to be located mostly in regions which make little visual impact on the reconstruction.

A final point about our implementation. Since we are interested in noise removal, avoiding artifacts is very important. At the signal-to-noise ratios we consider, high-order-accuracy interpolation formulas which generate substantial artifacts (as many high-order formulas do) can be less useful than low-order-accuracy schemes which are relatively artifact-free. A known artifact of exact interpolation of trigonometric polynomials is as follows: substantial long-range disturbances can be generated by local perturbations such as discontinuities. In this sense, our crude interpolation may actually turn out to be preferable for some purposes.

Exact Reconstruction and Stability

The Cartesian-to-rectopolar conversion we have suggested here is reversible. That is to say, given the rectopolar values output from this method, one can recover the original Cartesian values exactly. To see this, take as given the following: **Claim:** *the assignment of Cartesian points as nearest neighbors of rectopolar points happens in such a way that each Cartesian point is assigned as the nearest neighbor of at least one rectopolar point.* It follows from this claim that each value in the original Cartesian input array is copied into at least one place in the output rectopolar array. Hence, perfect reconstruction is obviously possible in principle – just by keeping track of where the entries are, have been copied to, and undoing the process.

Our reconstruction rule obtains, for each point on the Cartesian grid, the *arithmetic mean of all the values in the rectopolar grid which have that Cartesian point as their nearest point*. This provides a numerically stable left inverse. Indeed, if applied to a perturbed set of rectopolar values, this rule gives an approximate reconstruction of the original unperturbed Cartesian values in which the approximation errors are smaller than the size of the perturbations suffered by the rectopolar values. (This final comment is reassuring in the present de-noising context, where our reconstructions will always be made by perturbing the empirical rectopolar FT of the noisy data.) Phrased in mathematical terms this gives

$$x_C = 1/\#R(C) \sum_{R \in R(C)} y_R,$$

where C is a given point on the Cartesian grid and $R(C)$ is the set of rectopolar points that are closest to C . Cardinality is denoted $\#$. Stability in ℓ_2 , for instance, follows from the observation

$$|x_C|^2 \leq 1/\#R(C) \sum_{R \in R(C)} |y_R|^2 \leq \sum_{R \in R(C)} |y_R|^2;$$

Since we have a partition of the set of rectopolar points, summing this last inequality across the Cartesian grid gives $\|x\|^2 \leq \|y\|^2$.

It remains to explain the italicized claim, because, as we have seen, from it flows the exact reconstruction property and stability of the inverse. Consider the rectopolar points in the hourglass region made of ‘basically vertical lines’, i.e. lines which make an angle less than $\pi/4$ with vertical, and more specifically those points on a single horizontal scan line. Assuming the scan line is not at the extreme top or bottom of the array, these points are spaced *strictly less than one unit apart*, where our unit is the spacing of the Cartesian grid. Therefore, when we consider a Cartesian grid point C belonging to this scan line and ask about the rectopolar points R_L and R_R which are closest to it on the left and right respectively, these two points cannot be as much as 1 unit apart: $\|R_L - R_R\| < 1$. Therefore at least one of the two points must be strictly less than $1/2$ unit away from the Cartesian point: i.e. either $\|R_L - C\| < 1/2$ or $\|R_R - C\| < 1/2$. Without loss of generality suppose that $\|R_L - C\| < 1/2$. Then clearly R_L has C as its closest Cartesian point. In short, every Cartesian point in the strict interior of the ‘hourglass’ associated with the ‘basically vertical’ lines arises as the strict closest Cartesian point of at least one rectopolar point. Similar statements can be made about points on the boundary of the hourglass, although the arguments supporting those statements are much simpler, essentially mere inspection. Similar statements can be made about the points in the transposed hourglass. The italicized claim is established.

One-dimensional Wavelet Transform

To complete the ridgelet transform, we must take a one-dimensional wavelet transform along the radial variable in Radon space. We now discuss the choice of digital one-dimensional wavelet transform.

Experience has shown that compactly-supported wavelets can lead to many visual artifacts when used in conjunction with nonlinear processing – such as hard-thresholding of individual wavelet coefficients – particularly for decimated wavelet schemes used at critical sampling. Also, because of the lack of localization of such compactly-supported wavelets in the frequency domain, fluctuations in coarse-scale wavelet coefficients can introduce fine-scale fluctuations; this is undesirable in our setting. Here we take a frequency-domain approach, where the discrete Fourier transform is reconstructed from the inverse Radon transform. These considerations lead us to use a band-limited wavelet, whose support is compact in the Fourier domain rather than the time domain. Other implementations have made a choice of compact support in the frequency domain as well [62, 61]. However, we have chosen a specific overcomplete system, based on work of Starck et al. [183, 191], who constructed such a wavelet transform and applied it to interferometric image reconstruction. The wavelet transform algorithm is based on a scaling function ϕ such that $\hat{\phi}$ vanishes outside of the interval $[-\nu_c, \nu_c]$. We defined the scaling function $\hat{\phi}$ as a renormalized B_3 -spline

$$\hat{\phi}(\nu) = \frac{3}{2}B_3(4\nu),$$

and $\hat{\psi}$ as the difference between two consecutive resolutions

$$\hat{\psi}(2\nu) = \hat{\phi}(\nu) - \hat{\phi}(2\nu).$$

Because $\hat{\psi}$ is compactly supported, the sampling theorem shows than one can easily build a pyramid of $n + n/2 + \dots + 1 = 2n$ elements: see [191] for details.

This transform enjoys the following features:

- The wavelet coefficients are directly calculated in the Fourier space. In the context of the ridgelet transform, this allows avoiding the computation of the one-dimensional inverse Fourier transform along each radial line.
- Each subband is sampled above the Nyquist rate, hence avoiding aliasing – a phenomenon typically encountered by critically sampled orthogonal wavelet transforms [170].

- The reconstruction is trivial. The wavelet coefficients simply need to be co-added to reconstruct the input signal at any given point. In our application, this implies that the ridgelet coefficients simply need to be co-added to reconstruct Fourier coefficients.

This wavelet transform introduces an extra redundancy factor, which might be viewed as an objection by advocates of orthogonality and critical sampling. However, we note that our goal in this implementation is not data compression/efficient coding – for which critical sampling might be relevant – but instead noise removal, for which it is well-known that overcompleteness can provide substantial advantages [43].

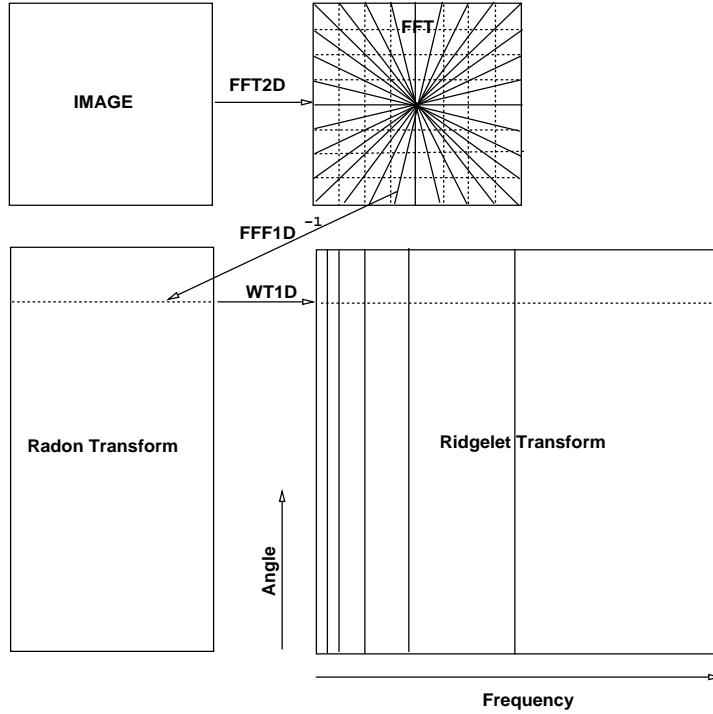


Figure 20.4: Ridgelet transform flowgraph. Each of the $2n$ radial lines in the Fourier domain is processed separately. The 1D inverse FFT is calculated along each radial line followed by a 1D nonorthogonal wavelet transform. In practice, the one-dimensional wavelet coefficients are directly calculated in the Fourier space.

Combining the Pieces

Figure 20.4 shows the flowgraph of the ridgelet transform. The ridgelet transform of an image of size $n \times n$ is an image of size $2n \times 2n$, introducing a redundancy factor equal to 4.

We note that, because our transform is made up of a chain of steps, each one of which is invertible, the whole transform is invertible, and so has the exact reconstruction property. For the same reason, the reconstruction is stable under perturbations of the coefficients.

Last but not least, our discrete transform is computationally attractive. Indeed, the algorithm we presented here has low complexity since it runs in $\tilde{O}(n^2 \log(n))$ flops for an $n \times n$ image.

20.3.2 The Orthonormal Finite Ridgelet Transform

The orthonormal finite ridgelet transform (OFRT) has been recently proposed [59, 58] for image compression and filtering. The transform, based on the finite Radon transform [130] and a 1D

orthogonal wavelet transform, is not redundant, and reversible. It would be a great alternative to the previously described ridgelet transform if the OFRT were not based on a strange definition of a line. In fact, a line in the OFRT is defined as a set of periodic equidistant points [130]. Figure 20.5 shows the backprojection of a ridgelet coefficient by the FFT-based ridgelet transform (left) and by the OFRT (right). It is clear that the backprojection of the OFRT is nothing like a ridge function.

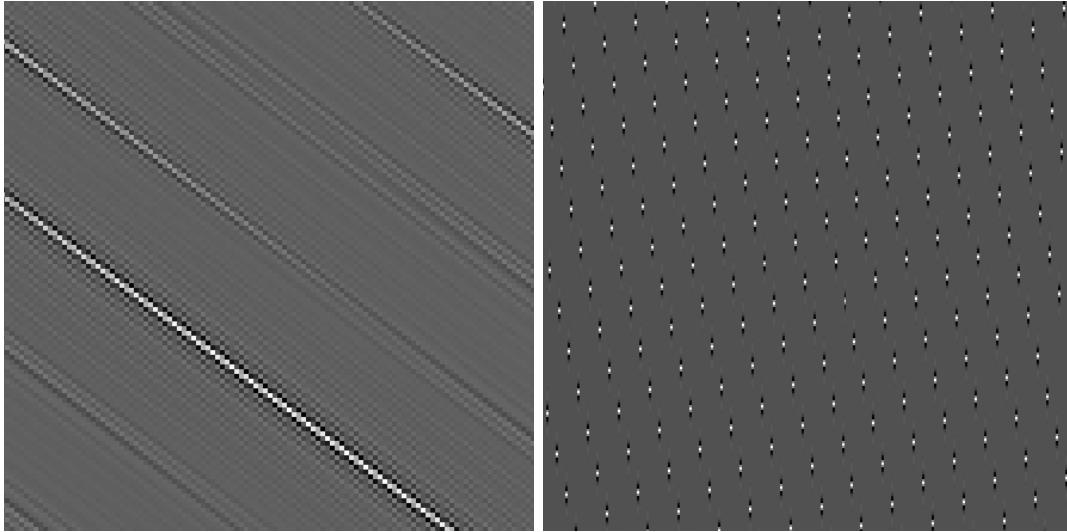


Figure 20.5: Left, backprojection of a ridgelet coefficient by the FFT-based ridgelet transform, and right, backprojection of a finite ridgelet coefficient.



Figure 20.6: Left, part of Lena image, and right, reconstruction after finite ridgelet coefficient thresholding.

Because of this specific definition of a line, the thresholding of the OFRT coefficients produces strong artifacts. Figure 20.6 left shows a part of the original standard Lena image, and Figure 20.6 right shows the reconstruction after the hard thresholding of the OFRT. A kind of noise has been added to the noise-free image! Finally, the OFRT presents another limitation: the image size must be a prime number. This last point is however not too restrictive, because

we generally use a partitioning when denoising the data, and a prime number block size can be used. The OFTR is interesting from the conceptual point of view, but will certainly be of no help for real applications.

20.3.3 The Slant Stack Ridgelet Transform

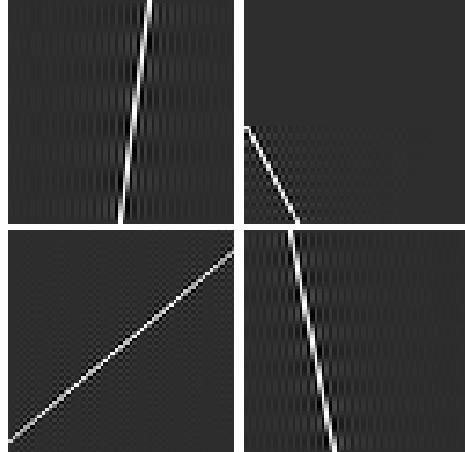


Figure 20.7: Backprojection of a point at four different locations in the Radon space.

The Fast Slant Stack [9] is geometrically more accurate than the previously described methods. The backprojection of a point in Radon space is exactly a ridge function in the spatial domain (see Figure 20.7). The transformation of an $n \times n$ image is a $2n \times 2n$ image. n line integrals with angle between $[-\frac{\pi}{4}, \frac{\pi}{4}]$ are calculated from the zero padded image on the y-axis, and n line integrals with angle between $[\frac{\pi}{4}, \frac{3\pi}{4}]$ are computed by zero padding the image on the x-axis. For a given angle inside $[-\frac{\pi}{4}, \frac{\pi}{4}]$, $2n$ line integrals are calculated by first shearing the zero-padded image, and then integrating the pixel values along all horizontal lines (resp. vertical lines for angles in $[\frac{\pi}{4}, \frac{3\pi}{4}]$). The shearing is performed column per column (resp. line per line) by using the 1D FFT. Figure 20.8 shows an example of the image shearing step with two different angles ($5\frac{\pi}{4}$ and $-\frac{\pi}{4}$). A ridgelet transform based on the Fast Slant Stack transform has been proposed in [60]. The connection between the Fast Slant Stack and the linogram has been investigated in [9], and a Fast Slant Stack is proposed, based on the 2D Fourier transform.

20.4 Local Ridgelet Transforms

A digital version of the ideas presented in section 20.2.2 decomposes the original n by n image into smoothly overlapping blocks of sidelength b pixels in such a way that the overlap between two vertically adjacent blocks is a rectangular array of size b by $b/2$; we use overlap to avoid blocking artifacts. For an n by n image, we count $2n/b$ such blocks in each direction.

The partitioning introduces redundancy, as a pixel belongs to 4 neighboring blocks. We present two competing strategies to perform the analysis and synthesis:

1. The block values are weighted (analysis) in such a way that the co-addition of all blocks reproduce exactly the original pixel value (synthesis).
2. The block values are those of the image pixel values (analysis) but are weighted when the image is reconstructed (synthesis).

Of course, there are intermediate strategies and one could apply smooth windowing at both the analysis and synthesis stages as discussed in Section 20.2.2, for example. In the first approach, the data are smoothly windowed and this presents the advantage to limit the analysis artifacts

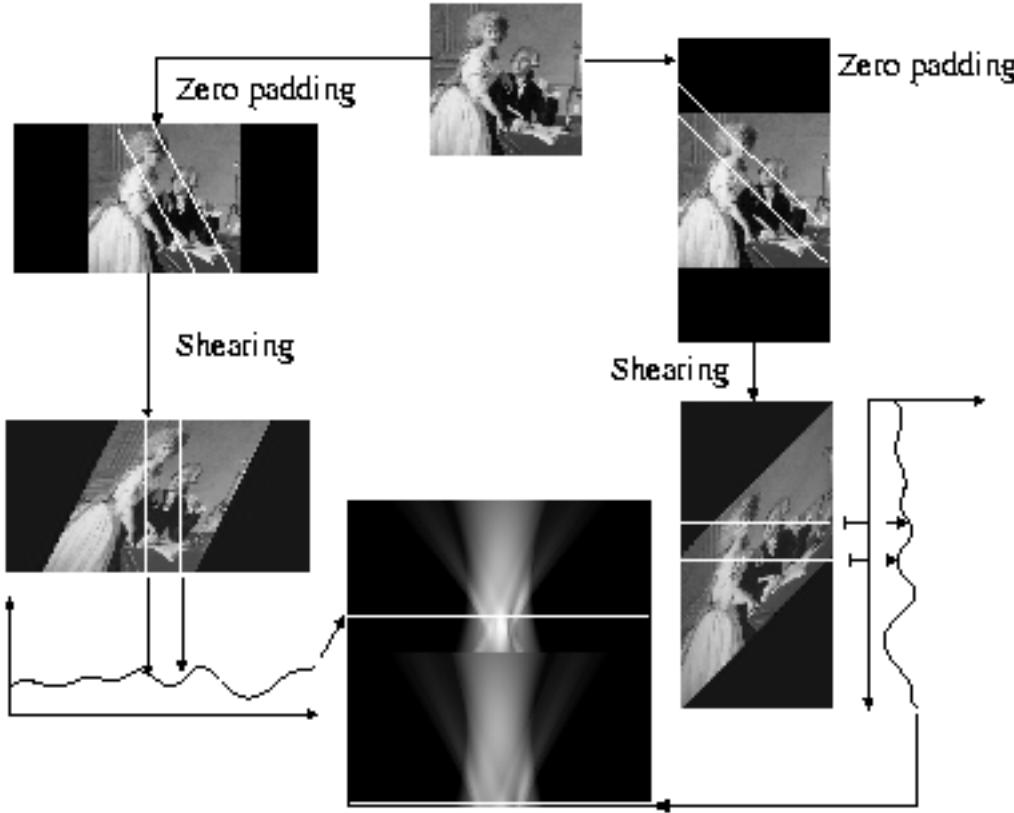


Figure 20.8: Slant Stack Transform of an image.

traditionally associated with boundaries. The drawback, however, is a loss of sensitivity. Indeed, suppose for the sake of simplicity that a vertical line with intensity level L intersects a given block of size b . Without loss of generality assume that the noise standard deviation is equal to 1. When the angular parameter of the Radon transform coincides with that of the line, we obtain a measurement with a signal intensity equal to bL while the noise standard deviation is equal to \sqrt{b} (in this case, the value of the signal-to-noise ratio (SNR) is $\sqrt{b}L$). If weights are applied at the analysis stage, the SNR is roughly equal to $L \sum_{i=1}^b w_i / \sqrt{\sum_{i=1}^b w_i^2} < \sqrt{b}L$. Experiments have shown that this sensitivity loss may have substantial effects in filtering applications and, therefore, the second approach seems more appropriate since our goal is image restoration.

We calculate a pixel value $f(i, j)$ from its four corresponding block values of half-size $\ell = b/2$, namely, $B_1(i_1, j_1)$, $B_2(i_2, j_1)$, $B_3(i_1, j_2)$ and $B_4(i_2, j_2)$ with $i_1, j_1 > b/2$ and $i_2 = i_1 - \ell, j_2 = j_1 - \ell$, in the following way:

$$\begin{aligned} f_1 &= w(i_2/\ell)B_1(i_1, j_1) + w(1 - i_2/\ell)B_2(i_2, j_1) \\ f_2 &= w(i_2/\ell)B_3(i_1, j_2) + w(1 - i_2/\ell)B_4(i_2, j_2) \\ f(i, j) &= w(j_2/\ell)f_1 + w(1 - j_2/\ell)f_2 \end{aligned} \quad (20.9)$$

with $w(x) = \cos^2(\pi x/2)$. Of course, one might select any other smooth, non-increasing function satisfying $w(0) = 1$, $w(1) = 0$, $w'(0) = 0$ and obeying the symmetry property $w(x) + w(1 - x) = 1$.

It is worth mentioning that the spatial partitioning introduces a redundancy factor equal to 4.

Finally, we note that in order to be in better agreement with the theory one should of course

introduce a normalizing factor depending on the block-size. However, since we are concerned about de-noising and the thresholding of individual coefficients, the normalization is a non-issue. Renormalizing coefficients automatically renormalizes corresponding thresholds in the exact same way: see section 21.

20.5 Digital Curvelet Transform

20.5.1 Discrete Curvelet Transform of Continuum Functions

We now briefly return to the continuum viewpoint of section 20.2.2. Suppose we set an initial goal to produce a decomposition using the multiscale ridgelet pyramid. The hope is that this would allow us to use thin ‘brushstrokes’ to reconstruct the image, with all lengths and widths available to us. In particular, this would seem allow us to trace sharp edges precisely using a few elongated elements with very narrow widths.

As mentioned in section 20.2.2, the full multiscale ridgelet pyramid is highly overcomplete. As a consequence, convenient algorithms like simple thresholding will not find sparse decompositions when such good decompositions exist. An important ingredient of the curvelet transform is to restore sparsity by reducing redundancy across scales. In detail, one introduces interscale orthogonality by means of subband filtering. Roughly speaking, different levels of the multiscale ridgelet pyramid are used to represent different subbands of a filter bank output. At the same time, this subband decomposition imposes a relationship between the width and length of the important frame elements so that they are anisotropic and obey $\text{width} = \text{length}^2$.

The discrete curvelet transform of a continuum function $f(x_1, x_2)$ makes use of a dyadic sequence of scales, and a bank of filters $(P_0f, \Delta_1f, \Delta_2f, \dots)$ with the property that the passband filter Δ_s is concentrated near the frequencies $[2^{2s}, 2^{2s+2}]$, e.g.

$$\Delta_s = \Psi_{2s} * f, \quad \widehat{\Psi_{2s}}(\xi) = \widehat{\Psi}(2^{-2s}\xi).$$

In wavelet theory, one uses a decomposition into dyadic subbands $[2^s, 2^{s+1}]$. In contrast, the subbands used in the discrete curvelet transform of continuum functions have the non-standard form $[2^{2s}, 2^{2s+2}]$. This is a non-standard feature of the discrete curvelet transform well worth remembering.

With the notation of section 20.2.2, the curvelet decomposition is the sequence of the following steps:

- *Subband Decomposition.* The object f is decomposed into subbands:

$$f \mapsto (P_0f, \Delta_1f, \Delta_2f, \dots).$$

- *Smooth Partitioning.* Each subband is smoothly windowed into ‘squares’ of an appropriate scale (of sidelength $\sim 2^{-s}$):

$$\Delta_s f \mapsto (w_Q \Delta_s f)_{Q \in \mathcal{Q}_s}.$$

- *Renormalization.* Each resulting square is renormalized to unit scale

$$g_Q = (T_Q)^{-1}(w_Q \Delta_s f), \quad Q \in \mathcal{Q}_s. \tag{20.10}$$

- *Ridgelet Analysis.* Each square is analyzed via the discrete ridgelet transform.

In this definition, the two dyadic subbands $[2^{2s}, 2^{2s+1}]$ and $[2^{2s+1}, 2^{2s+2}]$ are merged before applying the ridgelet transform.

20.5.2 Digital Realization

In developing a transform for digital n by n data which is analogous to the discrete curvelet transform of a continuous function $f(x_1, x_2)$, we replace each of the continuum concepts with the appropriate digital concept mentioned in the sections above. In general, the translation is rather obvious and direct. However, experience shows that one modification is essential; we found that, rather than merging the two dyadic subbands $[2^{2s}, 2^{2s+1}]$ and $[2^{2s+1}, 2^{2s+2}]$ as in the theoretical work, in the digital application, leaving these subbands separate, applying spatial partitioning to each subband and applying the ridgelet transform on each subband separately led to improved visual and numerical results.

We believe that the “à trous” subband filtering algorithm is especially well-adapted to the needs of the digital curvelet transform. The algorithm decomposes an n by n image I as a superposition of the form

$$I(x, y) = c_J(x, y) + \sum_{j=1}^J w_j(x, y),$$

where c_J is a coarse or smooth version of the original image I and w_j represents “the details of I ” at scale 2^{-j} : see [191] for more information. Thus, the algorithm outputs $J + 1$ subband arrays of size $n \times n$. (The indexing is such that, here, $j = 1$ corresponds to the finest scale, or high frequencies.)

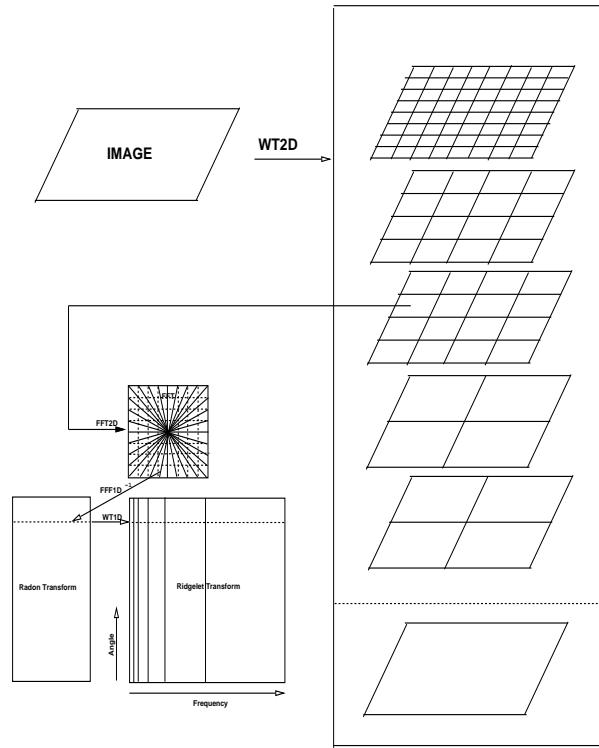


Figure 20.9: Curvelet transform flowgraph. The figure illustrates the decomposition of the original image into subbands followed by the spatial partitioning of each subband. The ridgelet transform is then applied to each block.

20.5.3 Algorithm

We now present a sketch of the discrete curvelet transform algorithm:

1. Apply the à trous algorithm with J scales,
2. set $B_1 = B_{min}$,
3. for $j = 1, \dots, J$ do,
 - (a) partition the subband w_j with a block size B_j and apply the digital ridgelet transform to each block,
 - (b) if j modulo 2 = 1 then $B_{j+1} = 2B_j$,
 - (c) else $B_{j+1} = B_j$.

The sidelength of the localizing windows is doubled *at every other* dyadic subband, hence maintaining the fundamental property of the curvelet transform which says that elements of length about $2^{-j/2}$ serve for the analysis and synthesis of the j -th subband $[2^j, 2^{j+1}]$. Note also that the coarse description of the image c_J is not processed. Figure 20.9 gives an overview of the organization of the algorithm.

This implementation of the curvelet transform is also redundant. The redundancy factor is equal to $16J+1$ whenever J scales are employed. Finally, the method enjoys exact reconstruction and stability, because this invertibility holds for each element of the processing chain.

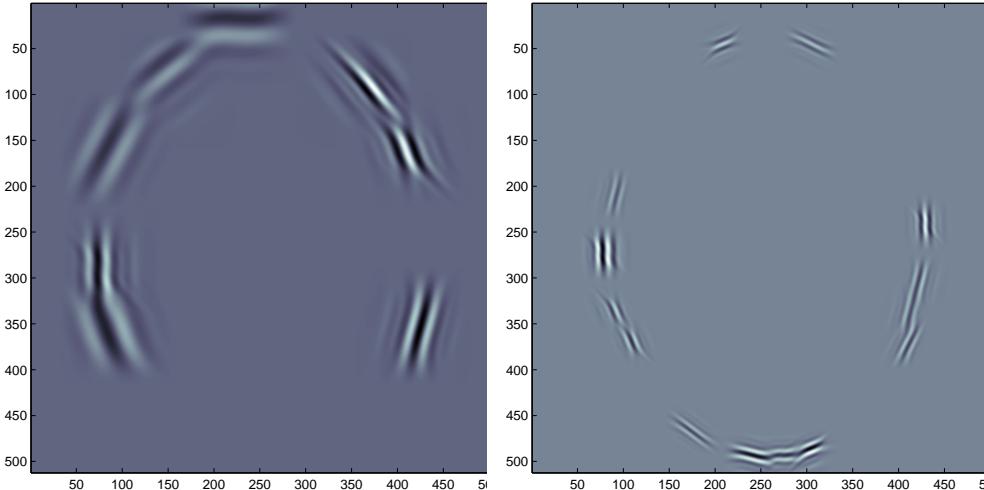


Figure 20.10: A few curvelets.

Figure 20.10 shows a few curvelets at different scales, orientations and locations.

Chapter 21

Filtering

21.1 Curvelet Coefficient Thresholding

We now apply our digital transforms for removing noise from image data. The methodology is standard and is outlined mainly for the sake of clarity and self-containedness.

Suppose that one is given noisy data of the form

$$x_{i,j} = f(i,j) + \sigma z_{i,j},$$

where f is the image to be recovered and z is white noise, i.e. $z_{i,j} \stackrel{i.i.d.}{\sim} N(0, 1)$. Unlike FFTs or FWTs, our discrete ridgelet (resp. curvelet) transform is not norm-preserving and, therefore, the variance of the noisy ridgelet (resp. curvelet) coefficients will depend on the ridgelet (resp. curvelet) index λ . For instance, letting F denote the discrete curvelet transform matrix, we have $Fz \stackrel{i.i.d.}{\sim} N(0, FF^T)$ where T denotes transpose. Because the computation of FF^T is prohibitively expensive, we calculated an approximate value $\tilde{\sigma}_\lambda^2$ of the individual variances using Monte Carlo simulations where the diagonal elements of FF^T are simply estimated by evaluating the curvelet transforms of a few standard white noise images.

Let y_λ be the noisy curvelet coefficients ($y = Fx$). We use the following hard-thresholding rule for estimating the unknown curvelet coefficients:

$$\hat{y}_\lambda = y_\lambda \quad \text{if} \quad |y_\lambda|/\sigma \geq k\tilde{\sigma}_\lambda \tag{21.1}$$

$$\hat{y}_\lambda = 0 \quad \text{if} \quad |y_\lambda|/\sigma < k\tilde{\sigma}_\lambda. \tag{21.2}$$

In our experiments, we actually chose a scale-dependent value for k ; we have $k = 4$ for the first scale ($j = 1$) while $k = 3$ for the others ($j > 1$).

Poisson Observations

Assume now that we have Poisson data $x_{i,j}$ with unknown mean $f(i,j)$. The Anscombe transformation [6]

$$\tilde{x} = 2\sqrt{x + \frac{3}{8}} \tag{21.3}$$

stabilizes the variance and we have $\tilde{x} = 2\sqrt{f} + \epsilon$ where ϵ is a vector with independent and approximately standard normal components. In practice, this is a good approximation whenever the number of counts is large enough, greater than 30 per pixel, say.

For a small number of counts, a possibility is to compute the Radon transform of the image, and then to apply the Anscombe transformation to the Radon data. The rationale is that, roughly speaking, the Radon transform corresponds to a summation of pixel values over lines and the sum of independent Poisson random variables is a Poisson random variable with intensity equal to the sum of the individual intensities. Hence, the intensity of the sum may

be quite large (hence validating the Gaussian approximation) even though the individual intensities may be small. This might be viewed as an interesting feature since, unlike wavelet transforms, the ridgelet and curvelet transforms tend to average data over elongated and rather large neighborhoods.

21.2 Filtering Experiments

Recovery of Linear Features

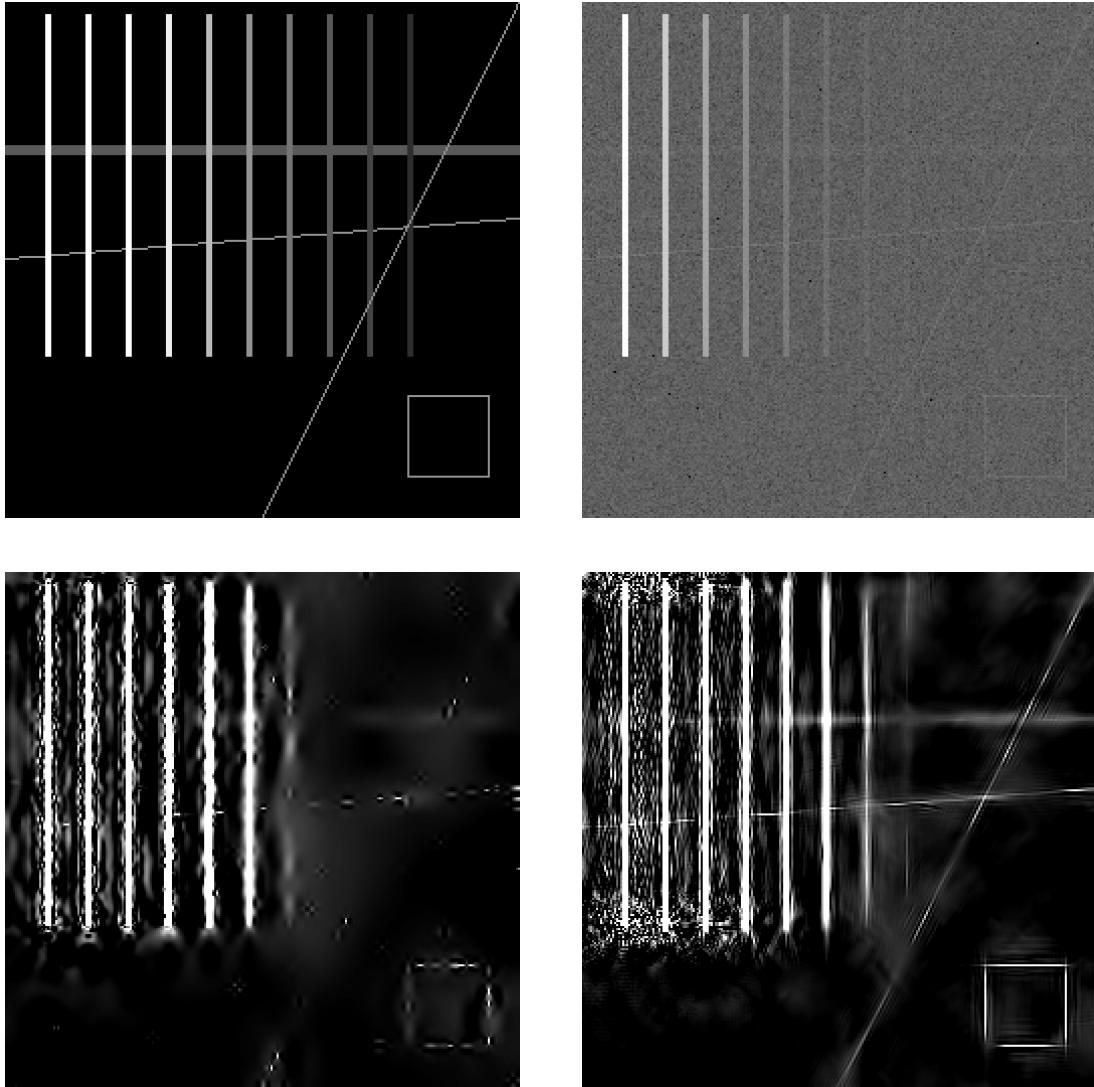


Figure 21.1: The top panels display a geometric image and that same image contaminated with Gaussian white noise. The bottom left and right panels display the restored images using the undecimated wavelet transform and the curvelet transform, respectively.

The next experiment (Figure 21.1) consists of an artificial image containing a few bars, lines and a square. The intensity is constant along each individual bar; from left to right, the intensities of the ten vertical bars (these are in fact thin rectangles which are 4 pixels wide and

170 pixels long) are equal to $\frac{32}{2^i}$, $i = 0, \dots, 9$. The intensity along all the other lines is equal to 1, and the noise standard deviation is 1/2. Displayed images have been log-transformed in order to better see the results at low signal to noise ratio.

The curvelet reconstruction of the nonvertical lines is obviously sharper than that obtained using wavelets. The curvelet transform also seems to go one step further as far as the reconstruction of the vertical lines is concerned. Roughly speaking, for those templates, the wavelet transform stops detecting signal at a SNR equal to 1 (we defined here the SNR as the intensity level of the pixels on the line, divided by the noise standard deviation of the noise) while the cut-off value equals 0.5 for the curvelet approach. It is important to note that the horizontal and vertical lines correspond to privileged directions for the wavelet transform, because the underlying basis functions are direct products of functions varying solely in the horizontal and vertical directions. Wavelet methods will give even poorer results on lines of the same intensity but tilting substantially away from the Cartesian axes. Cf. the reconstructions of the faint diagonal lines in the image.

Recovery of Curves

In this experiment (Figure 21.2), we have added Gaussian noise to “War and Peace,” a drawing from Picasso which contains many curved features. Figure 21.2 bottom left and right show respectively the restored images by the undecimated wavelet transform and the curvelet transform. Curves are more sharply recovered with the curvelet transform.

The authors are working on new methods (some of which will be based on the curvelet transform) to extract and recover curves from noisy data with greater accuracy and, therefore, this example is merely to be taken for illustrative purposes.

Denoising of a Color Image

In a wavelet based denoising scenario, color RGB images are generally mapped into the YUV space, and each YUV band is then filtered independently from the others. The goal here is to see whether the curvelet transform would give improved results. We used four of the classical color images, namely **Lena**, **Peppers**, **Baboon**, and **Barbara** (all images except perhaps **Barbara** are available from the USC-SIPI Image Database [50]). We performed a series of experiments and summarized our findings in Figure 21.3 which again displays the PSNR (peak signal-to-noise ratio) versus the noise standard deviation for the four images.

In all cases, the curvelet transform outperforms the wavelet transforms in terms of PSNR – at least for moderate and large values of the noise level. In addition, the curvelet transform outputs images that are visually more pleasant.

Saturn Rings

Gaussian white noise with a standard deviation fixed to 20 was added to the **Saturn** image. We employed several methods to filter the noisy image:

1. Thresholding of the Curvelet transform.
2. Bi-orthogonal undecimated wavelet de-noising methods using the Daubechies-Antonini 7/9 filters (FWT-7/9) and hard thresholding.
3. A trous wavelet transform algorithm and hard thresholding.

Our experiments are reported in Figure 21.4. The curvelet reconstruction does not contain the quantity of disturbing artifacts along edges that one sees in wavelet reconstructions. An examination of the details of the restored images is instructive. One notices that the decimated wavelet transform exhibits distortions of the boundaries and suffers substantial loss of important detail. The à trous wavelet transform gives better boundaries, but completely omits to reconstruct certain ridges. In addition, it exhibits numerous small-scale embedded blemishes; setting higher thresholds to avoid these blemishes would cause even more of the intrinsic structure to be missed.

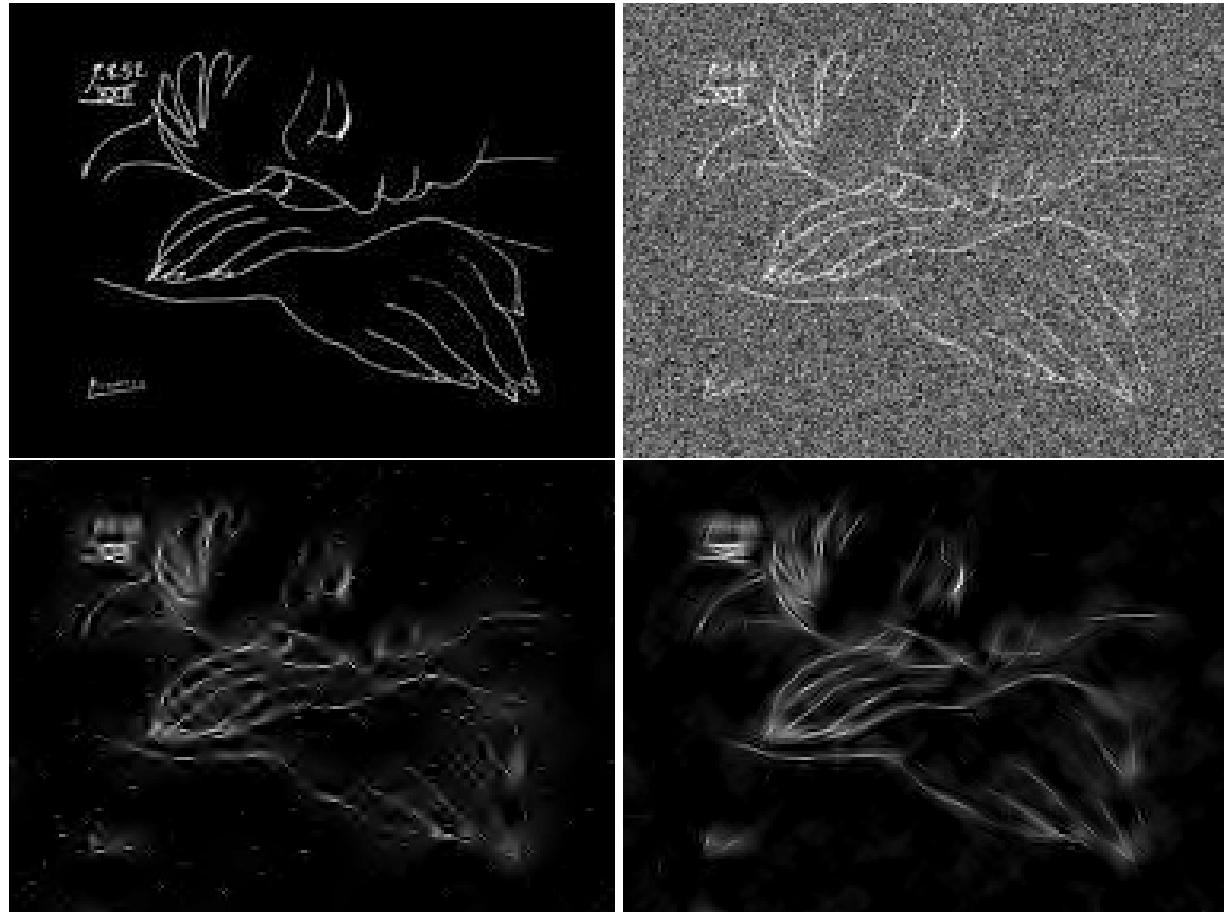


Figure 21.2: The top panels display a Picasso picture (*War and Peace*) and that same image contaminated with Gaussian white noise. The bottom left and right panels display the restored images using the undecimated wavelet transform and the curvelet transform respectively.

Supernova with Poisson noise

Figure 21.5 shows an example of an X-ray image filtering by the ridgelet transform using such an approach. Figure 21.5 left and right show respectively the XMM/Newton image of the Kepler SN1604 supernova and the ridgelet filtered image (using a five sigma hard thresholding).

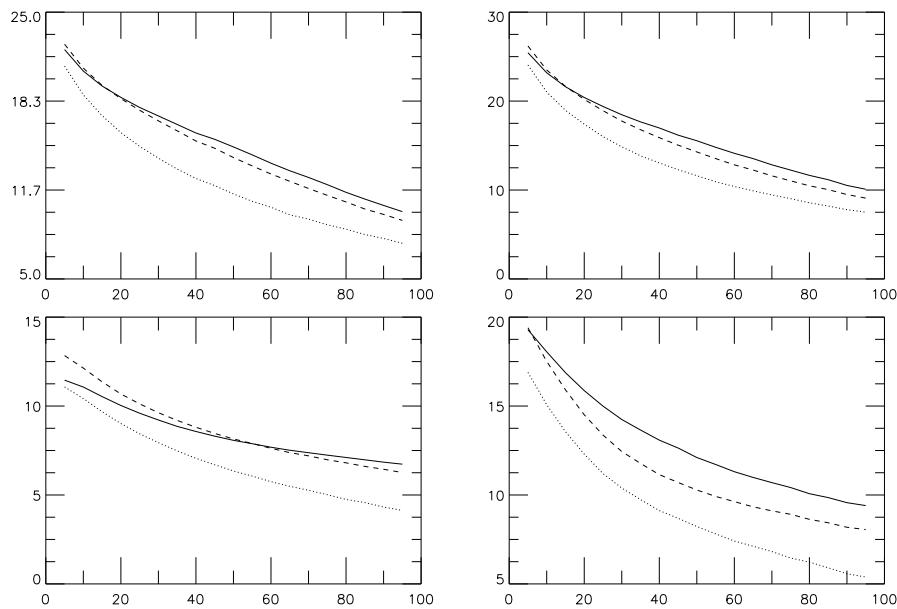


Figure 21.3: PSNR versus noise standard deviation using different filtering methods. YUV and curvelet, YUV and undecimated wavelet, and YUV and decimated wavelet transforms are represented respectively with a continuous, dashed, and dotted line. The upper left panel corresponds to **Lena** (RGB), the upper right to **pepper** (RGB), the bottom left to **Baboon** (RGB), and the bottom right to **Barbara** (RGB).

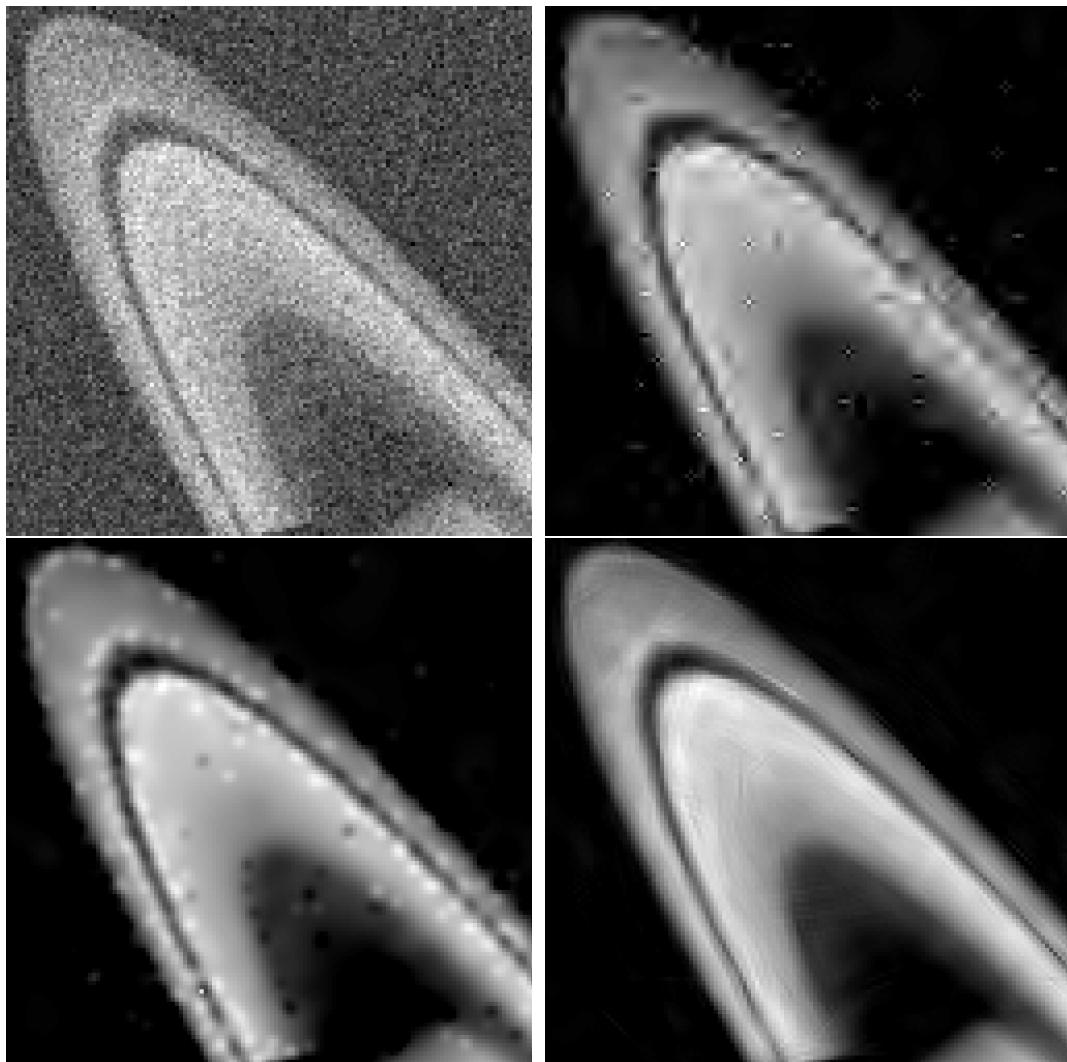


Figure 21.4: Top left, part of Saturn image with Gaussian noise. Top right, filtered image using the undecimated bi-orthogonal wavelet transform. Bottom left and right, filtered image by the à trous wavelet transform algorithm and the curvelet transform.

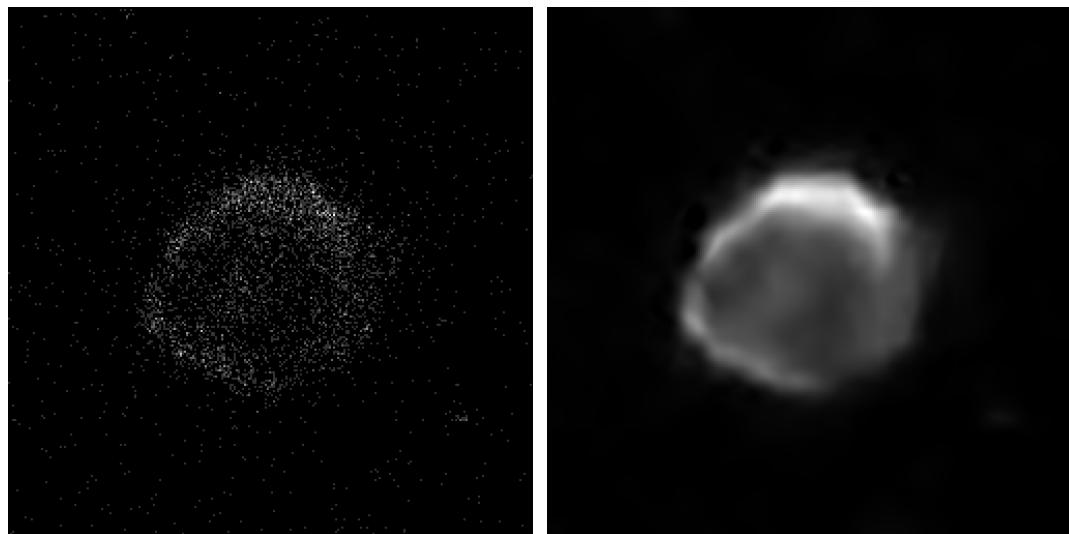


Figure 21.5: Left, XMM/Newton image of the Kepler SN1604 supernova. Right, ridgelet filtered image.

Chapter 22

The Combined Filtering Method

22.1 Introduction

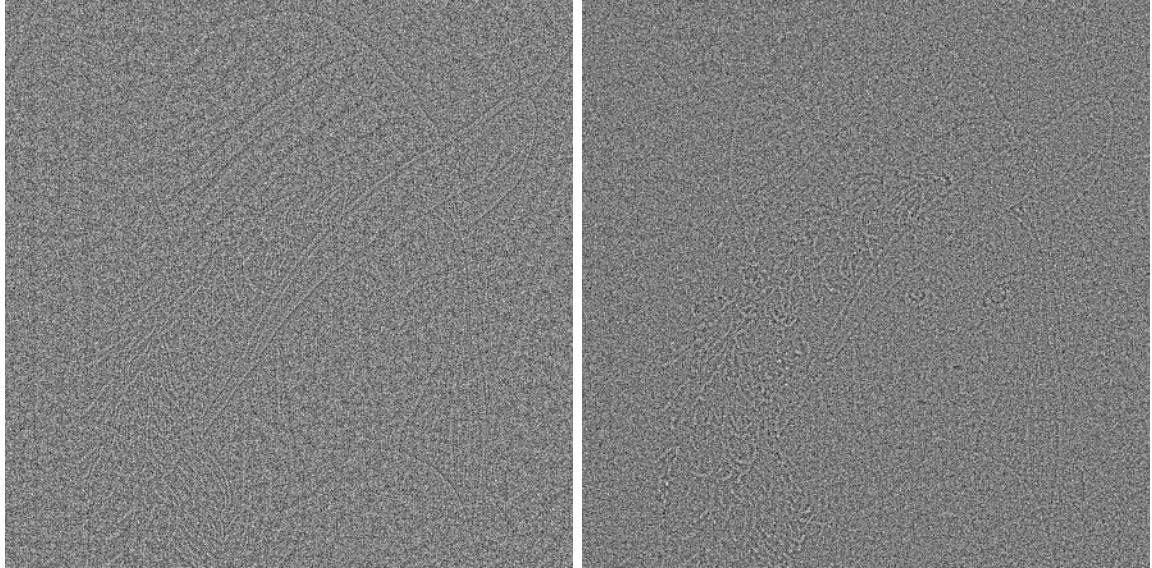


Figure 22.1: Residual for thresholding of the undecimated wavelet transform and thresholding of the curvelet transform.

Although the results obtained by simply thresholding the curvelet expansion are encouraging, there is of course ample room for further improvement. A quick inspection of the residual images for both the wavelet and curvelet transforms shown in Figure 22.1 reveals the existence of very different features. For instance, wavelets do not restore long edges with high fidelity while curvelets are seriously challenged by small features such as Lena's eyes. Loosely speaking, each transform has its own area of expertise and this complementarity may be of great potential. This section will develop a denoising strategy based on the idea of combining both transforms.

In general, suppose that we are given K linear transforms T_1, \dots, T_K and let α_k be the coefficient sequence of an object x after applying the transform T_k , i.e. $\alpha_k = T_k x$. We will suppose that for each transform T_k we have available a reconstruction rule that we will denote by T_k^{-1} although this is clearly an abuse of notation. Finally, T will denote the block diagonal matrix with the T_k 's as building blocks and α the amalgamation of the α_k 's.

A hard thresholding rule associated with the transform T_k synthesizes an estimate \tilde{s}_k via the formula

$$\tilde{s}_k = T_k^{-1} \delta(\alpha_k) \quad (22.1)$$

where δ is a rule that sets to zero all the coordinates of α_k whose absolute value falls below a given sequence of thresholds (such coordinates are said to be non-significant).

In practice, a widely used approach is to compute the average of the \tilde{s}_k 's giving a reconstruction of the form

$$\tilde{s} = \sum_k \tilde{s}_k / K. \quad (22.2)$$

For instance, in the literature of image processing it is common to average reconstructions obtained after thresholding the wavelet coefficients of translated versions of the original dataset (cycle-spinning), i.e. the T_k 's are obtained by composing translations and the wavelet transform. In our setup, we do not find this solution very appealing since this creates the opportunity to average high-quality and low-quality reconstructions.

22.2 The Combined Filtering Principle

Given data y of the form $y = s + \sigma z$, where s is the image we wish to recover and z is standard white noise, we propose solving the following optimization problem [186]:

$$\min \|T\tilde{s}\|_{\ell_1}, \quad \text{subject to } s \in C, \quad (22.3)$$

where C is the set of vectors \tilde{s} which obey the linear constraints

$$\begin{cases} \tilde{s} \geq 0, \\ |T\tilde{s} - Ty| \leq e; \end{cases} \quad (22.4)$$

here, the second inequality constraint only concerns the set of significant coefficients, i.e. those indices μ such that $\alpha_\mu = (Ty)_\mu$ exceeds (in absolute value) a threshold t_μ . Given a vector of tolerance (e_μ) , we seek a solution whose coefficients $(T\tilde{s})_\mu$ are within e_μ of the noisy empirical α_μ 's. Think of α_μ as being given by

$$y = \langle y, \varphi_\mu \rangle,$$

so that α_μ is normally distributed with mean $\langle f, \varphi_\mu \rangle$ and variance $\sigma_\mu^2 = \sigma^2 \|\varphi_\mu\|_2^2$. In practice, the threshold values range typically between three and four times the noise level σ_μ and in our experiments we will put $e_\mu = \sigma_\mu/2$. In short, our constraints guarantee that the reconstruction will take into account any pattern which is detected as significant by any of the K transforms.

We use an ℓ_1 penalty on the coefficient sequence because we are interested in *low complexity* reconstructions. There are other possible choices of complexity penalties; for instance, an alternative to (22.3) would be

$$\min \|\tilde{s}\|_{TV}, \quad \text{subject to } s \in C.$$

where $\|\cdot\|_{TV}$ is the Total Variation norm, i.e. the discrete equivalent of the integral of the Euclidean norm of the gradient.

22.3 The Minimization Method

We propose solving (22.3) using the method of hybrid steepest descent (HSD) [216]. HSD consists of building the sequence

$$s^{n+1} = P(s^n) - \lambda_{n+1} \nabla_J(P(s^n)); \quad (22.5)$$

Here, P is the ℓ_2 projection operator onto the feasible set C , ∇_J is the gradient of equation 22.3, and $(\lambda_n)_{n \geq 1}$ is a sequence obeying $(\lambda_n)_{n \geq 1} \in [0, 1]$ and $\lim_{n \rightarrow +\infty} \lambda_n = 0$.

Unfortunately, the projection operator P is not easily determined and in practice we will use the following proxy; compute $T\tilde{s}$ and replace those coefficients which do not obey the constraints $|T\tilde{s} - Ty| \leq e$ (those which fall outside of the prescribed interval) by those of y ; apply the inverse transform.

The combined filtering algorithm is:

1. Initialize $L_{\max} = 1$, the number of iterations N_i , and $\delta_\lambda = \frac{L_{\max}}{N_i}$.
2. Estimate the noise standard deviation σ , and set $e_k = \frac{\sigma}{2}$.
3. For $k = 1, \dots, K$ calculate the transform: $\alpha_k^{(s)} = T_k s$.
4. Set $\lambda = L_{\max}$, $n = 0$, and \tilde{s}^n to 0.
5. While $\lambda >= 0$ do
 - $u = \tilde{s}^n$.
 - For $k = 1, \dots, K$ do
 - Calculate the transform $\alpha_k = T_k u$.
 - For all coefficients $\alpha_{k,l}$ do
 - * Calculate the residual $r_{k,l} = \alpha_{k,l}^{(s)} - \alpha_{k,l}$
 - * if $\alpha_{k,l}^{(s)}$ is significant and $|r_{k,l}| > e_{k,l}$ then $\alpha_{k,l} = \alpha_{k,l}^{(s)}$
 - * $\alpha_{k,l} = \text{sgn}(\alpha_{k,l})(|\alpha_{k,l}| - \lambda)_+$.
 - $u = T_k^{-1} \alpha_k$
 - Threshold negative values in u and $\tilde{s}^{n+1} = u$.
 - $n = n + 1$, $\lambda = \lambda - \delta_\lambda$, and goto 5.

22.4 Experiments

Method	PSNR	Comments
Noisy image	22.13	
OWT7-9 + k-sigma Hard thresh.	28.35	many artifacts
UWT7-9 + k-sigma Hard thresh.	31.94	very few artifacts
Curvelet (B=16)	31.95	no artifact
Combined filtering	32.72	no artifact

Table 22.1: PSNR after filtering the simulated image (Lena + Gaussian noise, sigma=20). In the combined filtering, a curvelet and an undecimated wavelet transform were used.

The noisy Lena image (the noise standard deviation being equal 20) was filtered by the undecimated wavelet transform, the curvelet transform, and by our combined transform approach (curvelet and undecimated wavelet transforms). The results are reported in Table 22.1. Figure 22.2 displays the noisy image (top left), and the restored image after denoising by the combined transforms (bottom right). Details are displayed in Figure 22.2 bottom left. Figure 22.2 bottom right shows the full residual image, and can be compared to the residual images shown in Figure 22.1. The residual is much better when the combined filtering is applied, and no feature can be detected any more by eye. This was not the case for either the wavelet and the curvelet filtering.

Figure 22.3 displays the PSNR of the solution versus the number of iterations. In this example, the algorithm is shown to converge rapidly. From a practical viewpoint only four or five iterations are truly needed. Note that the result obtained after a single iteration is already superior to those available using methods based on the thresholding of wavelet or curvelet coefficients alone.

Several papers have been recently published, based on the concept of minimizing the total variation under constraints in the wavelet domain [67, 54, 120] or in the curvelet domain [34].

Method	PSNR (coeff. l_1 norm minim.)	PSNR (TV minim.)
Undecimated wavelet only	32.00	32.43
Curvelet only	32.03	32.40
Wavelet + curvelet	32.72	32.77
(Combined filtering)		

Table 22.2: PSNR after filtering the simulated image (Lena + Gaussian noise, sigma=20). In the combined filtering, a curvelet and an undecimated wavelet transform have been used.

Our combined approach can be seen as a generalization of these methods. We carried out a set of experiments in order to estimate (i) if the total variation is better than the l_1 norm of the multiscale coefficients, and (ii) if the combined approach improves the results compared to a single transform based method.

In our example, Gaussian noise with a standard deviation equal to 20 was added to the classical Lena image (512 by 512). Several methods were used to filter the noisy image:

1. TV + constraint in the wavelet domain.
2. TV + constraint in the curvelet domain.
3. Wavelet l_1 norm minimization + wavelet constraints.
4. Curvelet l_1 norm minimization + curvelet constraints.
5. Combined filtering method using multiscale coefficient l_1 norm minimization.
6. Combined filtering method using TV minimization.

We use the PSNR as an “objective” measure of performance. The noisy image PSNR is 22.13. PSNR results from the different tested methods are reported in Table 22.2.

We observe that combined filtering leads to a significant improvement when compared to a single transform based method. The TV penalization gives better results when a single transform is used, while it seems not to have too much importance for the combined filtering approach. We will see in the following that the latter is not true for the deconvolution problem.

22.5 Discussion

We believe that the denoising experiments presented in this paper are of very high quality:

1. Combined filtering leads to a real improvement both in terms of PSNR and visual appearance.
2. The combined approach arguably challenges the eye to distinguish structure/features from residual images of real image data (at least for the range of noise levels that was considered here). Single transforms cannot manage such a feat.

We also note that the combined reconstruction may tend to be free of major artifacts which is very much unlike typical thresholding rules. Although the ease of implementation is clear we did not address the computational issues associated with our method. In a nutshell, the algorithms we described require calculating each transform and its inverse only a limited number of times.

In our examples, we constructed a combined transform from linear transforms (wavelets, ridgelets and curvelets) but our paradigm extends to any kind of nonlinear transform such as the Pyramidal Median Transform [191] or morphological multiscale transforms [89].



Figure 22.2: Noisy image (top left), and filtered image based on the combined transform (top right). Bottom left panel shows a detail of the filtered image. The full residual image is displayed on the bottom right.

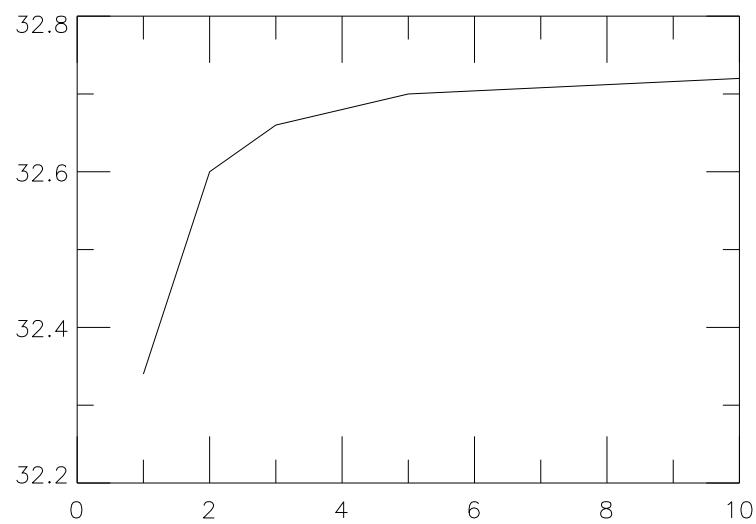


Figure 22.3: PSNR versus the number of iterations.

Chapter 23

Contrast Enhancement

23.1 Introduction

Because some features are hardly detectable by eye in an image, we often transform it before display. Histogram equalization is one the most well-known methods for contrast enhancement. Such an approach is generally useful for images with a poor intensity distribution. Since edges play a fundamental role in image understanding, a way to enhance the contrast is to enhance the edges. For example, we can add to the original image its Laplacian ($I' = I + \gamma\Delta I$, where γ is a parameter). Only features at the finest scale are enhanced (linearly). For a high γ value, only the high frequencies are visible. Multiscale edge enhancement [208] can be seen as a generalization of this approach to all resolution levels.

In color images, objects can exhibit variations in color saturation with little or no correspondence in luminance variation. Several methods have been proposed in the past for color image enhancement [205]. The retinex concept was introduced by Land [109] as a model for human color constancy. The single scale retinex (SSR) method [101] consists of applying the following transform to each band i of the color image:

$$R_i(x, y) = \log(I_i(x, y)) - \log(F(x, y) * I_i(x, y)) \quad (23.1)$$

where $R_i(x, y)$ is the retinex output, $I_i(x, y)$ is the image distribution in the i th spectral band, and F is a Gaussian function. A gain/offset is applied to the retinex output which clips the highest and lowest signal excursions. This can be done by a k-sigma clipping. The retinex method is efficient for dynamic range compression, but does not provide good tonal rendition [157]. The Multiscale Retinex (MSR) combines several SSR outputs to produce a single output image which has both good dynamic range compression and color constancy, and good tonal rendition [100]. The MSR can be defined by:

$$R_{MSR_i} = \sum_{j=1}^N w_j R_{i,j} \quad (23.2)$$

with

$$R_{i,j}(x, y) = \log(I_i(x, y)) - \log(F_j(x, y) * I_i(x, y)) \quad (23.3)$$

N is the number of scales, $R_{i,j}$ is the i th spectral component of the MSR output, and w_j is the weight associated with the scale j . The Gaussian F_j is given by:

$$F_j(x, y) = K \exp -\frac{r^2}{c_j^2} \quad (23.4)$$

c_j defines the width of the Gaussian. In [100], three scales were recommended with c_j values equal respectively to 15,80,250, and all weights w_j fixed to $\frac{1}{N}$. The Multiscale Retinex introduces

the concept of multiresolution for contrast enhancement. Velde [208] has explicitly introduced the wavelet transform and has proposed an algorithm which modifies the wavelet coefficients in order to amplify faint features. The idea is to first transform the image using the dyadic wavelet transform (two directions per scale). The gradient $G_{j,k}$ at scale j and at pixel location k is calculated at each scale j from the wavelet coefficients $w_{j,k}^{(h)}$ and $w_{j,k}^{(v)}$ relative to the horizontal and vertical wavelet bands: $G_{j,k} = \sqrt{(w_{j,k}^{(h)})^2 + (w_{j,k}^{(v)})^2}$. Then the two wavelet coefficients at scale j and at position k are multiplied by $y(G_{j,k})$, where y is defined by:

$$\begin{aligned} y(x) &= \left(\frac{m}{c}\right)^p \text{ if } |x| < c \\ y(x) &= \left(\frac{m}{|x|}\right)^p \text{ if } c \leq |x| < m \\ y(x) &= 1 \text{ if } |x| \geq m \end{aligned} \quad (23.5)$$

Three parameters are needed: p , m and c . p determines the degree of non-linearity in the

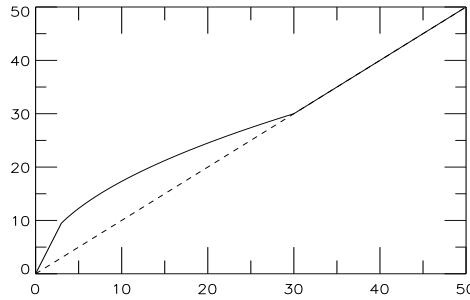


Figure 23.1: Enhanced coefficients versus original coefficients. Parameters are $m=30$, $c=3$ and $p=0.5$.

nonlinear rescaling of the luminance, and must be in $]0, 1[$. Coefficients larger than m are not modified by the algorithm. The c parameter corresponds to the noise level. Figure 23.1 shows the modified wavelet coefficients versus the original wavelet coefficients for a given set of parameters ($m = 30$, $c = 3$ and $p = 0.5$). Finally, the enhanced image is obtained by the inverse wavelet transform from the modified wavelet coefficients. For color images, a similar method can be used, but by calculating the multiscale gradient $\Gamma_{j,k}$ from the multiscale gradient of the three L , u , v components: $\Gamma_j(i) = \sqrt{\|G_{j,k}^L\|^2 + \|G_{j,k}^u\|^2 + \|G_{j,k}^v\|^2}$. All wavelet coefficients at scale j and at position k are multiplied by $y(\Gamma_{j,k})$, the enhanced \tilde{L} , \tilde{u} , \tilde{v} components are reconstructed from the modified wavelet coefficients, and the $(\tilde{L}, \tilde{u}, \tilde{v})$ image is transformed into an RGB image. More details can be found in [208].

Wavelet bases present some limitations, because they are not adapted to the detection of highly anisotropic elements, such as alignments in an image, or sheets in a cube. Recently, other multiscale systems like ridgelets [30] and curvelets [31, 185] which are very different from wavelet-like systems have been developed. Curvelets and ridgelets take the form of basis elements which exhibit very high directional sensitivity and are highly anisotropic. The curvelet transform uses the ridgelet transform in its digital implementation. We first describe the ridgelet and the curvelet transform, then we show how contrast enhancement can be obtained from the curvelet coefficients.

23.2 Contrast Enhancement by the Curvelet Transform

Since the curvelet transform is well-adapted to represent images containing edges, it is a good candidate for edge enhancement [179, 192]. Curvelet coefficients can be modified in order to

enhance edges in an image. A function y_c must be defined which modifies the values of the curvelet coefficients. It could be a function similar to the one defined for the wavelet coefficients [208] (see equation 23.5). This function presents however the drawback of amplifying the noise (linearly) as well as the signal of interest. We introduce explicitly the noise standard deviation σ in the equation:

$$\begin{aligned} y_c(x, \sigma) &= 1 \text{ if } x < c\sigma \\ y_c(x, \sigma) &= \frac{x - c\sigma}{c\sigma} \left(\frac{m}{c\sigma} \right)^p + \frac{2c\sigma - x}{c\sigma} \text{ if } c\sigma < x < m \\ y_c(x, \sigma) &= \left(\frac{m}{x} \right)^p \text{ if } m \leq x < m \\ y_c(x, \sigma) &= \left(\frac{m}{x} \right)^s \text{ if } x \geq m \end{aligned} \quad (23.6)$$

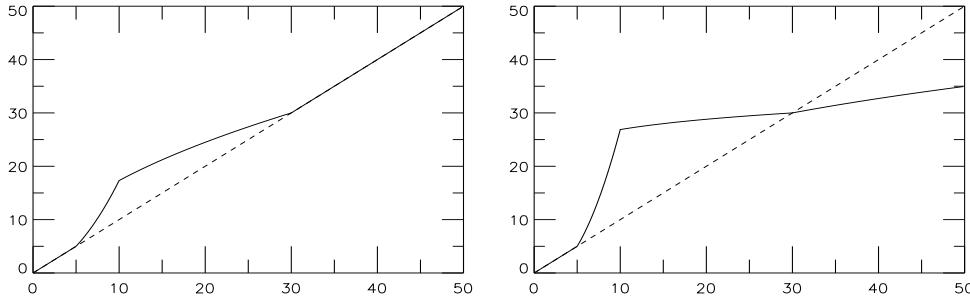


Figure 23.2: Enhanced coefficients versus original coefficients. Left, parameters are $m=30, c=0.5, s=0$, and $p=0.5$. Right, parameters are $m=30, c=0.5, s=0.7, p=0.9$.

We have fixed $m = c = p = 0.5$ and $s = 0$ in all our experiments. p determines the degree of non-linearity and s introduces a saturation. c becomes a normalized parameter, and a c value larger than 3 guarantees that the noise will not be amplified. The m parameter can be defined either from the noise standard deviation ($m = K_m \sigma$) or from the maximum curvelet coefficient M_c of the relative band ($m = l M_c$, with $l < 1$). The first choice allows the user to define the coefficients to amplify as a function of their signal-to-noise ratio, while the second one gives an easy and general way to fix the m parameter independently of the range of the pixel values. Figure 23.2 shows the curve representing the enhanced coefficients versus the original coefficients for two sets of parameters. In the second case, a saturation is added.

The curvelet enhancement method for grayscale images consists of the following steps:

1. Estimate the noise standard deviation σ in the input image I .
2. Calculate the curvelet transform of the input image. We get a set of bands w_j , each band w_j contains N_j coefficients and corresponds to a given resolution level.
3. Calculate the noise standard deviation σ_j for each band j of the curvelet transform (see [185] more details on this step).
4. For each band j do
 - Calculate the maximum M_j of the band.
 - Multiply each curvelet coefficient $w_{j,k}$ by $y_c(|w_{j,k}|, \sigma_j)$.
5. Reconstruct the enhanced image from the modified curvelet coefficients.

For color images, we apply first the curvelet transform on the three components L, u, v . For each curvelet coefficient, we calculate $e = \sqrt{c_L^2 + c_u^2 + c_v^2}$, where (c_L, c_u, c_v) are respectively

the curvelet coefficients of the three components, and the modified coefficients are obtained by: $(\tilde{c}_L, \tilde{c}_u, \tilde{c}_v) = (y_c(e, \sigma)c_L, y_c(e, \sigma)c_u, y_c(e, \sigma)c_v)$.

Values in the enhanced components can be larger than the authorized upper limit (in general 255), and we found it necessary to add a final step to our method, which is a sigma-clipping saturation.

23.3 Examples

Saturn Image

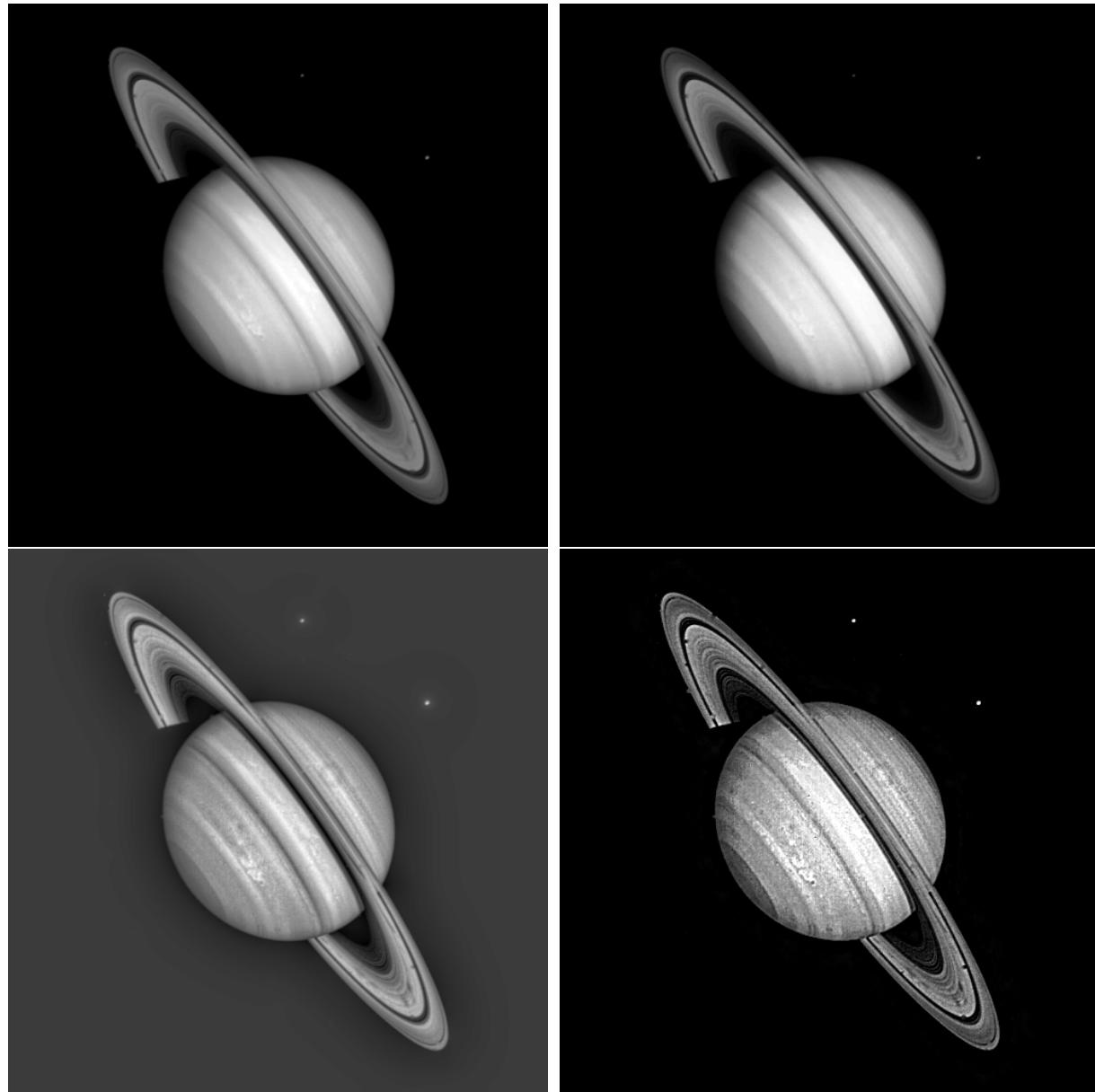


Figure 23.3: Top, Saturn image and its histogram equalization. Bottom, enhancement image by the wavelet transform and the curvelet transform.

Figure 23.3 shows respectively from left to right and from top to bottom the Saturn image,

the histogram equalized image, the wavelet multiscale edge enhanced image and the curvelet multiscale edge enhanced image (parameters were $s = 0$, $p = 0.5$, $c = 3$, and $l = 0.5$). The curvelet multiscale edge enhanced image shows clearly better the rings and edges of Saturn.

Satellite Image

Figure 23.4 shows the results for the enhancement of a grayscale satellite image, and Figure 23.5 shows the results for the enhancement of a color image (Kodak image of the day 14/05/01) by the retinex, the multiscale retinex and the curvelet multiscale edge enhancement methods. Figure 23.6 shows the results for the enhancement of a color image (Kodak image of the day 11/12/01).

23.4 Discussion

A number of properties, respected by the curvelet filtering described here, are important for contrast stretching:

1. Noise must not be amplified in enhancing edges.
2. Colors should not be unduly modified. In multiscale retinex, for example, a tendency towards increased grayness is seen. This is not the case using curvelets.
3. It is very advantageous if block effects do not occur. Block overlapping is usually not necessary in curvelet-based contrast enhancement, unlike in the case of noise filtering.

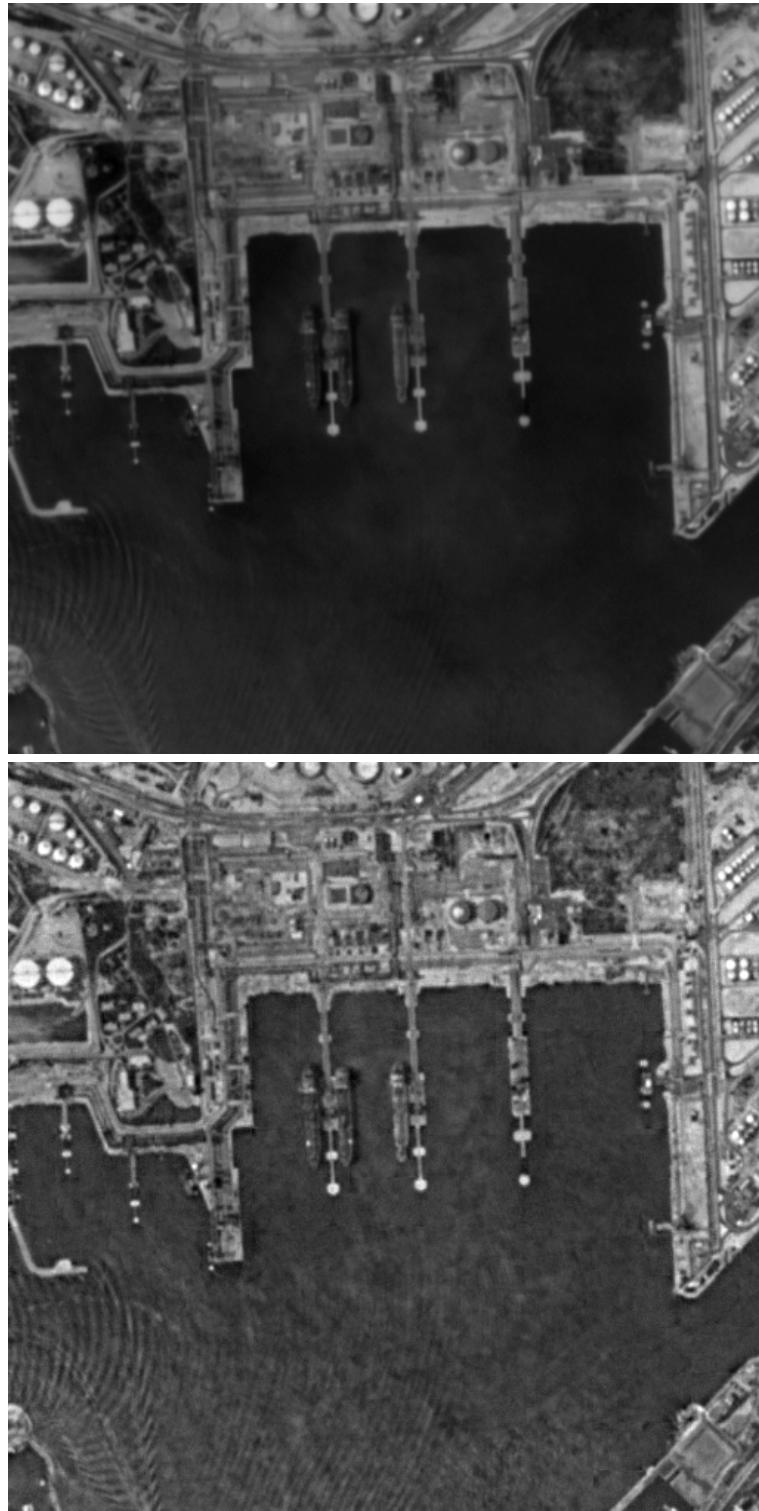


Figure 23.4: Top, grayscale image, and bottom, curvelet enhanced image.



Figure 23.5: Top, color image (Kodak picture of the day 14/05/02) and retinex method. Bottom, multiscale retinex method and multiscale edge enhancement.



Figure 23.6: Left, color image (Kodak picture of the day 11/12/01), and right, curvelet enhanced image.

Chapter 24

MR/4 Programs

24.1 Ridgelet Transform

24.1.1 im_radon

Program *im_radon* makes an (inverse-) Radon transform of a square $n \times n$ image. The output file which contains the transformation has a suffix, .rad. If the output file name given by the user does not contain this suffix, it is automatically added. The “.rad” file is a FITS format file, and can be manipulated by any package implementing the FITS format, or can be converted to another format using the *im_convert* program. For the two first Radon transform methods, the user can change the number of directions and the resolution. For other methods, the number of directions and the resolution are fixed, and the x and y options are not valid. Options f, w and s allow the user to perform a filtering-backprojection. They are valid only when the selected Radon method is the second one. “w” fixes the width of the filter and the “s” is the sigma parameter of the Gaussian filter.

USAGE: **im_radon options image_in trans_out**

where options are:

- **[-m type_of_radon_method]**

1. Radon transform (resp. backprojection) in spatial domain.
By default, the output image is a $2n \times n$ image.
2. Radon projection in spatial domain and reconstruction in Fourier domain. By default, the output image is a $2n \times n$ image. The reconstruction is available only for image with a size n being a power of 2.
3. Radon transformation and reconstruction in Fourier space (i.e. Linogram).
The output is a $2n \times n$ image. The number of rows is multiplied by two.
4. Finite Radon Transform.
The output image is a $(n + 1) \times n$ image. The input image size n must be a prime number.
5. Slant Stack Radon transform.
The output image has twice the number of rows and twice the number of columns of the input image. The output is a $2n \times 2n$ image. The reconstruction is not available with this transform.

Default is Radon transformation and reconstruction in Fourier space.

- **[-y OutputLineNumber]**

For the RADON transform, OutputLineNumber = number of projection, and default is twice the number of input image rows. Only valid for Radon methods 1 and 2.

- **[-x OutputColumnName]**

For the RADON transform, OutputLineNumber = number of pixels per projection, and default is the input image column number. Only valid for Radon methods 1 and 2.

- **[-r]**

Inverse Radon transform.

- **[-f]**

Filter each scan of the Radon transform. Only valid for Radon method 2.

- **[-w FilterWidth]**

Filter width. Only valid for Radon method 2. Default is 100.

- **[-s SigmaParameter]**

Sigma parameter for the filtering. Only valid for Radon method 2. Default is 10.

- **[-v]**

Verbose. Default is no

Examples:

- `im_radon image.fits trans`

Apply the Radon transform to an image.

- `im_info -r trans.rad rec`

Reconstruct an image from its Radon transform.

24.1.2 rid_trans

Program *rid_trans* makes the ridgelet transform (and the inverse when -r option is set). The output file which contains the transformation has a suffix, .rid. If the output file name given by the user does not contain this suffix, it is automatically added. The “.rid” file is a FITS format file, and can be manipulated by any package implementing the FITS format, or can be converted to another format using the *im_convert* program. The default transform is the second one (RectoPolar Ridgelets using a FFT based WT). The two first transform are based on the RectoPolar (i.e. linogram) radon transform, but the first applies a standard bi-orthogonal wavelet transform (WT) on the Radon image rows, while the second uses the Fourier-based WT which introduces a redundancy of 2. For an $n \times n$ image, the output has $2n$ lines and n column with the first transform, and is a $2n \times 2n$ image for the second. When the overlapping is set, the size is doubled in each direction.

USAGE: rid_trans options image_in trans_out

where options are:

- **[-t type_of_ridgelet]**

1. RectoPolar Ridgelet Transform using a standard bi-orthogonal WT.
2. RectoPolar Ridgelet Transform using a FFT based Pyramidal WT.
3. Finite ridgelet transform.

Default is 2.

- **[-n number_of_scale]**

Number of scales used in the wavelet transform. Default is automatically calculated.

- **[-b BlockSize]**

Block Size. Default is image size.

- **[-i]**
Print statistical information about each band. Default is no.
- **[-O]**
No block overlapping. Default is no. When this option is set, the number of rows and columns is multiplied by two.
- **[-r]**
Inverse Ridgelet transform.
- **[-x]**
Write all bands separately as images in the FITS format with prefix 'band.j' (j being the band number).
- **[-v]**
Verbose. Default is no

Examples:

- rid_transform image.fits trans
Apply the Ridgelet transform to an image.
- rid_transform -r trans.rid rec
Reconstruct an image from its Ridgelet transform.

24.1.3 rid_stat

Program *rid_stat* makes the ridgelet transform, and gives statistical information on the ridgelet coefficients. At each scale, it calculates the standard deviation, the skewness, the kurtosis, the minimum, and the maximum. The output file is a fits file containing a two-dimensional array $T[J - 1, 5]$ (J being the number of scales), with the following syntax:

- $T[j, 0]$ = standard deviation of the jth ridgelet band.
- $T[j, 1]$ = skewness of the jth ridgelet band.
- $T[j, 2]$ = kurtosis of the jth ridgelet band.
- $T[j, 3]$ = minimum of the jth ridgelet band.
- $T[j, 4]$ = maximum of the jth ridgelet band.

The last ridgelet scale is not used. If the “-A” option is set, these statistics are calculated only for the ridgelet coefficients relative the specified angle.

USAGE: rid_stat options image_in trans_out

where options are:

- **[-t type_of_ridgelet]**
 1. RectoPolar Ridgelet Transform using a standard bi-orthogonal WT.
 2. RectoPolar Ridgelet Transform using a FFT based Pyramidal WT.
 3. Finite ridgelet transform.

Default is 2.
- **[-n number_of_scale]**
Number of scales used in the wavelet transform. Default is automatically calculated.
- **[-b BlockSize]**
Block Size. Default is 16.

- **[-O]**
Use overlapping block. Default is no.
- **[-A Angle]**
Statistics for a given angle. The value must be given in degrees. Default is no, statistics are calculated from all coefficients.
- **[-v]**
Verbose. Default is no

Example:

- `rid_stat -v image.fits tabstat`

24.1.4 rid_filter

Program *rid_filter* filters an image using the ridgelet transform.

USAGE: `rid_filter options image_in imag_out`

where options are

- **[-t type_of_ridgelet]**
 1. RectoPolar Ridgelet Transform using a standard bi-orthogonal WT.
 2. RectoPolar Ridgelet Transform using a FFT based pyramidal WT.
 3. Finite ridgelet transform.

Default is 2.

- **[-n number_of_scale]**
Number of scales used in the wavelet transform. Default is automatically calculated.
- **[-h]**
Apply the ridgelet transform only on the high frequencies. Default is no.
- **[-b BlockSize]**
Block Size. Default is image size.
- **[-F FirstDetectionScale]**
First detection scale. Default is 1.
- **[-s Nsigma]**
False detection rate. The false detection rate for a detection is given

$$\epsilon = \operatorname{erfc}(NSigma/\sqrt{2}) \quad (24.1)$$

Nsigma parameter allows us to express the false detection rate even if it is not Gaussian noise.

Default is 3.

- **[-g sigma]**
Gaussian noise: σ = noise standard deviation.
Default is automatically estimated.
- **[-p]**
Poisson noise.
- **[-O]**
Do not apply block overlapping. By default, block overlapping is used.
- **[-v]**
Verbose.

Examples:

- `rid_filter image.fits fima`
Filter an image using all default options.
- `rid_filter -h -s5 image.fits fima`
Five sigma filtering, filtering only the high frequencies.

24.2 Curvelet Transform

24.2.1 cur_trans

Program `cur_trans` determines the curvelet transform (and the inverse when `-r` option is set). The output file which contains the transformation has a suffix, `.cur`. If the output file name given by the user does not contain this suffix, it is automatically added. The “`.cur`” file is a 3D FITS format file, and can be manipulated by any package implementing the FITS format. The curvelet transform uses the ridgelet transform, and the default ridgelet transform is the RectoPolar one with a FFT based pyramidal WT.

USAGE: `cur_trans options image_in trans_out`

where options are:

- **`[-t type_of_ridgelet]`**
1. RectoPolar Ridgelet Transform using a standard bi-orthogonal WT.
2. RectoPolar Ridgelet Transform using a FFT based pyramidal WT.
3. Finite ridgelet transform.
Default is 2.
- **`[-n number_of_scale]`**
Number of scales used in the 2D wavelet transform. Default is 4.
- **`[-N number_of_scale]`**
Number of scales used in the ridgelet transform. Default is automatically calculated.
- **`[-b BlockSize]`**
Block Size. Default is 16.
- **`[-r]`**
Inverse Curvelet transform.
- **`[-i]`**
Print statistical information about each band. Default is no.
- **`[-O]`**
Block overlapping. Default is no.
- **`[-x]`**
Write all bands separately as images in the FITS format with prefix ‘band_j’ (j being the band number).
- **`[-v]`**
Verbose. Default is no.

Examples:

- `cur_trans -i image.fits trans`
Curvelet transform of an image.
- `cur_trans -r trans.cur rec`
Image reconstruction from its curvelet transform.

24.2.2 cur_stat

Program *cur_stat* determines the curvelet transform, and gives statistical information on the curvelet coefficients. At each scale, it calculates the standard deviation, the skewness, the kurtosis, the minimum, and the maximum. The output file is a FITS file containing a two-dimensional array $T[J - 1, 5]$ (J being the number of bands), with the following syntax:

- $T[j, 0]$ = standard deviation of the j th ridgelet band.
- $T[j, 1]$ = skewness of the j th ridgelet band.
- $T[j, 2]$ = kurtosis of the j th ridgelet band.
- $T[j, 3]$ = minimum of the j th ridgelet band.
- $T[j, 4]$ = maximum of the j th ridgelet band.

USAGE: cur_stat options image_in trans_out

where options are:

- **[-n number_of_scale]**
Number of scales used in the wavelet transform. Default is automatically calculated.
- **[-b BlockSize]**
Block Size. Default is 16.
- **[-O]**
Use overlapping block. Default is no.
- **[-v]**
Verbose. Default is no

Example:

- cur_stat -v image.fits tabstat

24.2.3 cur_filter

Program *cur_filter* filters an image using the curvelet transform.

USAGE: cur_filter options image_in imag_out

where options are

- **[-t type_of_ridgelet]**
 1. RectoPolar Ridgelet Transform using a standard bi-orthogonal WT.
 2. RectoPolar Ridgelet Transform using a FFT based Pyramidal WT.
 3. Finite ridgelet transform.

Default is 2.
- **[-n number_of_scale]**
Number of scales used in the 2D wavelet transform. Default is 4.
- **[-N number_of_scale]**
Number of scales used in the ridgelet transform. Default is automatically calculated.
- **[-b BlockSize]**
Block Size. Default is 16.

- **[-g sigma]**
Gaussian noise: sigma = noise standard deviation.
Default is automatically estimated.
- **[-s Nsigma]**
False detection rate.
Default is 3.
- **[-O]**
Do not apply block overlapping. By default, block overlapping is used.
- **[-P]**
Suppress the positivity constraint. Default is no.
- **[-I NoiseFileName]**
If the noise is stationary, the program can estimate the correct thresholds from a realization of the noise.
- **[-v]**
Verbose

Examples:

- `cur_filter image.fits sol`
Curvelet filtering of an image.
- `cur_filter -n 5 -s4 image.fits sol`
Curvelet filtering of an image, using five resolution levels, and a 4-sigma detection.

24.2.4 cur_colfilter

Program *cur_colfilter* filters a color image using the curvelet transform.

USAGE: *cur_colfilter* options *image_in* *image_out*

where options are

- **[-n number_of_scale]**
Number of scales used in the 2D wavelet transform. Default is 4.
- **[-N number_of_scale]**
Number of scales used in the ridgelet transform. Default is automatically calculated.
- **[-b BlockSize]**
Block Size. Default is 16.
- **[-g sigma]**
Gaussian noise: sigma = noise standard deviation.
Default is automatically estimated.
- **[-s Nsigma]**
False detection rate.
Default is 3.
- **[-O]**
Do not apply block overlapping. By default, block overlapping is used.
- **[-v]**
Verbose

Example:

- cur_colfilter image.fits sol
Curvelet transform of a color image.

24.2.5 cur_contrast

Program *cur_contrast* filters a color image using the curvelet transform.

USAGE: cur_contrast options image_in imag_out

where options are

- **[-n number_of_scales]**
Number of scales used in the wavelet transform. Default is 4.
- **[-N number_of_scales]**
Number of scales used in the ridgelet transform. Default is automatically calculated.
- **[-b BlockSize]**
Block size used by the curvelet transform. Default is 16.
- **[-O]**
Use overlapping block. Default is no.
- **[-g sigma]**
Noise standard deviation. Only used when filtering is performed. Default is automatically estimated.
- **[-s NSignalLow]**
Coefficient < NSignalLow*SigmaNoise is not modified. Default is 5.
- **[-S NSignalUp]**
Coefficient > NSignalUp*SigmaNoise is not modified. Default is 20.
- **[-M MaxCoeff]**
If MaxBandCoef is the maximum coefficient in a given curvelet band, Coefficient > MaxBandCoef*MaxCoeff is not modified. Default is 0.5.
- **[-P P_parameter]**
Determine the degree on non-linearity. P must be in]0,1[. Default is 0.5.
- **[-T P_parameter]**
Curvelet coefficient saturation parameter. T must be in [0,1]. Default is 0.
- **[-c]**
By default a sigma clipping is performed. When this option is set, no sigma clipping is performed.
- **[-K ClippingValue]**
Clipping value. Default is 3.
- **[-L Saturation]**
Saturate the reconstructed image. A coefficient larger than Saturation*MaxData is set to Saturation*MaxData. Default is 1. If L is set to 0, then no saturation is applied.

Examples:

- cur_contrast image.fits image_out.fits
Enhance the contrast using the curvelet transform.
- cur_contrast -O image.fits image_out.fits
Enhance the contrast using the curvelet transform and block overlapping.

- `cur_colcontrast -M 0.8 image.fits image_out.fits`
Enhance more the contrast.

24.2.6 cur_colcontrast

The program `col_colcontrast` enhances the contrast of a color image using the curvelet transform. The command line is:

USAGE: cur_colcontrast option in_image out_image

where options are:

- **[-n number_of_scales]**
Number of scales used in the wavelet transform. Default is 4.
- **[-N number_of_scales]**
Number of scales used in the ridgelet transform. Default is automatically calculated.
- **[-b BlockSize]**
Block size used by the curvelet transform. Default is 16.
- **[-O]**
Use overlapping block. Default is no.
- **[-g sigma]**
Noise standard deviation. Default is automatically estimated.
- **[-s NSigmaLow]**
Coefficient < NSigmaLow*SigmaNoise is not modified. Default is 5.
- **[-S NSigmaUp]**
Coefficient > NSigmaUp*SigmaNoise is not modified. Default is 20.
- **[-M MaxCoeff]**
If MaxBandCoef is the maximum coefficient in a given curvelet band, Coefficient > MaxBandCoef*MaxCoeff is not modified. Default is 0.5.
- **[-P P_parameter]**
Determine the degree of non-linearity. P must be in]0,1[. Default is 0.5.
- **[-T P_parameter]**
Curvelet coefficient saturation parameter. T must be in [0,1]. Default is 0.
- **[-c]**
By default a sigma clipping is performed. When this option is set, no sigma clipping is performed.
- **[-K ClippingValue]**
Clipping value. Default is 3.
- **[-L Luminance_Saturation]**
Values in the luminance map which are larger than Saturation*MaxData are set to Saturation*MaxData. Default is 1.

Examples:

- `cur_colcontrast image.tiff image_out.tiff`
Enhance the contrast using the curvelet transform.
- `cur_colcontrast -n 5 image.tiff image_out.tiff`
Ditto, but use five scales instead of four.
- `cur_colcontrast -M 0.9 image.tiff image_out.tiff`
Enhance more the contrast.

24.3 Combined Filtering

24.3.1 cb_filter

Program *cb_filter* filters an image corrupted by Gaussian noise by the combined filtering method. By default, the undecimated bi-orthogonal WT and the curvelet transform are used. The number of iterations is defaulted 10. In general, the algorithm converges with less than six iterations. If the “-T” option is set, the Total Variation is minimized instead of the l_1 norm of the multiscale coefficients. A deconvolution can also be performed using the “-P” option. In this case, a division is first done in Fourier space between the input image and the point spread function. All Fourier components with a norm lower than ϵ (default value is 10^{-3}) are set to zero. Then the deconvolved image is filtered by the combined filtering method using the new noise properties (still Gaussian, but not white).

USAGE: `cb_filter options image_in trans_out`

where options are:

- **[-t TransformSelection]**
 1. A trous algorithm
 2. Bi-orthogonal WT with 7/9 filters
 3. Ridgelet transform
 4. Curvelet transform
 5. Mirror Basis WT
- **[-n number_of_scales]**
Number of scales used in the à trous wavelet transform and the curvelet transform. Default is 4.
- **[-b BlockSize]**
Block Size in the ridgelet transform. Default is image size.
- **[-i NbrIter]**
Number of iterations. Default is 10.
- **[-F FirstDetectionScale]**
First detection scale in the ridgelet transform. Default is 1.
- **[-k]**
Kill the last scale in ridgelet. Default no.
- **[-K]**
Kill last scale in the à trous algorithm and the curvelet. Default no.
- **[-L FirstSoftThreshold]**
First soft thresholding value. Default is 0.5.
- **[-l LastSoftThreshold]**
Last soft thresholding value. Default is 0.5.
- **[-u]**
Number of undecimated scales in the WT. Default is 1.
- **[-s Nsigma]**
False detection rate. Default is 4.
- **[-g sigma]**
Gaussian noise: $\text{sigma} = \text{noise standard deviation}$.
Default is automatically estimated.

- **[-O]**
No block overlapping. Default is no.
- **[-T]**
Minimize the Total Variation instead of the L1 norm.
- **[-P PsfFile]**
Apply a deconvolution using the PSF in the file PsfFile.
- **[-e Eps]**
Remove frequencies with $|PP^*| < \epsilon$. Default is 10^{-3} .
- **[-C TolCoef]**
Default is 0.5.
- **[-v]**
Verbose. Default is no.

Example:

- cb_filter image.fits sol
Image filtering by the combined filtering method, using both the curvelet and wavelet transform.

Bibliography

- [1] J.G. Ables. Maximum entropy spectral analysis. *Astronomy and Astrophysics*, 15:383–393, 1974.
- [2] E.H. Adelson, E. Simoncelli, and R. Hingorani. Optimal image addition using the wavelet transform. *SPIE Visual Communication and Image Processing II*, 845:50–58, 1987.
- [3] U. Amato and D.T. Vuza. Besov regularization, thresholding and wavelets for smoothing data. *Numerical Functional Analysis and Optimization*, 18:461–493, 1997.
- [4] U. Amato and D.T. Vuza. Wavelet approximation of a function from samples affected by noise. *Revue Roumaine de Mathématiques Pures et Appliquées*, 42:481–499, 1997.
- [5] H.C. Andrews and C.L. Patterson. Singular value decompositions and digital image processing. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 24:26–53, 1976.
- [6] F.J. Anscombe. The transformation of Poisson, binomial and negative-binomial data. *Biometrika*, 15:246–254, 1948.
- [7] M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies. Image coding using wavelet transform. *IEEE Transactions on Image Processing*, 1:205–220, 1992.
- [8] A. Arneodo, F. Argoul, E. Bacry, J. Elezgaray, and J. F. Muzy. *Ondelettes, Multifractales et Turbulences*. Diderot, Arts et Sciences, Paris, 1995.
- [9] A. Averbuch, R.R. Coifman, D.L. Donoho, M. Israeli, and J. Waldén. Fast Slant Stack: A notion of Radon transform for data in a cartesian grid which is rapidly computable, algebraically exact, geometrically faithful and invertible. *SIAM J. Sci. Comput.*, 2001. To appear.
- [10] C. Baccigalupi, L. Bedini, C. Burigana, G. De Zotti, A. Farusi, D. Maino, M. Maris, F. Perrotta, E. Salerno, L. Toffolatti, and A. Tonazzini. Neural networks and the separation of cosmic microwave background and astrophysical signals in sky maps. *Monthly Notices of the Royal Astronomical Society*, 318:769–780, November 2000.
- [11] J.G. Bartlett, F. Bonnarel, D. Egret, F. Genova, H. Ziaeepour, O. Bienaymé, F. Ochsenbein, M. Crézé, J. Florsch, M. Louys, and Ph. Paillou. The ALADIN project: status report. In G.H. Jacoby and J. Barnes, editors, *Astronomical Data Analysis Software and Systems V*, pages 489–492, 1996.
- [12] Z. Bashir and M.E. El-Hawary. Short term load forecasting by using wavelet neural networks. In *Canadian Conference on Electrical and Computer Engineering*, pages 163–166, 2000.
- [13] A. Bijaoui. *Introduction au Traitement Numérique des Images*. Masson, 1984.
- [14] A. Bijaoui, Y. Bobichon, Y. Fang, and F. Rué. Méthodes multiéchelles appliquées à l’analyse des images radar à ouverture synthétique. *Traitement du Signal*, 14:179–193, 1997.

- [15] A. Bijaoui, Y. Bobichon, and L. Huang. Digital image compression in astronomy – morphology or wavelets. *Vistas in Astronomy*, 40:587–594, 1996.
- [16] A. Bijaoui, P. Bury, and E. Slezak. Catalog analysis with multiresolution insights. 1. Detection and characterization of significant structures. Technical report, Observatoire de la Côte d’Azur, 1994.
- [17] A. Bijaoui and V. Doazan. Analysis of rapid variations in the spectra of α Col by cross-correlation. *Astronomy and Astrophysics*, 70:285–291, 1984.
- [18] A. Bijaoui and G. Jammal. On the distribution of the wavelet coefficient for a Poisson noise. *Signal Processing*, 81:1789–1800, 2001.
- [19] A. Bijaoui and F. Rué. A multiscale vision model adapted to astronomical images. *Signal Processing*, 46:229–243, 1995.
- [20] A. Bijaoui, F. Rué, and B. Vandame. Multiscale vision and its application to astronomy. In *Proceedings of the Fifth Workshop in Data Analysis in Astronomy*, pages 337–343. World Scientific, 1992.
- [21] A. Bijaoui, J.-L. Starck, and F. Murtagh. Restauration des images multi-échelles par l’algorithme à trous. *Traitemet du Signal*, 3:11, 1994.
- [22] L. Blanc-Féraud and M. Barlaud. Edge preserving restoration of astrophysical images. *Vistas in Astronomy*, vol. 40(no. 4):pp. 531–538, 1996.
- [23] Y. Bobichon and A. Bijaoui. A regularized image restoration algorithm for lossy compression in astronomy. *Experimental Astronomy*, 7:239–255, 1997.
- [24] T.R. Bontekoe, E. Koper, and D.J.M. Kester. Pyramid maximum entropy images of IRAS survey data. *Astronomy and Astrophysics*, 284:1037–1053, 1994.
- [25] E.J. Breen, R.J. Jones, and H. Talbot. Mathematical morphology: A useful set of tools for image analysis. *Statistics and Computing*, 10:105–120, 2000.
- [26] S. Brette and J. Idier. Optimized single site update algorithms for image deblurring. *Proceedings of IEEE ICIP*, pages 65–68, 1996.
- [27] J.P. Burg. Multichannel maximum entropy spectral analysis. Annual Meeting International Society Exploratory Geophysics, Reprinted in *Modern Spectral Analysis*, D.G. Childers, ed., IEEE Press, 34–41, 1978.
- [28] P. Bury. *De la distribution de matière à grande échelle à partir des amas d’Abell*. PhD thesis, Université de Nice Sophia Antipolis, Janvier 1995.
- [29] R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo. Wavelet transforms that map integers to integers. *Applied and Computational Harmonic Analysis*, 5:332–369, 1998.
- [30] E. J. Candès. Harmonic analysis of neural networks. *Applied and Computational Harmonic Analysis*, 6:197–218, 1999.
- [31] E. J. Candès and D. L. Donoho. Curvelets – a surprisingly effective nonadaptive representation for objects with edges. In A. Cohen, C. Rabut, and L.L. Schumaker, editors, *Curve and Surface Fitting: Saint-Malo 1999*, Nashville, TN, 1999. Vanderbilt University Press.
- [32] E. J. Candès and D. L. Donoho. Ridgelets: the Key to Higher-dimensional Intermittency? *Philosophical Transactions of the Royal Society of London A*, 357:2495–2509, 1999.
- [33] E.J. Candès and D. Donoho. Ridgelets: the key to high dimensional intermittency? *Philosophical Transactions of the Royal Society of London A*, 357:2495–2509, 1999.

- [34] E.J. Candès and F. Guo. New multiscale transforms, minimum total variation synthesis: Applications to edge-preserving image reconstruction. *Signal Processing*, 82(11):1519–1543, 2002.
- [35] J.F. Cardoso. Blind signal separation: Statistical principles. *Proceedings of the IEEE*, 86:2009–2025, October 1998.
- [36] A. Chambolle, R.A. DeVore, N. Lee, and B.J. Lucier. Nonlinear wavelet image processing: Variational problems, compression, and noise removal through wavelet shrinkage. *IEEE Transactions on Signal Processing*, 7:319–335, 1998.
- [37] M.K. Charter. Drug absorption in man, and its measurement by MaxEnt. In W.V.D. Linden and V. Dose, editors, *Maximum Entropy and Bayesian Methods*, pages 325–339. Kluwer, 1990.
- [38] H.A. Chipman, E.D. Kolaczyk, and R.E. McCulloch. Adaptive Bayesian wavelet shrinkage. *Journal of the American Statistical Association*, 92:1413–1421, 1997.
- [39] C.H. Chui. *Wavelet Analysis and Its Applications*. Academic Press, 1992.
- [40] P.H. Van Cittert. Zum Einfluß der Spaltbreite auf die Intensitätsverteilung in Spektrallinien II. *Zeitschrift für Physik*, 69:298–308, 1931.
- [41] A. Cohen, I. Daubechies, and J.C. Feauveau. Biorthogonal bases of compactly supported wavelets. *Communications in Pure and Applied Mathematics*, 45:485–560, 1992.
- [42] L. Cohen. *Time-Frequency Analysis*. Prentice-Hall, 1995.
- [43] R.R. Coifman and D.L. Donoho. Translation invariant de-noising. In A. Antoniadis and G. Oppenheim, editors, *Wavelets and Statistics*, pages 125–150. Springer-Verlag, 1995.
- [44] R.R. Coifman, Y. Meyer, and M.V. Wickerhauser. Wavelet analysis and signal processing. In M.B. Ruskai, G. Beylkin, R. Coifman, I. Daubechies, S. Mallat, Y. Meyer, and L. Raphael, editors, *Wavelets and Their Applications*, pages 153–178. Jones and Bartlett Publishers, 1992.
- [45] T. J. Cornwell. Image Restoration. In *NATO ASIC Proc. 274: Diffraction-Limited Imaging with Very Large Telescopes*, pages 273–+, 1989.
- [46] R. Cristi and M. Tummula. Multirate, multiresolution, recursive Kalman filter. *Signal Processing*, 80:1945–1958, 2000.
- [47] M. Crouse, R. Nowak, and R. Baraniuk. Wavelet-based statistical signal processing using hidden Markov models. *IEEE Transactions on Signal Processing*, 46:886–902, 1998.
- [48] J.C. Dainty. Laser speckle and related phenomena. In *Topics in Applied Physics*, volume 9, Berlin, 1975. Springer-Verlag.
- [49] P.E. Danielsson and M. Hammerin. High accuracy rotation of images. *CVGIP: Graphical Models and Image Processing*, 45:340–344, 1992.
- [50] USC-SIPI Image Database. <http://sipi.usc.edu/services/database/-Database.html>.
- [51] I. Daubechies. Orthogonal bases of compactly supported wavelets. *Communications in Pure and Applied Mathematics*, 41:909–996, 1988.
- [52] I. Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, 1992.
- [53] S.R. Deans. *The Radon Transform and Some of Its Applications*. Wiley, 1983.

- [54] P. Dhérété, S. Durand, J. Froment, and B. Rougé. A best wavelet packet basis for joint image deblurring-denoising and compression. In *SPIE's 47th Annual Meeting, Seattle, July 7-11, 2002*.
- [55] J.P. Djamdji. *Analyse en ondelettes et mise en correspondance en télédétection*. PhD thesis, Université de Nice Sophia Antipolis, 2 Décembre 1993.
- [56] J.P. Djamdji, A. Bijaoui, and R. Maniere. Geometrical registration of images: the multi-resolution approach. *Photogrammetric Engineering and Remote Sensing*, 59(5):645–653, May 1993.
- [57] J.P. Djamdji, A. Bijaoui, and R. Maniere. Geometrical Registration of Remotely Sensed Images with the Use of the Wavelet Transform. *SPIE's International Symposium on Optical Engineering and Photonics*, pages 412–422, 12-16 April - Orlando 1993. Volume 1938.
- [58] M. N. Do and M. Vetterli. Image denoising using the orthonormal finite ridgelet transform. In *SPIE conference on Signal and Image Processing: Wavelet Applications in Signal and Image Processing VIII*, August 2000.
- [59] M. N. Do and M. Vetterli. Orthonormal finite ridgelet transform for image compression. In *Proc. of IEEE International Conference on Image Processing (ICIP)*, September 2000.
- [60] D. Donoho and A.G. Flesia. Digital Ridgelet Transform Based on True Ridge Functions. In J. Schmeidler and G.V. Welland, editors, *Beyond Wavelets*. Academic Press, 2002.
- [61] D. L. Donoho. Fast ridgelet transforms in dimension 2. Technical report, Stanford University, Department of Statistics, Stanford CA 94305–4065, 1997.
- [62] D. L. Donoho. Digital ridgelet transform via rectopolar coordinate transform. Technical report, Stanford University, 1998.
- [63] D.L. Donoho. Nonlinear wavelet methods for recovery of signals, densities, and spectra from indirect and noisy data. In American Mathematical Society, editor, *Proceedings of Symposia in Applied Mathematics*, volume 47, pages 173–205, 1993.
- [64] D.L. Donoho. Orthonormal ridgelets and linear singularities. *Siam J. Math Anal.*, 31(5):1062–1099, 2000.
- [65] D.L. Donoho and I.M. Johnstone. Ideal spatial adaptation via wavelet shrinkage. *Biometrika*, 81:425–455, 1994.
- [66] D.L. Donoho and I.M. Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *American Statistical Association*, 90:1200–1224, 1995.
- [67] S. Durand and J. Froment. Reconstruction of wavelet coefficients using total variation minimization. Technical Report 2001-18, CMLA, November 2001.
- [68] P. Edholm and G.T. Herman. Linograms and image reconstruction from projections. *IEEE Transactions on Medical Imaging*, 6(4):301–307, 1987.
- [69] P. Edholm and G.T. Herman. Image reconstruction from linograms: Implementation and evaluation. *IEEE Transactions on Medical Imaging*, 7(3):239–246, 1988.
- [70] B.R. Epstein, R. Hingorani, J.M. Shapiro, and M. Czigler. Multispectral KLT-wavelet data compression for Landsat Thematic Mapper images. In *Data Compression Conference*, pages 200–208, Snowbird, UT, March 1992.
- [71] K. Falconer. *Fractal Geometry: Mathematical Foundations and Applications*. Wiley, Chichester, 1990.

- [72] J.C. Feauveau. *Analyse multirésolution par ondelettes non-orthogonales et bancs de filtres numériques*. PhD thesis, Université Paris Sud, 1990.
- [73] F. Fisher. *Fractal Image Compression: Theory and Applications*. Springer-Verlag, 1994.
- [74] G. Franceschetti, V. Pascazio, and G. Schirinzi. Iterative homomorphic technique for speckle reduction in synthetic aperture radar imaging. *Journal of the Optical Society of America A*, 12:686–694, 1995.
- [75] W. Frei and C. Chen. Fast boundary detection: a generalization and a new algorithm. *IEEE Transactions on Computers*, C-26:988–998, 1977.
- [76] B.R. Frieden. *Image Enhancement and Restoration*. Springer-Verlag, 1978.
- [77] B.R. Frieden. *Probability, Statistical Optics, and Data Testing: A Problem Solving Approach*. Springer-Verlag, 1978.
- [78] B.R. Frieden. *Physics from Fisher Information*. Cambridge University Press, 1998.
- [79] B. Furht. *GVGIP: Graphical Models and Image Processing*, 55:359–369, 1993.
- [80] B. Furht. A survey of multimedia compression techniques and standards. Part II: Video compression. *Real-Time Imaging*, 1:49–67, 1995.
- [81] N.P. Galatsanos and A.K. Katsaggelos. Methods for choosing the regularization parameter and estimating the noise variance in image restoration and their relation. *IEEE Transactions on Image Processing*, 1:322–336, 1992.
- [82] D. Geman and G. Reynolds. Constrained restoration and the recovery of discontinuities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14(no. 3):367–383, 1992.
- [83] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6(no. 6):721–741, 1984.
- [84] G.H. Golub, M. Heath, and G. Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21:215–223, 1979.
- [85] R.C. Gonzalez and R. Woods. *Digital Image Processing*. Addison-Wesley, 1992.
- [86] J.J. Green and B.R. Hunt. Improved restoration of space object imagery. *Journal of the Optical Society of America A*, 16:2858–2865, 1999.
- [87] A. Grossmann, R. Kronland-Martinet, and J. Morlet. Reading and understanding the continuous wavelet transform. In J.M. Combes, A. Grossmann, and Ph. Tchamitchian, editors, *Wavelets: Time-Frequency Methods and Phase-Space*, pages 2–20. Springer-Verlag, 1989.
- [88] S.F. Gull and J. Skilling. *MEMSYS5 Quantified Maximum Entropy User's Manual*. Royston, England, 1991.
- [89] H.J.A.M. Heijmans and J. Goutsias. Multiresolution signal decomposition schemes. Part 2: Morphological wavelets. *IEEE Transactions on Image Processing*, 9:1897–1913, 2000.
- [90] D..H. Hoekman. Speckle ensemble statistics of logarithmically scaled data. *IEEE Transactions on Geoscience and Remote Sensing*, 29:180–184, 1991.
- [91] J.A. Högbom. Aperture synthesis with a non-regular distribution of interferometer baselines. *Astronomy and Astrophysics Supplement Series*, 15:417–426, 1974.

- [92] M. Holschneider, R. Kronland-Martinet, J. Morlet, and P. Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. In *Wavelets: Time-Frequency Methods and Phase-Space*, pages 286–297. Springer-Verlag, 1989.
- [93] L. Hong, G. Chen, and C.K. Chui. A filter-bank based Kalman filter technique for wavelet estimation and decomposition of random signals. *IEEE Transactions on Circuits and Systems - II Analog and Digital Signal Processing*, 45(2):237–241, 1998.
- [94] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441, 498–520, 1933.
- [95] L. Huang and A. Bijaoui. Astronomical image data compression by morphological skeleton transformation. *Experimental Astronomy*, 1:311–327, 1991.
- [96] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, 1990.
- [97] M. Jansen and D. Roose. Bayesian correction of wavelet threshold procedures for image de-noising. In *Proceedings of the Joint Statistical Meeting, Bayesian Statistical Science*. American Statistical Association, 1998.
- [98] P.A. Jansson, R.H. Hunt, and E.K. Peyler. Resolution enhancement of spectra. *Journal of the Optical Society of America*, 60:596–599, 1970.
- [99] E.T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106:171–190, 1957.
- [100] D. J. Jobson, Z. Rahman, and G. A. Woodell. A multi-scale retinex for bridging the gap between color images and the human observation of scenes. *IEEE Transactions on Image Processing*, 6(7):965–976, 1997.
- [101] D. J. Jobson, Z. Rahman, and G. A. Woodell. Properties and performance of a center/surround retinex. *IEEE Transactions on Image Processing*, 6(3):451–462, 1997.
- [102] K. Karhunen. English translation by I. Selin, On Linear Methods in Probability Theory, The Rand Corporation, Doc. T-131, August 11, 1960, 1947.
- [103] A. K. Katsaggelos. *Digital Image Restoration*. Springer-Verlag, Berlin, 1991.
- [104] E. Kolaczyk. Nonparametric estimation of gamma-ray burst intensities using Haar wavelets. *Astrophysical Journal*, 483:349–349, 1997.
- [105] E. Kolaczyk and D. Dixon. Nonparametric estimation of intensity maps using Haar wavelets and Poisson noise characteristics. *Astrophysical Journal*, 534:490–505, 2000.
- [106] K. Konstantinides, B. Natarajan, and G.S. Yovanov. Noise estimation using block-based singular value decomposition. *IEEE Transactions on Image Processing*, 6:479–483, 1997.
- [107] K. Konstantinides and K. Yao. Statistical analysis of effective singular value in matrix rank determination. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36:757–763, 1988.
- [108] H. Krim, K. Drouiche, and J. C. Pesquet. Multiscale detection of nonstationary signals. In *Time-Frequency and Time-Scale Analysis*, pages 105–108, Victoria, B.C., Canada, 1992. IEEE Signal Processing Society.
- [109] E. Land. Recent advances in retinex theory. *Vision Research*, 26(1):7–21, 1986.
- [110] L. Landweber. An iteration formula for Fredholm integral equations of the first kind. *American Journal of Mathematics*, 73:615–624, 1951.
- [111] A. Lannes and S. Roques. Resolution and robustness in image processing: a new regularization principle. *Journal of the Optical Society of America*, 4:189–199, 1987.

- [112] T.R. Lauer. Combining undersampled dithered images. *Publications of the Astronomical Society of the Pacific*, 111:227–237, 1999.
- [113] H.C. Lee. U.S. Patent 5 010 504, Apr. 1991, 1991. Digital image noise noise suppression method using CVD block transform.
- [114] J. Lee. Optimized quadtree for Karhunen-Loëve transform in multispectral image coding. *IEEE Transactions on Image Processing*, 8:453–461, 1999.
- [115] P. Levy, editor. *Fonctions Aléatoires de Second Ordre*, Paris, 1948. Hermann.
- [116] J. Llacer and J. Núñez. Iterative maximum likelihood estimator and Bayesian algorithms for image reconstruction in astronomy. In R.L. White and R.J. Allen, editors, *The Restoration of HST Images and Spectra*, Baltimore, 1990. Space Telescope Science Institute.
- [117] H. Lorenz and G.M. Richter. Adaptive filtering of HST images: preprocessing for deconvolution. In P. Benvenuti and E. Schreier, editors, *Science with the Hubble Space Telescope*, pages 203–206, Garching, 1993. European Southern Observatory.
- [118] L.B. Lucy. An iteration technique for the rectification of observed distributions. *Astronomical Journal*, 79:745–754, 1974.
- [119] P. Magain, F. Courbin, and S. Sohy. Deconvolution with correct sampling. *Astrophysical Journal*, 494:472, 1998.
- [120] F. Malgouyres. Mathematical analysis of a model which combines total variation and wavelet for image restoration. *Journal of information processes*, 2(1):1–10, 2002.
- [121] S. Mallat. *A Wavelet Tour of Signal Processing*. Academic Press, 1998.
- [122] S. Mallat and W. L. Hwang. Singularity detection and processing with wavelets. *IEEE Transactions on Information Theory*, 38:617–643, 1992.
- [123] S. Mallat and S. Zhong. Characterization of signals from multiscale edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14:710–732, 1992.
- [124] S.G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:674–693, 1989.
- [125] S.G. Mallat. Zero crossings of a wavelet transform. *IEEE Transactions on Information Theory*, 37:1019–1033, 1991.
- [126] B. Mandelbrot. *The Fractal Geometry of Nature*. Freeman, New York, 1983.
- [127] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London B*, 207:187–217, 1980.
- [128] G. Matheron. *Elements pour une Théorie des Milieux Poreux*. Masson, 1967.
- [129] G. Matheron. *Random Sets and Integral Geometry*. Wiley, 1975.
- [130] F. Matus and J. Flusser. Image representations via a finite Radon transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(10):996–1006, 1993.
- [131] R.M. Mersereau and A.V. Oppenheim. Digital reconstruction of multidimensional signals from their projections. *Proc. IEEE*, 62(10):1319–1338, 1974.
- [132] A. Mohammad-Djafari. Maximum entropie et problèmes inverses en imagerie. *Traitemen du Signal*, 11:87–116, 1994.
- [133] A. Mohammad-Djafari. Entropie en traitement du signal. *Traitemen du Signal*, 15:545–551, 1998.

- [134] R. Molina. On the hierarchical Bayesian approach to image restoration. Applications to astronomical images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:1122–1128, 1994.
- [135] P. Moulin and J. Liu. Analysis of multiresolution image denoising schemes using generalized gaussian and complexity priors. *IEEE Transactions on Information Theory*, 45:909–919, 1999.
- [136] F. Murtagh and J.-L. Starck. Multiresolution image analysis using wavelets: Some recent results and some current directions. *ST-ECF Newsletter No. 21*, pages 19–20, 1994.
- [137] F. Murtagh and J.-L. Starck. Pattern clustering based on noise modeling in wavelet space. *Pattern Recognition*, 31(7):847–855, 1998.
- [138] F. Murtagh, J.-L. Starck, and A. Bijaoui. Image analysis using multiresolution: New results. *ST-ECF Newsletter*, 22:6–8, 1995.
- [139] F. Murtagh, J.-L. Starck, and A. Bijaoui. Image restoration with noise suppression using a multiresolution support. *Astronomy and Astrophysics, Supplement Series*, 112:179–189, 1995.
- [140] F. Murtagh, J.-L. Starck, and A. Bijaoui. Multiresolution in astronomical image processing: A general framework. *The International Journal of Image Systems and Technology*, 6:332–338, 1995.
- [141] F. Murtagh, J.-L. Starck, P.F. Honoré, and W. Zeilinger. Multiresolution transforms for object detection and for image transmission. *Vistas in Astronomy*, 40(4):595–602, 1996.
- [142] F. Murtagh, J.-L. Starck, and M. Louys. Very high quality image compression based on noise modeling. *International Journal of Imaging Systems and Technology*, 9:38–45, 1998.
- [143] F. Murtagh, W.W. Zeilinger, and J.-L. Starck. Detection of faint extended structures by multiresolution wavelet analysis. *ESO Messenger*, 73:37–39, 1993.
- [144] R. Narayan and R. Nityananda. Maximum entropy image restoration in astronomy. *Annual Review of Astronomy and Astrophysics*, 24:127–170, 1986.
- [145] G. P. Nason. Wavelet regression by cross-validation. Technical report, Department of Mathematics, University of Bristol, 1994.
- [146] G.P. Nason. Wavelet shrinkage by cross-validation. *Journal of the Royal Statistical Society B*, 58:463–479, 1996.
- [147] B.K. Natarajan. Filtering random noise from deterministic signals via data compression. *IEEE Transactions on Image Processing*, 43:2595–2605, 1995.
- [148] R. Nowak and R. Baraniuk. Wavelet-domain filtering for photon imaging systems. *IEEE Transactions on Image Processing*, 8:666–678, 1999.
- [149] D. Nuzillard and A. Bijaoui. Blind source separation and analysis of multispectral astronomical images. *Astronomy and Astrophysics, Supplement Series*, 147:129–138, November 2000.
- [150] S.I. Olsen. Estimation of noise in images: An evaluation. *CVGIP: Graphical Models and Image Processing*, 55:319–323, 1993.
- [151] G. Paladin and A. Vulpiani. Anomalous scaling laws in multifractal objects. *Physics Reports*, 156:147–225, 1987.
- [152] E. Pantin and J.-L. Starck. Deconvolution of astronomical images using the multiscale maximum entropy method. *Astronomy and Astrophysics, Supplement Series*, 315:575–585, 1996.

- [153] M. Pierre and J.-L. Starck. X-ray structures in galaxy cores. *Astronomy and Astrophysics*, 128:801–818, 1998.
- [154] W.K. Pratt. *Digital Image Processing*. Wiley, 1991.
- [155] W.H. Press. Wavelet-based compression software for FITS images. In D.M. Worrall, C. Biemesderfer, and J. Barnes, editors, *Astronomical Data Analysis Software and Systems I*, pages 3–16. Astronomical Society of the Pacific, 1992.
- [156] J.M.S. Prewitt. Object enhancement and extraction. In B.S. Lipkin and A. Rosenfeld, editors, *Picture Processing and Psychopictorics*. Academic Press, 1970.
- [157] Z. Rahman, D. J. Jobson, and G. A. Woodell. Multi-scale retinex for color image enhancement. In *IEEE International Conference on Image Processing*. IEEE, 1996.
- [158] W.H. Richardson. Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America*, 62:55–59, 1972.
- [159] L.G. Roberts. Machine perception of three-dimensional solids. In J.T. Tippett, editor, *Optical and Electro-Optical Information Processing*. MIT Press, 1965.
- [160] F. Rué and A. Bijaoui. A multiscale vision model to analyse field astronomical images. *Experimental Astronomy*, 7:129–160, 1997.
- [161] M.M. Ruskai, G. Beylkin, R. Coifman, I. Daubechies, S. Mallat, Y. Meyer, and L. Raphael. *Wavelets and Their Applications*. Jones and Barlett, 1992.
- [162] J.A. Saghri, A.G. Tescher, and J.T Reagan. Practical transform coding of multispectral imagery. *IEEE Signal Processing Magazine*, 12:32–43, 1995.
- [163] A. Said and W.A. Pearlman. A new, fast, and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:243–250, 1996.
- [164] Peter Schröder and Wim Sweldens. Spherical wavelets: Efficiently representing functions on the sphere. *Computer Graphics Proceedings (SIGGRAPH 95)*, pages 161–172, 1995.
- [165] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, 1982.
- [166] C.E. Shannon. A mathematical theory for communication. *Bell System Technical Journal*, 27:379–423, 1948.
- [167] J.M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41:3445–3462, 1993.
- [168] M.J. Shensa. Discrete wavelet transforms: Wedding the à trous and Mallat algorithms. *IEEE Transactions on Signal Processing*, 40:2464–2482, 1992.
- [169] L.A. Shepp and Y. Vardi. Maximum likelihood reconstruction for emission tomography. *IEEE Transactions on Medical Imaging*, MI-2:113–122, 1982.
- [170] E.P. Simoncelli, W.T Freeman, E.H. Adelson, and D.J. Heeger. Shiftable multi-scale transforms [or "what's wrong with orthonormal wavelets"]. *IEEE Trans. Information Theory*, 1992.
- [171] J. Skilling. Classic maximum entropy. In *Maximum Entropy and Bayesian Methods*, pages 45–52. Kluwer, 1989.
- [172] E. Slezak, V. de Lapparent, and A. Bijaoui. Objective detection of voids and high-density structures in the first CfA redshift survey slice. *Astrophysical Journal*, 409:517–529, 1993.

- [173] E. Slezak, V. de Lapparent, and A. Bijaoui. Objective detection of voids and high density structures in the first CfA redshift survey slice. *Astrophysical Journal*, 409:517–529, 1993.
- [174] E. Slezak, F. Durret, and D. Gerbal. A wavelet analysis search for substructures in eleven X-ray clusters of galaxies. *Astrophysical Journal*, 108:1996–2008, 1994.
- [175] P. Soille. *Morphological Data Analysis*. Springer-Verlag, 1999.
- [176] S. Soltani, D. Boichu, P. Simard, and S. Canu. The long-term memory prediction by multiscale decomposition. *Signal Processing*, 80:2195–2205, 2000.
- [177] J.-L. Starck. *Analyse en ondelettes et Haute Résolution Angulaire*. PhD thesis, Université de Nice Sophia Antipolis, 4 December 1992.
- [178] J.-L. Starck. The wavelet context. Technical report, European Southern Observatory – Image Processing Group, Release 93NOV 1993.
- [179] J.-L. Starck. Image processing by the curvelet transform. In R. Leonardi, editor, *2002 Tyrrhenian International Workshop on Digital Communications, Advanced Methods for Multimedia Signal Processing*. SPIE, 2002.
- [180] J.-L. Starck, H. Aussel, D. Elbaz, D. Fadda, and C. Cesarsky. Faint source detection in ISOCAM images. *Astronomy and Astrophysics, Supplement Series*, 138:365–379, 1999.
- [181] J.-L. Starck and A. Bijaoui. Filtering and deconvolution by the wavelet transform. *Signal Processing*, 35:195–211, 1994.
- [182] J.-L. Starck and A. Bijaoui. Multiresolution deconvolution. *Journal of the Optical Society of America A*, 11(4), 1994.
- [183] J.-L. Starck, A. Bijaoui, B. Lopez, and C. Perrier. Image reconstruction by the wavelet transform applied to aperture synthesis. *Astronomy and Astrophysics*, 283:349–360, 1994.
- [184] J.-L. Starck, A. Bijaoui, and F. Murtagh. Multiresolution support applied to image filtering and deconvolution. *CVGIP: Graphical Models and Image Processing*, 57:420–431, 1995.
- [185] J.-L. Starck, E. Candès, and D.L. Donoho. The curvelet transform for image denoising. *IEEE Transactions on Image Processing*, 11(6):131–141, 2002.
- [186] J.-L. Starck, D.L. Donoho, and E. Candès. Very high quality image restoration. In A. Laine, M.A. Unser, and A. Aldroubi, editors, *SPIE conference on Signal and Image Processing: Wavelet Applications in Signal and Image Processing IX, San Diego, 1-4 August*. SPIE, 2001.
- [187] J.-L. Starck and F. Murtagh. Image restoration with noise suppression using the wavelet transform. *Astronomy and Astrophysics*, 288:343–348, 1994.
- [188] J.-L. Starck and F. Murtagh. Multiresolution image analysis using wavelets: recent results. *Bulletin of the American Astronomical Society*, 26(2):1003–1005, 1994.
- [189] J.-L. Starck and F. Murtagh. Automatic noise estimation from the multiresolution support. *Publications of the Astronomical Society of the Pacific*, 110:193–199, 1998.
- [190] J.-L. Starck and F. Murtagh. *Astronomical Image and Data Analysis*. Springer-Verlag, 2002.
- [191] J.-L. Starck, F. Murtagh, and A. Bijaoui. *Image Processing and Data Analysis: The Multiscale Approach*. Cambridge University Press, 1998.
- [192] J.-L. Starck, F. Murtagh, E. Candes, and D.L. Donoho. Gray and color image contrast enhancement by the curvelet transform. *IEEE Transactions on Image Processing*, 2002. submitted.

- [193] J.-L. Starck, F. Murtagh, and D. Durand. New results in astronomical image compression. *NASA Science Information Systems Newsletter*, 2(39), 1996.
- [194] J.-L. Starck, F. Murtagh, and R. Gastaud. A new entropy measure based on the wavelet transform and noise modeling. *IEEE Transactions on Circuits and Systems II*, 45:1118–1124, 1998.
- [195] J.-L. Starck, F. Murtagh, B. Pirenne, and M. Albrecht. Astronomical image compression based on noise suppression. *Publications of the Astronomical Society of the Pacific*, 108:446–455, 1996.
- [196] J.-L. Starck, E. Pantin, and F. Murtagh. Deconvolution in astronomy: a review. *Publications of the Astronomical Society of the Pacific*, 2002. in press.
- [197] J.-L. Starck and M. Pierre. Structure detection in low intensity X-ray images. *Astronomy and Astrophysics, Supplement Series*, 128, 1998.
- [198] J.-L. Starck, R. Siebenmorgen, and R. Grede. Spectral analysis by the wavelet transform. *Astrophysical Journal*, 482:1011–1020, 1997.
- [199] G. Strang and T. Nguyen. *Wavelet and Filter Banks*. Wellesley-Cambridge Press, 1996.
- [200] W. Sweldens. The lifting scheme: A construction of second generation wavelets. *SIAM Journal on Mathematical Analysis*, 29:511–546, 1997.
- [201] W. Sweldens and P. Schröder. Building your own wavelets at home. In *Wavelets in Computer Graphics*, pages 15–87. ACM SIGGRAPH Course notes, 1996.
- [202] A.N. Tikhonov, A.V. Goncharski, V.V. Stepanov, and I.V. Kochikov. Ill-posed image processing problems. *Soviet Physics – Doklady*, 32:456–458, 1987.
- [203] K. E. Timmermann and R. Nowak. Multiscale modeling and estimation of Poisson processes with applications to photon-limited imaging. *IEEE Transactions on Signal Processing*, 46:886–902, 1999.
- [204] D.M. Titterington. General structure of regularization procedures in image reconstruction. *Astronomy and Astrophysics*, 144:381–387, 1985.
- [205] A. Toet. Multiscale color image enhancement. In *SPIE International Conference on Image Processing and its Applications*, pages 583–585. SPIE, 1992.
- [206] D. Tretter and C. Bouman. Optimal transforms for multispectral and multilayer image coding. *IEEE Transactions on Image Processing*, 4:308–396, 1995.
- [207] M. Unser, P. Thévenaz, and L. Yaroslavsky. Convolution-based interpolation for fast, high quality rotation of images. *IEEE Transactions on Image Processing*, 4:1371–1381, 1995.
- [208] K.V. Velde. Multi-scale color image enhancement. In *Proceedings of the International Conference on Image Processing*, volume 3, pages 584–587. ICIP, 1999.
- [209] J. Ville. Théorie et applications de la notion de signal analytique. *Cables et Transmissions*, 2A:61–74, 1948.
- [210] B.P. Wakker and U.J. Schwarz. The multi-resolution Clean and its application to the short-spacing problem in interferometry. *Annual Reviews of Astronomy and Astrophysics*, 200:312, 1988.
- [211] A.B. Watson, G.Y. Yang, J.A. Solomon, and J. Villasenor. Visibility of wavelet quantization noise. *IEEE Transactions on Image Processing*, 6:1164–1174, 1997.
- [212] N. Weir. Application of maximum entropy techniques to HST data. In *3rd ESO/ST-ECF Data Analysis Workshop*, 1991.

- [213] N. Weir. A multi-channel method of maximum entropy image restoration. In D.M. Worrall, C. Biemesderfer, and J. Barnes, editors, *Astronomical Data Analysis Software and System 1*, pages 186–190. Astronomical Society of the Pacific, 1992.
- [214] R. White, M. Postman, and M. Lattanzi. Compression of the Guide Star digitised Schmidt plates. In H.T. MacGillivray and E.B. Thompson, editors, *Digitized Optical Sky Surveys*, pages 167–175. Kluwer, 1992.
- [215] E.P. Wigner. On the quantum correction for thermodynamic equilibrium. *Physical Review*, 40:749–759, 1932.
- [216] Isao Yamada. The hybrid steepest descent method for the variational inequality problem over the intersection of fixed point sets of nonexpansive mappings. In D. Butnariu, Y. Censor, and S. Reich, editors, *Inherently Parallel Algorithms in Feasibility and Optimization and Their Applications*. Elsevier, 2001.
- [217] G. Zheng, J.-L. Starck, J. Campbell, and F. Murtagh. The wavelet transform for filtering financial data streams. *the Journal of Computational Intelligence in Finance*, 7(3):18–35, 1999.
- [218] M. Zibulevsky and B.A. Pearlmutter. Blind source separation by sparse decomposition in a signal dictionary. *Neural Computation*, 13:863–882, 2001.
- [219] M. Zibulevsky and Y.Y. Zeevi. Extraction of a single source from multichannel data using sparse decomposition. Technical report, Faculty of Electrical Engineering, Technion – Israel Institute of Technology, 2001.

Index

- mw_deconv, 278
a trous wavelet transform, 140, 255
Anscombe transformation, 285, 286

blind source separation, 340

cb_filter, 396
CEA_FILTER, 49
CEA_LICENSE_HOST, 17
cea_lm, 17
CEA_MR_DIR, 18
CEA_VM_DIR, 19
CEA_VM_SIZE, 19
CLEAN, 250–253
clean beam, 251
clean map, 251
closing, 35
coaddition, 300
col_colcontrast, 395
col_comp, 306
col_contrast, 308
col_decomp, 307
col_filter, 305
color images, 305
compression, 93
continuous wavelet transform, 39
Cosmic Microwave Background, 340
cur_colfilter, 393
cur_contrast, 394
cur_filter, 393
cur_stat, 392
cur_trans, 391

data
 3C120, 340
deconvolution, 87, 247, 253, 254, 302
detection, 109
dilation, 34
dirty map, 250

edge detection, 129
entropy, 205
erosion, 34

fft, 30
fft_cf_2d, 31
fft_f_2d, 31
filtering, 63, 221, 316, 332

format, 19
fourier, 30
Fourier transform, 250, 252
fractal, 165

ICA, 340
im1d_convert, 143
im1d_info, 143
im1d_stf, 161
im1d_tend, 145
im1d_tfreq, 163
im3d_coadd, 300
im3d_convert, 297
im3d_deconv, 303
im3d_get, 298
im3d_info, 298
im3d_op, 299
im3d_put, 299
im3d_simu, 299
im_closing, 35
im_convert, 21
im_dct, 32
im_deconv, 87
im_dilate, 34
im_dst, 32
im_edge, 134
im_erode, 34
im_get, 24
im_info, 21
im_mirror, 27
im_op, 24
im_opening, 35
im_pca, 36
im_pca_rec, 37
im_put, 24
im_radon, 387
im_random, 25
im_rot, 28
im_segment, 27
im_simu, 26
im_snr, 23
im_thin, 35
im_threshold, 27
independent component analysis, 340
intrinsic correlation function, 210

JADE, 340

Kullback-Leibler, 340
 Mallat's multiresolution, 42, 166, 170
 mathematical morphology, 33
 maximum entropy method, 210
 MEM, 210
 mf1d_chain, 173
 mf1d_create_ds, 173
 mf1d_repart, 174
 mf_analyse, 174
 mr1d_acor, 158
 mr1d_detect, 151
 mr1d_filter, 149
 mr1d_max, 153
 mr1d_maxrecons, 153
 mr1d_nowcast, 158, 159
 mr1d_recons, 149
 mr1d_trans, 146
 mr3d_cat2fits, 319
 mr3d_extract, 315
 mr3d_filter, 316
 mr3d_insert, 315
 mr3d_pfilter, 321
 mr3d_phisto, 319
 mr3d_psupport, 320
 mr3d_recons, 316
 mr3d_trans, 314
 mr_abaque, 78
 mr_at_edge, 138
 mr_background, 57
 mr_comp, 100
 mr_compare, 57
 mr_contrast, 140
 mr_decomp, 105
 mr_deconv, 89
 mr_detect, 113
 mr_edge, 137
 mr_extract, 50
 mr_filter, 67
 mr_fusion, 122
 mr_hfilter, 84
 mr_info, 54
 mr_insert, 50
 mr_lcomp, 103
 mr_mrc, 91
 mr_pfilter, 80
 mr_psupport, 79
 mr_rec_edge, 138
 mr_recons, 51
 mr_rfilter, 73, 74
 mr_segment, 55
 mr_sigma, 56
 mr_support, 52
 mr_transform, 42
 mr_upresol, 106
 mr_visu, 51
 multi-channel, 323
 multiresolution support, 16, 52, 256
 Multiscale, 39
 multiscale entropy, 214
 mw1d_filter, 274
 mw_comb_filter, 276
 mw_entrop, 273
 mw_filter, 275
 mw_proba, 273
 noise, 255, 256, 285, 286
 Principal component analysis, 36
 pyramid, 210, 211
 registration, 121
 regularization, 209
 regularization, Tikhonov, 249
 restoration, 61
 Richardson-Lucy deconvolution, 249, 255, 258
 rid_stat, 389
 rid_trans, 388
 rotation, 28
 skeleton, 35
 speckle, 73
 stationary signal, 64, 247
 support, multiresolution, 16, 256
 Tikhonov regularization, 249
 transform
 Wigner-Ville transform, 163
 Van Cittert deconvolution, 248, 254, 255, 257
 variance stabilization, 286
 wavelet transform, 59, 140, 252, 255, 285, 286
 wavelet transform, continuous, 39
 wavelet transform, Morlet-Grossman, 39
 wcomp, 99
 Wigner-Ville transform, 163
 wk1d_filter, 334
 wk1d_trans, 327
 wk1d_trec, 327
 wk_filter, 335
 wk_memfilter, 336
 wk_trans, 328
 wk_trec, 330
 ww_filter, 338