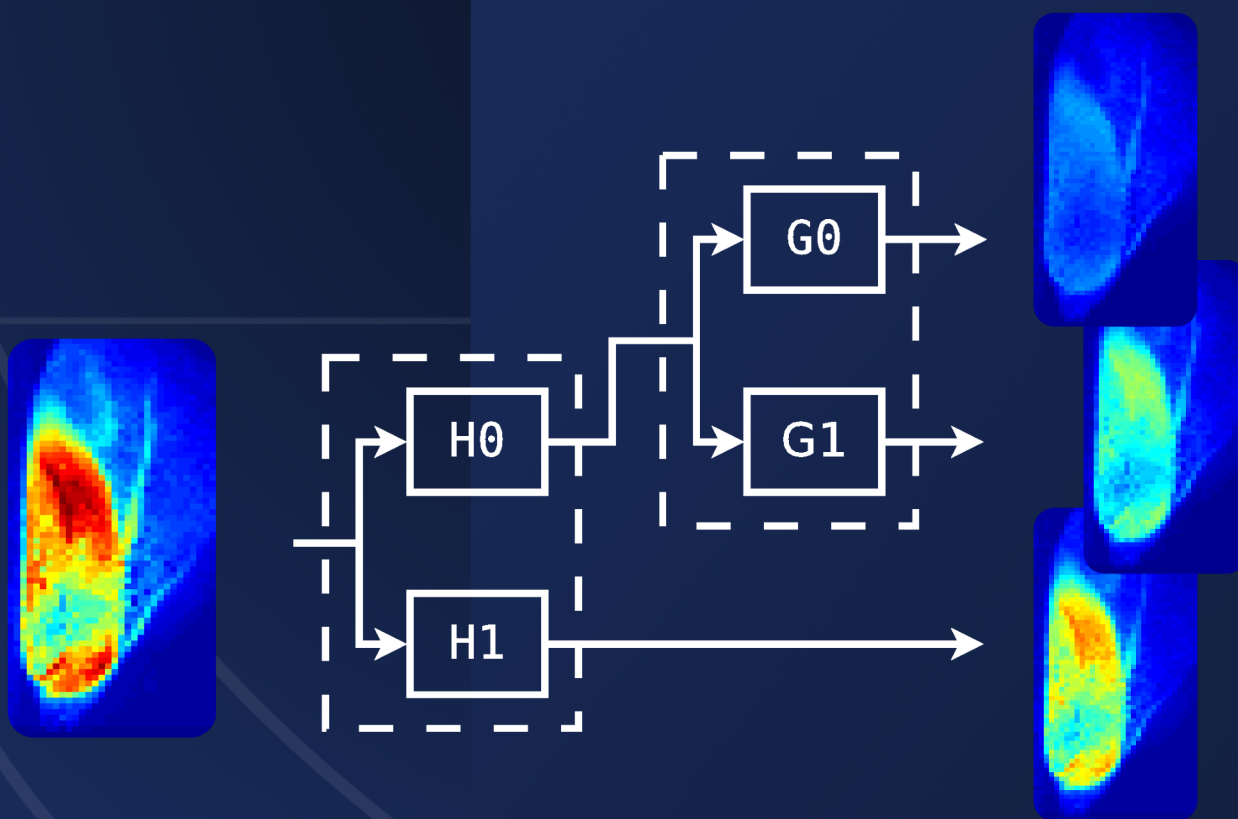


A Practical Guide to **Biospeckle Laser Analysis** *Theory and Software*



Roberto Alves Braga Júnior
Fernando Pujaico Rivera
Junio Moreira

A Practical Guide to Biospeckle Laser Analysis

Theory and Software



Roberto Alves Braga Júnior
Fernando Pujaico Rivera
Junio Moreira

A Practical Guide to Biospeckle Laser Analysis

Theory and Software



© 2016 by Roberto Alves Braga Júnior, Fernando Pujaico Rivera and Junio Moreira
Nenhuma parte desta publicação pode ser reproduzida, por qualquer meio ou forma, sem a
autorização escrita e prévia dos detentores do copyright.
Direitos de publicação reservados à Editora UFLA.
Impresso no Brasil – ISBN: 9-788581-270517

UNIVERSIDADE FEDERAL DE LAVRAS

Reitor: José Roberto Soares Scolforo

Vice-Reitora: Édila Vilela de Resende Von Pinho



Editora UFLA

Campus UFLA - Pavilhão 5

Caixa Postal 3037 – 37200-000 – Lavras – MG

Tel: (35) 3829-1532 – Fax: (35) 3829-1551

E-mail: editora@editora.ufla.br

Homepage: www.editora.ufla.br

Diretoria Executiva: Marco Aurélio Carbone Carneiro (Diretor) e Nilton Curi (Vice-Diretor)

Conselho Editorial: Marco Aurélio Carbone Carneiro (Presidente), Nilton Curi, Francisval de Melo Carvalho, Alberto Colombo, João Domingos Scalon, Wilson Magela Gonçalves.

Administração: Flávio Monteiro de Oliveira

Secretária: Késia Portela de Assis

Comercial/Financeiro: Damiana Joana Geraldo Souza

Revisão de Texto em Inglês: Heloisa Helena Nogueira Chaves

Referências Bibliográficas: Editora UFLA

Editoração Eletrônica: Fernando Pujaico Rivera, Roberto Alves Braga Júnior

Capa: Fernando Pujaico Rivera

**Ficha catalográfica elaborada pela Coordenadoria de Processos
Técnicos da Biblioteca Universitária da UFLA**

Braga Júnior, Roberto Alves.

A practical guide to Biospeckle Laser analysis: theory and
software / Roberto Alves Braga Júnior, Fernando Pujaico
Rivera, Junio Moreira. – Lavras : Ed. UFLA, 2016.

158 p. : il.

Bibliografia.

1. Dynamic Laser Speckle. 2. Tutorial. 3. M code. 4. Signal
Digital Filtering. 5. Numerical and Graphical Methods. I.
Rivera, Fernando Pujaico. II. Moreira, Junio. III. Título.

CDD – 621.366

To my wife Vanessa and sons Matheus and Lucas.

Roberto

Amicum lege feliciter, vivas, gaudeas, floreas in Deo.

Fernando

*I dedicate this work to my family, particularly to my mother Iraci, and friends
for all the support, understanding and assistance they have given me.*

Junio

*This book was only possible because of the support of
UFLA, Fapemig, Capes, CNPq and Finep during the last
20 years.*

A foreword to “A Practical Guide to Biospeckle Laser Analysis”

This book is an offshoot of a meeting with Roberto and me, when he visited the UK in 2011 as part of a UK-Brazil collaborative research program. We have been involved in the development of laser based instrumentation for more than 15 years and shared similar thoughts, but, in different parts of the world. Since then, we had several face-to-face and online meetings and discussions on Laser Biospeckle and image processing. Through this book, Roberto and his co-authors address some of the common practical issues in the area of Laser Biospeckle.

Laser Speckle is a random intensity pattern produced by many light waves of different phase and amplitude but with the same frequency. During the initial stages of laser development and imaging, this was considered to be a noise as it produces grainy or glittery type images. The famous saying “All that glitters is not Gold” was fitting properly with the Laser Speckle phenomena. But, later on, these “glitters” were found to be carrying “golden” information about the surface and near-surfaces and lead to the emergence of various non-destructive type imaging applications. This book is an outgrowth of two decades of research and teaching by the authors on the development of laser speckle based systems. Again, various technical presentations and discussions with applied researchers, including me, have opened-up the avenue for this practical guide on Biospeckle Laser. The objective of this guide is to introduce fundamental concepts of laser speckle from applications specific to Biosystems and to develop a basic understanding on the relevant software and image processing techniques, smoothly.

In this book, which is mostly based on case studies and experiments with a flavour of theory, the authors have tried to cover most of the practical difficulties experienced in the application of the method and keep the theoretical data to a minimum. In most of the cases the authors use examples to illustrate specific points, especially where data or software interpretation are involved. For those who are new to biospeckle systems, will find little difficulty in keeping up-to-date in the field as many developments emerge regularly. The authors have made an attempt to keep the literature/developments up-to-date.

This is a very useful book for those who are working in the areas of biospeckle and their applications. It is well structured, providing experimental and theoretical details needed to understand and develop laser based biospeckle systems. The first part of the guide provides a step-by-step approach for setting-up the systems along with the details on the software interface for data collection and analysis. In the second part, a detailed list of library software tools provides the various data analysing functions and their effects with realistic examples. This will help the user to develop or carryout user specific modifications based on their experimental needs. The BSL tool library reference manual helps to get a quick insight into these functions. The high quality pictures and figures, mostly based on practical examples, provide better understanding and easy learning. Overall, this book provides a guide for developing biospeckle laser systems with a practical touch.

Radhakrishna Prabhu

Dy. Director for Northern Research Partnership MedTech JRI and Lecturer in School of Engineering, Robert Gordon University Aberdeen, UK.

Contents

I	Part 1: Theory	
1	Experimental Guidelines	25
1.1	Introduction	25
1.2	Biospeckle Laser System	27
1.3	Optical Setup - Hardware	28
1.4	On-line Adjustments	31
1.5	Quality Tests	31
1.5.1	Saturation and Under-exposition Tests	31
1.5.2	Contrast Test	33
1.5.3	Homogeneity Test	35
1.6	Methods of Data Analysis - On-line Methods	37
1.6.1	MHI	37
1.7	Methods of Data Analysis - Off-line Methods	40
1.7.1	Graphic Methods	40
1.7.2	Numerical Methods	43
1.7.3	Frequency Analysis	47
1.8	Frequently Asked Questions - FAQ	52
1.8.1	How Should I Mount my Experiment?	52
1.8.2	How Should I Analyze my Data?	53
1.9	Exercises	53

2	Introduction	57
2.1	Overview of BSL Tool Library	57
2.2	News	57
3	Getting Started with BSL Tool Library	59
3.1	User and System Requirements	59
3.1.1	User Requirements	59
3.1.2	System Requirements	60
3.2	Good Programming Practices	60
3.2.1	Where Must I Place my Libraries?	61
3.2.2	How to Make a New Project?	63
3.2.3	How to Load my Libraries?	63
3.2.4	Other Recommendations	65
3.3	Installing BSL Tool Library	66
3.3.1	Method 1: Online - Only in OCTAVE	66
3.3.2	Method 2: Offline - Only in OCTAVE	66
3.3.3	Method 3: In MATLAB or OCTAVE	66
4	BSL Tool Library <i>cookbook</i>	69
4.1	Image Datapack	69
4.2	Time History Speckle Pattern	71
4.3	Co-Occurrence Matrix	72
4.4	Probability Mass Function	73
4.4.1	Probability Mass Function of Absolute Difference	73
4.4.2	Probability Mass Function of Regular Difference	74
4.5	Numerical Methods	75
4.5.1	Inertia Moment	75
4.5.2	Absolute Value of the Differences	76
4.5.3	Regular Value of the Differences	78
4.5.4	Numerical Analysis of the Average Differences	80
4.5.5	Time History Speckle Pattern to Correlation Coefficient	81
4.5.6	Spatial-Temporal Speckle Correlation	82
4.6	Graphic Methods	83
4.6.1	Average Difference or Fujii Method	83
4.6.2	Generalized Difference Method - GD	83
4.6.3	Temporal Speckle Contrast - Standard Deviation - Mean's Method	84
4.6.4	Inertia Moment Method in Graphic Mode	85
4.6.5	Absolute Value of the Differences Method in Graphic Mode	85
4.6.6	Regular Value of the Differences Method in Graphic Mode	86
4.6.7	Parametrized form of Temporal Difference Method	87
4.6.8	Temporal Speckle Kurtosis Matrix	87
4.6.9	Temporal Speckle Skewness Matrix	88
4.6.10	Motion History Image	90

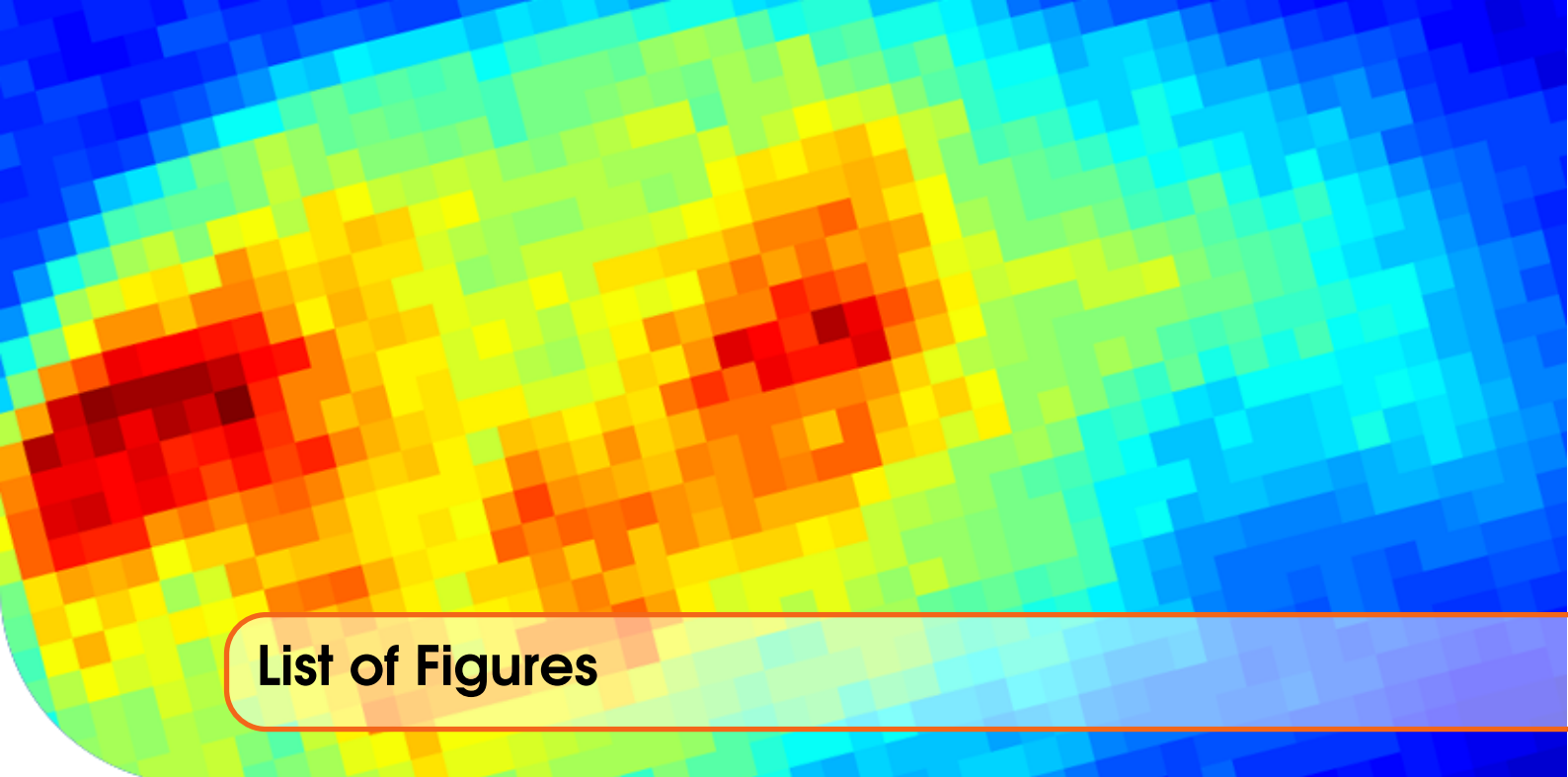
4.7	Quality Tests	91
4.7.1	Analyzing the Saturation and the Sub-exposition of Light	91
4.7.2	Spatial Speckle Contrast Window Method	91
4.7.3	Homogeneity of Spatial Variability	92
4.8	Frequency Analysis	93
4.8.1	Filtering in Frequency Bands	93
4.8.2	Decomposition Using the Discrete Wavelet Transform	96
4.8.3	Reconstruction Using the Inverse Discrete Wavelet Transform	97
4.8.4	Partial Reconstruction Using The Inverse Discrete Wavelet Transform	99
4.8.5	Filtering Using the Continuous Wavelet Transform of a Discrete Signal	100
4.9	Extra Functions	103
4.9.1	Mean Values in Analysis Windows	103
4.9.2	Matrix Threshold	105
4.9.3	Binary Entropy of Probability Mass Function	106
5	Additional Informations	109
5.1	Frequently Asked Questions	109
5.2	Interesting Links	110
5.3	BSL - Picture Packages	110

III

Part 3: BSL Tool Library Reference Manual

6	Functions' Descriptions	113
6.1	Loading the Datapack	113
6.1.1	Function datapack() - Creating a 3D Matrix	113
6.1.2	Function datacut() - Selecting and Cutting the Image Datapack	114
6.1.3	Function datapack_to_gif() - Datapack to GIF file	115
6.2	Creating the Time History Speckle Pattern	116
6.2.1	Function thsp() - Time History Speckle Pattern	116
6.2.2	Function thsp_line() - Time History Speckle Pattern of a Line	118
6.2.3	Function thsp_random() - Time History Speckle Pattern of a Random Points Set	118
6.2.4	Function thsp_gaussian() - Time History Speckle Pattern of a Random Points Set with Gaussian Distribution	118
6.3	Creating and Working with the Co-occurrence Matrix	119
6.3.1	Function coom() - Co-occurrence Matrix	120
6.3.2	Function pmfad() - Probability Mass Function of the Absolute Difference	121
6.3.3	Function pmfrd() - Probability Mass Function of the Regular Difference	121
6.4	Calculating the Inertia Moment	122
6.4.1	Function inertiamoment() - Inertia Moment Method	123
6.5	Calculating the Absolute Value of the Differences	124
6.5.1	Function avd() - Absolute Value of the Differences Method	125
6.6	Calculating the Regular Value of the Differences	126
6.6.1	Function rvd() - Regular Value of the Differences Method	127
6.7	Numerical Analysis of the Average Difference Method	128
6.7.1	Function numad() - Numerical Analysis of the Average Difference Method	128

6.8	Calculating the Correlation in the Speckle Pattern	129
6.8.1	Function thsp2corr() - Time History Speckle Pattern to Correlation Coefficient	130
6.8.2	Function stscorr() - Spatial-Temporal Speckle Correlation	131
6.9	Graphical Methods	131
6.9.1	Function fujii() - Fujii Method	131
6.9.2	Function gendiff() - Generalized Differences Method	133
6.9.3	Function stdcont() - Contrast - Deviation - Mean Method	133
6.9.4	Function graphim() - Inertia Moment Method in Graphic Mode	135
6.9.5	Function graphavd() - AVD Method in Graphic Mode	135
6.9.6	Function graphrvd() - RVD Method in Graphic Mode	136
6.9.7	Function graphptd() - Parametrized form of Temporal Difference Method	137
6.9.8	Function graphkurt() - Temporal Speckle Kurtosis Matrix	138
6.9.9	Function graphskew() - Temporal Speckle Skewness Matrix	139
6.9.10	Function graphmhi() - Motion History Image	140
6.10	Frequency Analysis	141
6.10.1	Function datapack_conv() - Convolution of Datapack	141
6.10.2	Function firfilterbank() - Step of a FIR Filter Bank	142
6.10.3	Function firsynthesisbank() - Step of a FIR Filter Bank in Synthesis Mode	144
6.10.4	Function firsynthesispath() - Synthesis of a Path in a Filter Bank	145
6.10.5	Function freqmod() - Modulus of Frequency Response	147
6.10.6	Function qmfmaker() - Quadrature Mirror Filter Maker	148
6.10.7	Function qmfmirror() - Mirror of a Filter in a Quadrature Mirror Filter Scheme	149
6.11	Extra Functions	150
6.11.1	Function hbpmf() - Binary Entropy of a Probability Mass Function	150
6.11.2	Function threshold2d() - Threshold of 2D Matrix	151
6.11.3	Function mwindowing() - Mean Values in the Windows of a Matrix	151
6.12	Quality Tests	152
6.12.1	Function satdark() - Analyzing Saturation and Sub-Exposition of Light	152
6.12.2	Function sscont() - Spatial Speckle Contrast Window Method	152
6.12.3	Function homogeneity() - Homogeneity of Spatial Variability	153
	Bibliography	157
	Books	157
	Sites	159
	Index	161



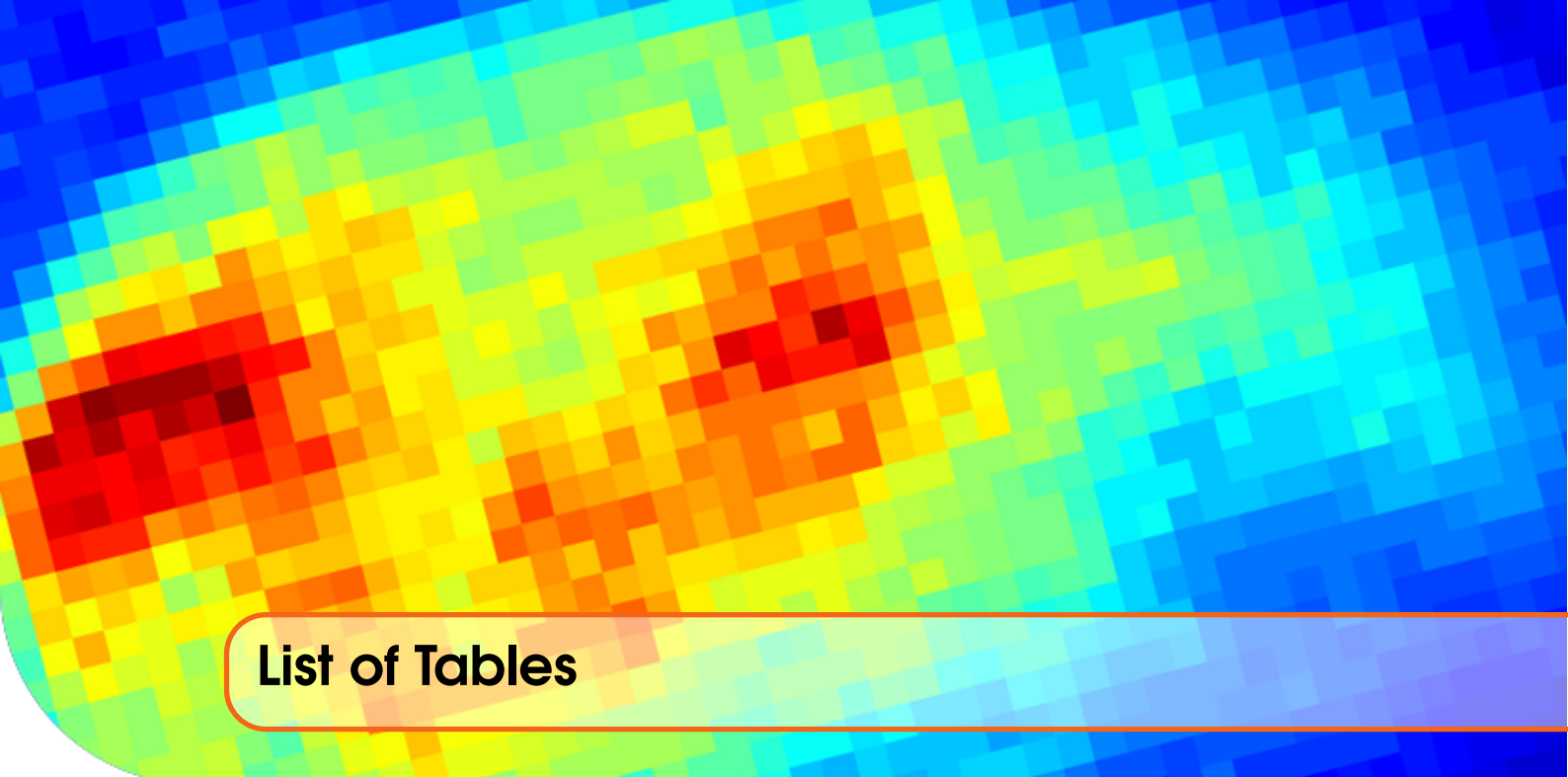
List of Figures

1.1	Speckle pattern of a seed on a table.	26
1.2	Disposition of the book in layers such as an onion. Label I means the first part with an overlook of the theme and comments about the approaches. Label II means the second part with the User Guide with information about the procedures you can use to analyze the data. Label III means the third part, the Reference Guide of the software with the details of all the procedures presented in the two other parts.	27
1.3	First image of a maize seed illuminated by a HeNe laser beam.	27
1.4	Biospeckle laser analysis adopting time and frequency domain approaches, using graphical or numerical outcomes by means of on-line or off-line procedures.	28
1.5	Forward-scattering approach with a basic disposition.	29
1.6	Backscattering approach with a basic disposition.	29
1.7	Image of a maize seed with two different regions of interest (c and d) on the seed. <i>Reprinted from Publication Optics and Lasers in Engineering, 61, Junio Moreira, R. R. Cardoso, R. A. Braga, Quality test protocol to dynamic laser speckle analysis, Pages No. 8-13, Copyright (2014), with permission from Elsevier.</i>	32
1.8	Histogram of the region of interest tagged by the letter c where the distribution of the levels of gray does not present over-exposition of the laser on the seed. <i>Reprinted from Publication Optics and Lasers in Engineering, 61, Junio Moreira, R. R. Cardoso, R. A. Braga, Quality test protocol to dynamic laser speckle analysis, Pages No. 8-13, Copyright (2014), with permission from Elsevier.</i>	32
1.9	Histogram of the region of interest tagged by the letter d where the distribution of the levels of gray presents over-exposition of the laser on the seed. <i>Reprinted from Publication Optics and Lasers in Engineering, 61, Junio Moreira, R. R. Cardoso, R. A. Braga, Quality test protocol to dynamic laser speckle analysis, Pages No. 8-13, Copyright (2014), with permission from Elsevier.</i>	33
1.10	Saturation test map with values of gray levels presenting the over-exposition of the laser on the seed in the region of interest in gray. The numbers mean the pixels over 50% over the upper limit. <i>Reprinted from Publication Optics and Lasers in Engineering, 61, Junio Moreira, R. R. Cardoso, R. A. Braga, Quality test protocol to dynamic laser speckle analysis, Pages No. 8-13, Copyright (2014), with permission from Elsevier.</i>	33

1.11	Saturation and under-exposition tests presenting the over-exposition of the laser on the seed.	34
1.12	Speckle patterns in different instants presenting the blurring at the first instant. <i>Reprinted from Publication Optics and Lasers in Engineering, 61, Junio Moreira, R. R. Cardoso, R. A. Braga, Quality test protocol to dynamic laser speckle analysis, Pages No. 8-13, Copyright (2014), with permission from Elsevier.</i>	34
1.13	Contrast outcome test in a maize seed on analysis windows of 30 by 30 pixels.	35
1.14	Result of applying the Inertia Moment indicator on analysis windows of 16 by 16 pixels.	36
1.15	Homogeneity outcome test in a maize seed.	36
1.16	Image of a person with his arm lifted after moving it. <i>Reprinted from Optics and Lasers in Engineering, 50, 3, R. P. Godinho, M. M. Silva, J. R. Nozela, R. A. Braga, On-line biospeckle assessment without loss of definition and resolution by motion history image, Pages 366-372, Copyright (2012), with permission from Elsevier.</i>	37
1.17	Subtraction of two consecutive images of a person moving his arm. <i>Reprinted from Optics and Lasers in Engineering, 50, 3, R. P. Godinho, M. M. Silva, J. R. Nozela, R. A. Braga, On-line biospeckle assessment without loss of definition and resolution by motion history image, Pages 366-372, Copyright (2012), with permission from Elsevier.</i>	38
1.18	Threshold of the subtraction of two consecutive images of a person moving his arm. <i>Reprinted from Optics and Lasers in Engineering, 50, 3, R. P. Godinho, M. M. Silva, J. R. Nozela, R. A. Braga, On-line biospeckle assessment without loss of definition and resolution by motion history image, Pages 366-372, Copyright (2012), with permission from Elsevier.</i>	38
1.19	Motion history of consecutive images presenting the evolution of the movement of an arm from bottom to top, thus presenting the last movement as clear gray. <i>Reprinted from Optics and Lasers in Engineering, 50, 3, R. P. Godinho, M. M. Silva, J. R. Nozela, R. A. Braga, On-line biospeckle assessment without loss of definition and resolution by motion history image, Pages 366-372, Copyright (2012), with permission from Elsevier.</i>	39
1.20	Motion history of consecutive images presenting the evolution of the movement of an arm from bottom to top, thus presenting the last movement as red. <i>Reprinted from Optics and Lasers in Engineering, 50, 3, R. P. Godinho, M. M. Silva, J. R. Nozela, R. A. Braga, On-line biospeckle assessment without loss of definition and resolution by motion history image, Pages 366-372, Copyright (2012), with permission from Elsevier.</i>	39
1.21	Motion history of consecutive images of the maize seed.	40
1.22	Mounted image of a coffee seedling with roots analyzed by the biospeckle laser presenting the map of activity. In the map, the pseudo-colors mean the level of activity from blue, low activity, to red, high activity.	41
1.23	Map of activities on a maize seed using the Fujii method.	42
1.24	Map of activities in a maize seed using the DG method.	42
1.25	Time History Speckle Pattern (<i>THSP</i>) formed from a collection of images of an animal sperm with high activity. The center line of each image was picked up, rotated and allocated side by side forming the <i>THSP</i> with the columns representing the history of the pixels along the time.	44
1.26	Time History Speckle Pattern (<i>THSP</i>) formed from a collection of images of an animal sperm with low activity. The center line of each image was picked up, rotated and allocated side by side forming the <i>THSP</i> with the columns representing the history of the pixels along the time.	44
1.27	Co-occurrence matrix of a <i>THSP</i> formed from a collection of images of an animal sperm with high activity.	45
1.28	Co-occurrence matrix of a <i>THSP</i> formed from a collection of images of an animal sperm with low activity.	45
1.29	Filtering in frequency bands using a filter banks.	48
1.30	Decomposition in frequency bands and reconstruction using <i>DWT</i>	49
1.31	Decomposition in frequency bands using <i>CWT</i>	51

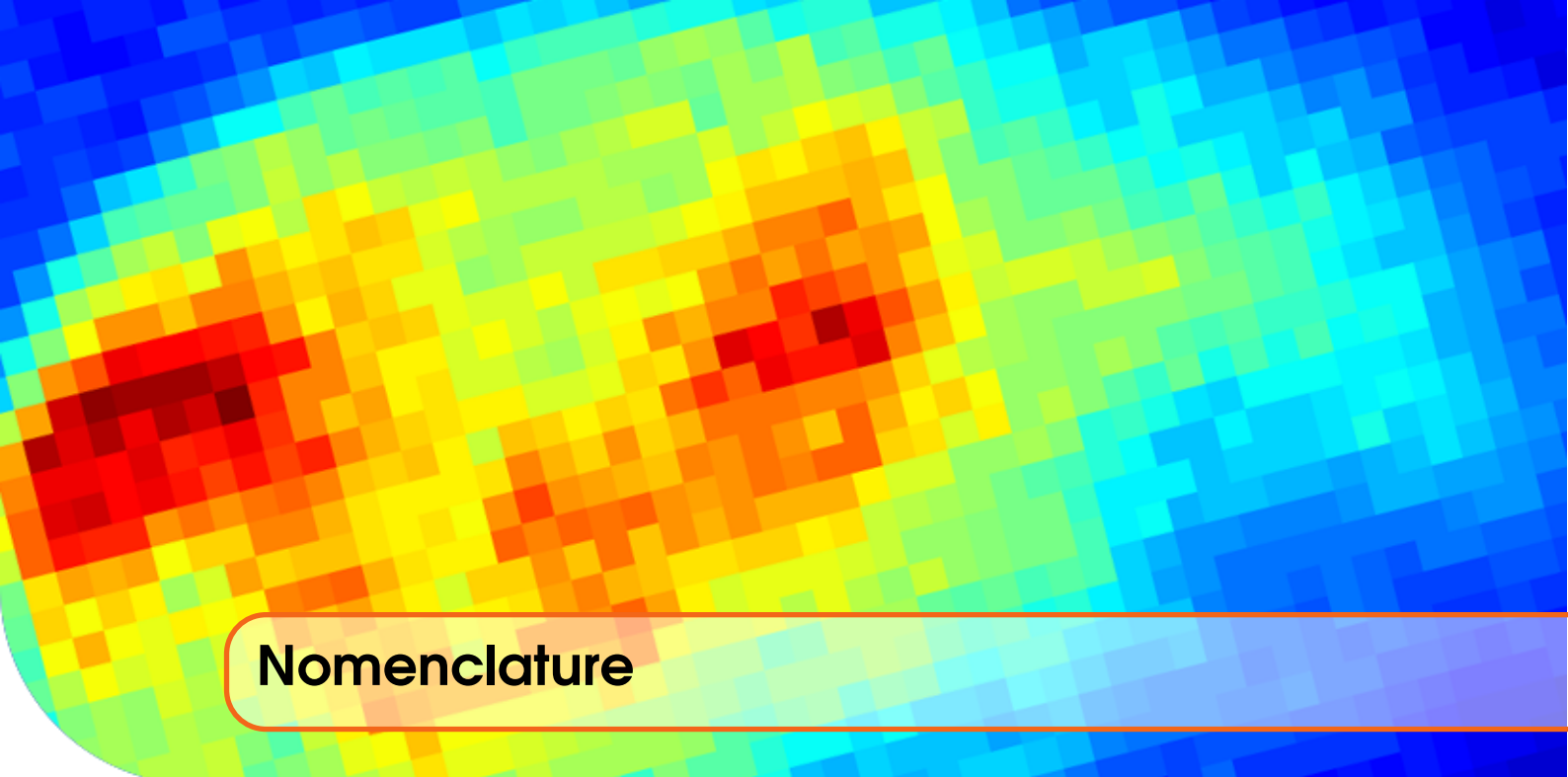
4.1	Cropping one image of a datapack.	70
4.2	Time history speckle pattern of a line from a biospeckle collection of images in a datapack.	71
4.3	Randomly selected points following a Gaussian distribution.	72
4.4	COM of a line in image datapack with the color bar representing the amount of times the jumps occurred.	73
4.5	Absolute probability mass function of a line from a collection of images.	74
4.6	Regular probability mass function of a line from a collection of images.	75
4.7	$\sqrt{IMI}$ and <code>pmfird()</code> function of a line from a collection of images.	77
4.8	Graphic of relations between $AVD1$, $\sqrt{AVD3}$ and <code>pmfad()</code> function.	78
4.9	Graphic of relations between $RVD1$, $\sqrt{RVD3}$ and <code>pmfird()</code> function.	80
4.10	Time history speckle pattern to correlation coefficient.	82
4.11	Spatial-temporal speckle correlation.	82
4.12	Map of activities obtained by the Fujii method on a collection of images of a coffee seed.	83
4.13	Map of activities obtained by the GD method on a collection of images of a coffee seed.	84
4.14	Matrices returned by <code>stdcont()</code> function, with (a) representing the Temporal speckle contrast matrix, named C, (b) representing the Temporal speckle standard deviation matrix, named D, and (c) representing the Temporal speckle mean matrix, named E.	85
4.15	Graphic inertia moment method of a coffee seed.	86
4.16	Graphic AVD method of a coffee seed.	86
4.17	Graphic RVD method of a coffee seed.	87
4.18	Parameterized form of Temporal Difference method.	88
4.19	Temporal speckle kurtosis matrix of a coffee seed	89
4.20	Temporal speckle skewness matrix of a coffee seed.	90
4.21	Motion history image.	90
4.22	Image with dark or saturated areas of a coffee seed, in this case introduced manually.	91
4.23	Matrices showed by the <code>satdark()</code> function.	92
4.24	Windowed spatial speckle contrast image, with $mC = 0.232431$ of a coffee seed. 92	
4.25	Matrices showed by the <code>homogeneity()</code> function.	93
4.26	Scheme of filtering in frequency bands.	94
4.27	Comparison between the AVD graphic activity indicator of the datapack DATA and its homologue evaluated from a datapack formed by the reconstructed signal. 95	
4.28	Comparison between the AVD graphic activity indicator of datapacks in different frequency bands.	96
4.29	Scheme of decomposition in coefficients with information of frequency bands. 97	
4.30	Reconstruction scheme of the coefficients obtained in a DWT	98
4.31	Comparison between the PTD activity indicator of datapack DATA and DATA_. 99	
4.32	Reconstruction scheme of a coefficient with the analysis path $[i, j]$	99
4.33	AVD method in graphic mode of the datapacks {DATA00_, DATA01_, DATA10_, DATA11_}.	101
4.34	Scheme of continuous wavelet transform of a discrete signal.	102
4.35	AVD method in graphic mode of the datapacks DATA0, DATA1 and DATA2. . .	104
4.36	Comparison between the matrices GAVD and GAVDW.	105
6.1	Datapack of a 3D matrix created by grouping NTIMES matrices with NLIN lines and NCOL columns.	113

6.2	Time history speckle pattern construction. Evolution on time of a set of M points in the speckle image. Thus, this set of points will be represented in the lines of a $THSP$ matrix, and the column represents the evolution on time	116
6.3	Time history speckle pattern of a line.	117
6.4	Co-occurrence matrix of a $THSP$ patterns.	120
6.5	Impulse response function $h[n]$	141
6.6	Decomposition scheme according to the used mode.	143
6.7	Quadrature mirror filter in synthesis mode.	144
6.8	Synthesis of a path in a filter bank.	146
6.9	Datapak DATA obtained at the end of a path $SEQ=[0 \ 1 \ 1]$	146
6.10	Frequency response of $D(Z)$	149



List of Tables

4.1	Joint probability mass function of random variables A and B	106
6.1	Inertia moment types.	123
6.2	AVD types.	124
6.3	RVD types.	126
6.4	Types of numerical analysis of the average difference.	128
6.5	Types of correlation coefficients	129
6.6	Use modes of the <i>FIR</i> filter bank.	143



Nomenclature

Datapak - Data package of speckle images

THSP - Time History Speckle Patterns

AD - Average Difference

AVD - Absolute Value of the Differences

BSL - Biospeckle laser

BSLTL - Biospeckle laser tool library

COM - Co-Occurrence Matrix, spatial dependence matrix, GLCM or GLCH

GD - Generalized Difference

GLCH - gray-level co-occurrence histograms

GLCM - gray-level co-occurrence matrices

IM - Inertia Moment

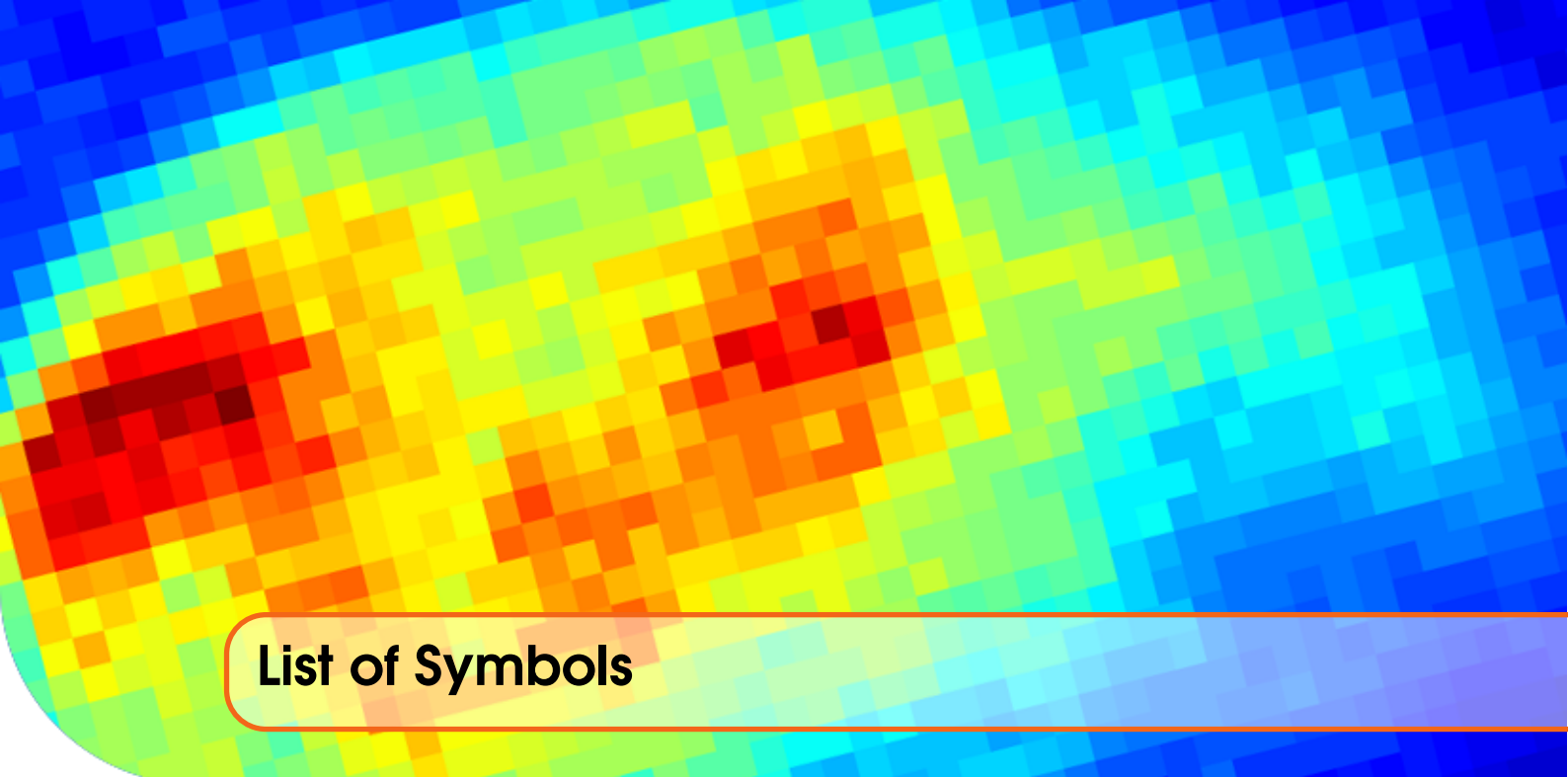
PMF - Probability mass function

PMFAD - Probability mass function of absolute difference

PMFRD - Probability mass function of regular difference

ROI - Region of interest

RVD - Regular Value of the Differences



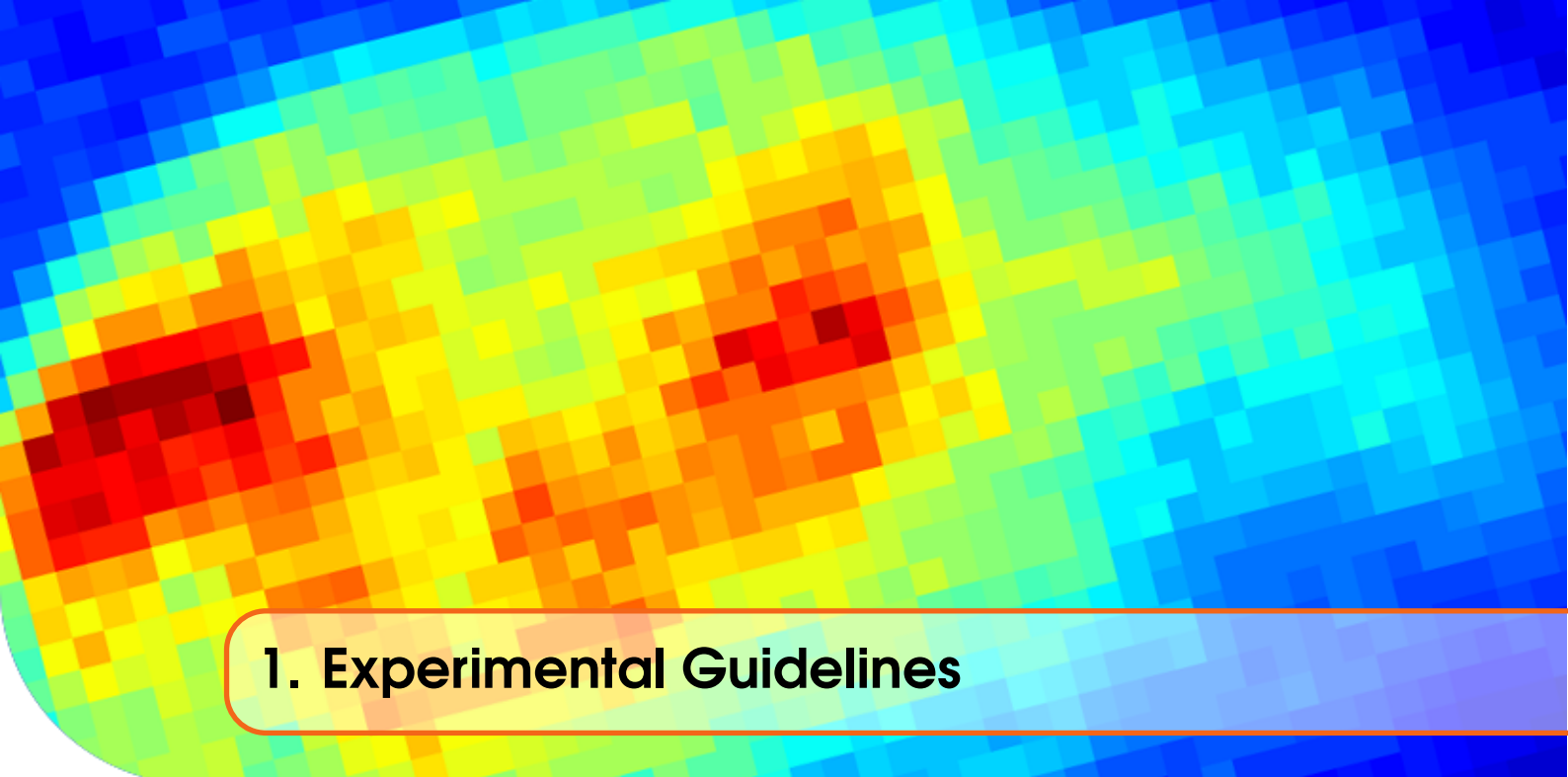
List of Symbols

$NTIMES$	-	Number of samples (images).
$NLIN$	-	Number of lines in the sample.
x	-	Pixel line position. $0 \leq x < NLIN$
$NCOL$	-	Number of columns in the sample.
y	-	Pixel column position. $0 \leq y < NCOL$
$\mathbf{I}_k$	-	Intensity matrix, in the k -th sample.
$I_k(x,y)$	-	Intensity of pixel, in the position (x,y) , in the k -th sample.
$E[a]$	-	Temporal expected value of a .
$\langle a \rangle$	-	Spatial expected value of a .



Part 1: Theory

1	Experimental Guidelines	25
1.1	Introduction	
1.2	Biospeckle Laser System	
1.3	Optical Setup - Hardware	
1.4	On-line Adjustments	
1.5	Quality Tests	
1.6	Methods of Data Analysis - On-line Methods	
1.7	Methods of Data Analysis - Off-line Methods	
1.8	Frequently Asked Questions - FAQ	
1.9	Exercises	



1. Experimental Guidelines

1.1 Introduction

The Biospeckle Laser (*BSL*) is an interferometric phenomenon, which has been observed as a sensitive way to monitor faint changes in the biological samples, and thus it has been adopted as a reliable tool that can be applied in many areas, from medicine to agriculture. The main advantage regarding its use is linked to the simplicity of the apparatus necessary to address the outcomes, associated to the fact that it is a Non-Destructive Test (*NDT*) which is relevant in biological applications.

The multitude of applications demanded the appearing of a range of methods to illuminate, to assemble the images and to provide their analysis. The absence of a standard, and even of a commercial equipment to create some common approaches can be considered the main limit to the biospeckle accessibility as a technology to measure biological activity.

The static appearance of a speckle pattern is expressed in Figure 1.1 and its evolution in time is named dynamic laser speckle, a general term applied to biological and non-biological samples. Therefore, the term *bio* preceding the word speckle represents the applications of the phenomenon in biological material. In Figure 1.1, we have the image of a round seed on a table illuminated by a laser beam and the grains represent the interference pattern in a static way. If you continue to observe the image in time, if the seed is alive and in an adequate moisture, you will see a boiling effect happening on it, i.e. the speckle pattern changing the shape and the grain illumination, whilst on the table the speckle pattern stands still without changes in its grains.

The term biospeckle was at first adopted by Asakura (1988) when presenting a feasible application to blood flow monitoring, a similar application presented by J. Briers (1975) and Fujii et al. (1985). Once the technique showed its viability, many accounts appeared in order to present new approaches to analyze the signal and to provide new applications in other areas of science.

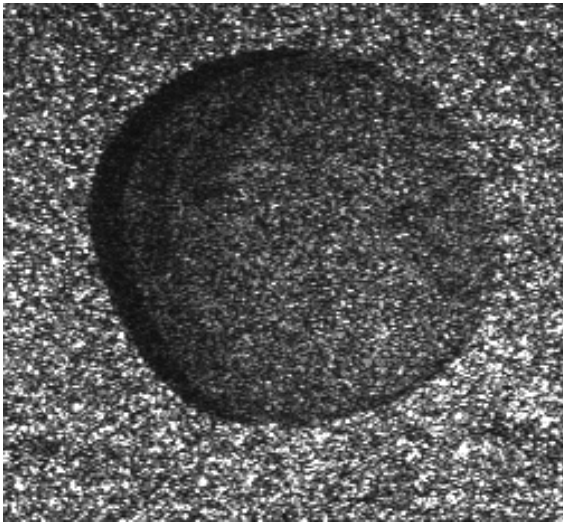


Figure 1.1: Speckle pattern of a seed on a table.

The analysis of the dynamic laser speckle as a source of information about the activity of a biological or even of a non-biological material demands a lot of image processing associated to mathematical and statistical approaches; these analysis were limited by the low capacity of computers and cameras in the earlier applications.

However, the development of digital electronics and laser sources opened new doors regarding the biospeckle laser phenomenon. New cameras, computers and lasers did a revolution in the applications as well as in the field of image and signal analysis. Thus, the complexity of the biospeckle adoption became related to the data processing rather than to the hardware itself.

In this book we would like to offer to new users, or even to experts, a collection of information related to hardware and particularly to software in order to create a guide, integrating all the developments we have done in the field of biospeckle laser applied to biosystems so far.

The software we have developed to analyze the biospeckle laser is presented here to OCTAVE and MATLAB environments, with additional suggestion of free executable procedures. The advantages of the software in OCTAVE or MATLAB versions compared to executable procedures are the facility to change and to tailor the instructions to your particular case, as well as to know exactly what you are doing. In this way, we presented a thorough documentation of the software in order to help you use the best approach to analyze your data. The free software as a collection of procedures is named here *Biospeckle Laser Tool Library (BSLTL)*.

The book is divided into three parts and built as an onion (Figure 1.2). The external part of the onion is the Part I, this part which you are reading now, and it has the aim to introduce you all the main concepts regarding the use of biospeckle laser, from how to build your apparatus to what you need to run the right procedure to get the information you want. The middle layer is Part II, or User Guide of the software, with all the procedures you need to run and analyze your data. Finally, in the core of the onion you will find the Reference Guide with all the details you need to understand the procedures you are planning to use. Thus, it is a complete guide which intends to help you to peel the onion easily, without crying.

The biospeckle theory will not be the main aim here, since it can be obtained on the book *Dynamic laser speckle and applications* by Rabal; Braga (2008). Therefore, the focus is to guide you

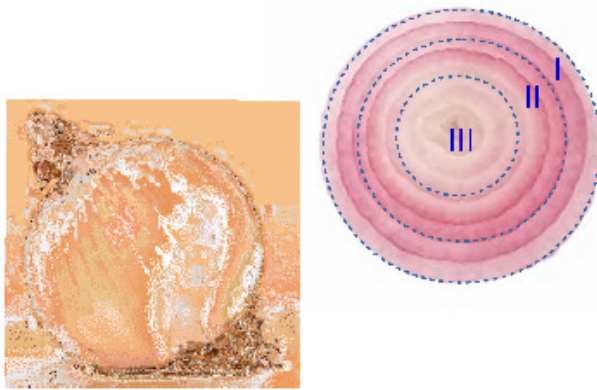


Figure 1.2: Disposition of the book in layers such as an onion. Label I means the first part with an overlook of the theme and comments about the approaches. Label II means the second part with the User Guide with information about the procedures you can use to analyze the data. Label III means the third part, the Reference Guide of the software with the details of all the procedures presented in the two other parts.

through the biospeckle laser application, tutoring you to chose the best experimental configuration (Part I), and the best way to analyze the data (Parts I to III) by means of the *BSLTL*. In addition, we have uploaded reference data in a [repository](#) (BRAGA, 2015), so you will be able to test the procedures presented in this guide using an actual data of a maize and of a coffee seed illuminated by a HeNe laser. The maize seed data has a collection of one hundred frames of a speckle pattern over time, and the first image can be seen in Figure 1.3.



Figure 1.3: First image of a maize seed illuminated by a HeNe laser beam.

1.2 Biospeckle Laser System

The term “biospeckle laser system” has been adopted to designate tools or techniques used to acquire, observe and analyze the dynamic phenomenon of the speckle pattern. The analysis can be classified as numerical and graphical. The data acquired can also be processed in time or frequency domains, and one can see tools being used in on-line and off-line approaches. The summary of

these classifications is plotted in Figure 1.4 where you can see the suggestion of previous steps before the data analysis.

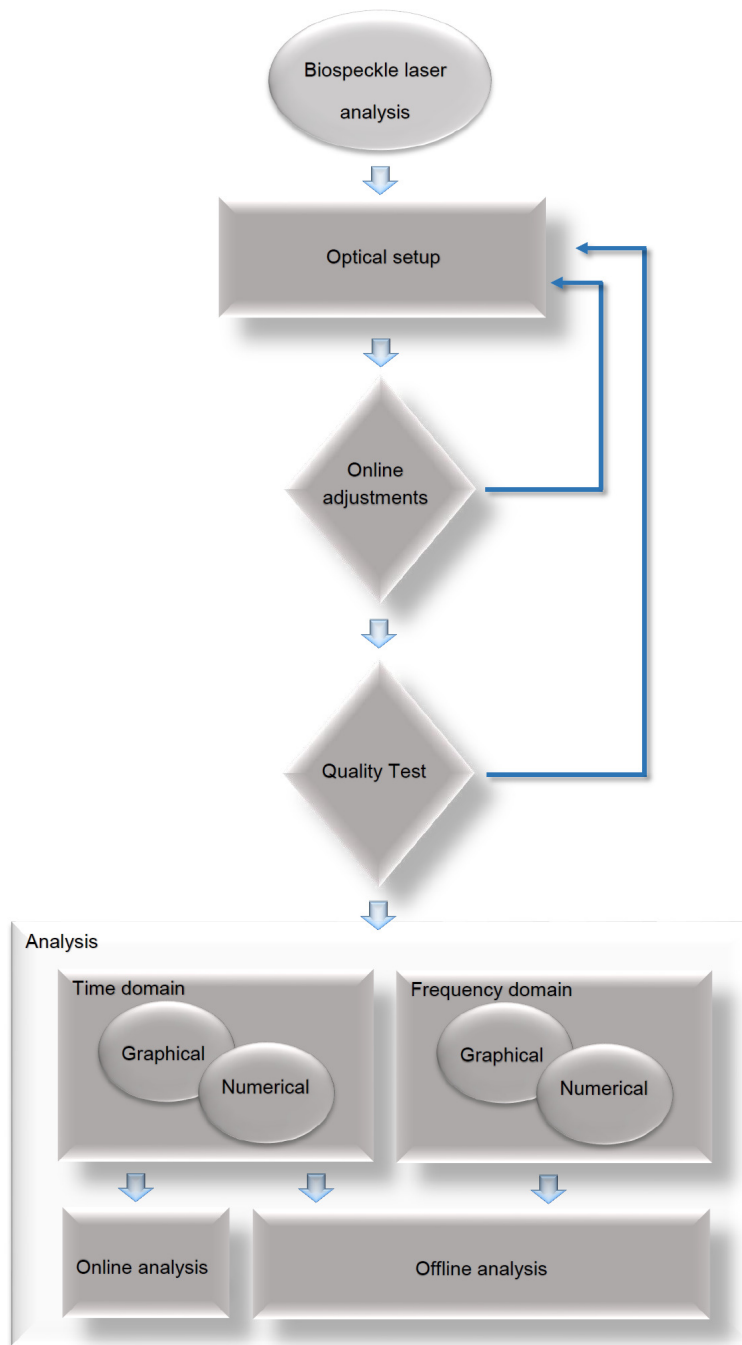


Figure 1.4: Biospeckle laser analysis adopting time and frequency domain approaches, using graphical or numerical outcomes by means of on-line or off-line procedures.

1.3 Optical Setup - Hardware

The hardware adopted to implement the Biospeckle Laser (*BSL*) monitoring can be divided in forward and backward scattering arrangements. The analysis of the biospeckle phenomenon will not vary in accordance with the two arrangements, though it is relevant to evaluate its application and to chose the best arrangement (forward or backward) for monitoring your sample.

The forward-scattering setup presents less sensitivity if compared to the backscattering one (RABAL; BRAGA, 2008, p. 21), and it is limited to applications restricted to the samples that are transparent enough to let the light pass through them. In turn, the analysis of transparent samples can be accomplished by the back-scattering with higher sensitivity than the forward arrangement as well as it is the way we analyze the opaque samples. In Figures 1.5 and 1.6 it is presented the basic design of the backscattering and the forward scattering setups with their optical components.

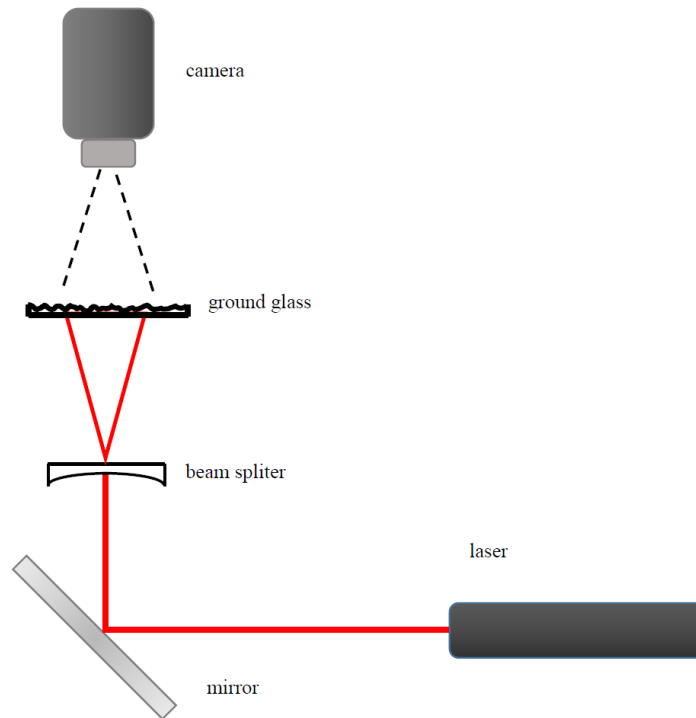


Figure 1.5:
Forward-scattering approach with a basic disposition.

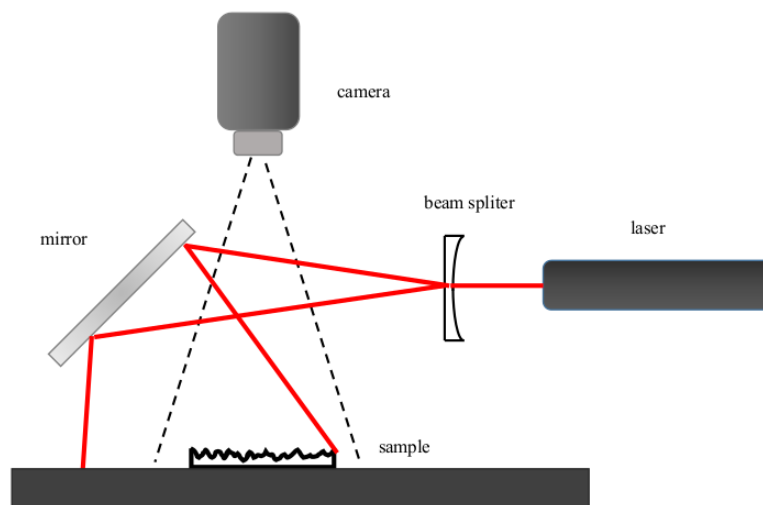


Figure 1.6:
Backscattering approach with a basic disposition.

At this point it is relevant to stress that the laser cannot bring information from the interior of an opaque sample, thereby restricting the information near the surface when the tissue allows some transmission of the light in the layers beneath the surface. In turn, when the sample allows high transmission such as a liquid or a gel, the backscattering approach must consider the adoption of a reflective background (white or light enough) to return the light to the camera.

In the optical setup, the lasers and the cameras are the components with great variations. HeNe lasers are the most used, however one can see great use of solid state lasers in the literature. Despite its usage, we have to be careful with the quality of the laser beam, choosing a solid state laser that has stability of the light intensity over time, as well as spatial homogeneity of the laser beam.

One can question the adoption of solid state lasers to measure the sensitive biospeckle due to the mode hopping phenomenon that can cause some fluctuations of the laser beam over time. However, despite the caution regarding this occurrence, we could not yet identify any compromise our results with the adoption of solid state lasers.

Regarding the cameras, one can see analogue devices first used in biospeckle applications, followed by the digital cameras which represented a great jump in the researches in the field, though one can see every day a new revolution represented by new features such as speed, resolution and communication with computers. One example is the *USB* interface, which can be considered the best interface between camera and computers, so far. Among the great variety of cameras in the market, the mini-microscopes with built-in macro lenses are a good options, since the speckle pattern demands magnification to be observed properly. The main drawback of the mini-microscopes is the absence of built-in iris control which is interesting for adjusting the size of the speckle grains and filtering the light.

The *USB* protocol is easy to access through many programs, such as the ones we adopt here in this guide. Thus, by means of known and stable functions we can access the data of a camera from the software and control it.

We shall move to another part of the optical arrangement that causes many queries, the beam expander. The expansion of the beam is a basic step in the biospeckle analysis, however you should have in mind the quality of the light in the surface avoiding, for example, the under-exposition. The best beam expander one can use is the microscope's objective, since it does not distort the beam like the common concave lenses.

The quality of the speckle pattern is the main query after the experimental configuration is set, and it is a key point concerning the biospeckle application. Thus, the challenge is to tailor the quality of the speckle pattern to each illuminated sample. The level of quality can be summarized by the quality of the grains, i.e. the size of the grain, that needs to be larger than a pixel, and its contrast, that is linked to the compatibility speed between the camera and the phenomenon as well as linked to the level of illumination.

This guide will present in the next sections the Quality Test Protocol to help the users to tailor their experimental configuration. Additional elements adopted to help the adjust of the level of illumination are the neutral filter that can be placed before the beam expander, and the ground glass usually adopted in the forward-scattering.

1.4 On-line Adjustments

We firmly suggest you to test your experimental configurations using on-line adjustments in order to check some basic points such as the ideal position of the region you are interested in (we will address it many times by region of interest or *ROI*), or even the quality of your illumination. In our laboratories we adopt an on-line tool we have developed, named **Speckle Tool** (BRAGA et al., 2014), that runs the Motion History Image (*MHI*) in an executable format. Please, read about it in Section 1.6.

1.5 Quality Tests

Quality Test Protocols were proposed (CARDOSO; BRAGA; RABAL, 2012; MOREIRA; CARDOSO; BRAGA, 2014) in order to guide the users to achieve the best data to their applications. The procedures help the user to guarantee the quality of the speckle pattern by means of three perspectives:

- Saturation test, concerning the over and under exposure of the light in the sample;
- Contrast test, with the evaluation of the compatibility between the camera and the phenomenon speed;
- Homogeneity test, with the creation of a map showing the areas where the activity can be homogeneous.

The three test were implemented by us in OCTAVE and MATLAB codes, and they are presented in details in the next chapters. One executable version of the quality test with reduced space to intervention of the user can be found in Braga; Silva (2014).

1.5.1 Saturation and Under-exposition Tests

The Saturation test is based on the fact that, to the point of view of the camera, the light of the laser on the sample must be within the range of 8 bits, i.e. it must be between 0 to 255, levels of gray. If you have over-exposition of the sample with the laser, some areas of interest will saturate, which means that the values of gray are mostly close or over to 255. The inverse will be the same, which means that it is yielding the under-exposition of the sample with dark areas, mostly close to zero. If the under or over-exposition occurs in a Region Of Interest (*ROI*) this will compromise your results. Therefore, this test only evaluates how close you are to the limits. In Figure 1.7, it is possible to see the speckle pattern in a maize seed, with two different *ROI*'s.

The analysis of the histogram of both *ROI*'s gives us the idea of the Saturation test outcome. In Figure 1.8, the *ROI* in the area tagged by *c* presents a histogram with gray level occurrences within the range 0 to 255, thus with a good condition regarding the exposition of the light.

However, in Figure 1.9, the *ROI* in the area tagged by *d* presents a clear shift of the gray level occurrences to the upper limit, meaning that the analysis of the data will be compromised because of the saturation.

The calculus of the saturation test, of the first image of a maize seed (BRAGA, 2015), can be seen in Figure 1.10, with the gray area delimited where the over-exposition occurred. The picture is divided into windows of 30 by 30 pixels. Pixel values less than 60 are considered dark, whereas

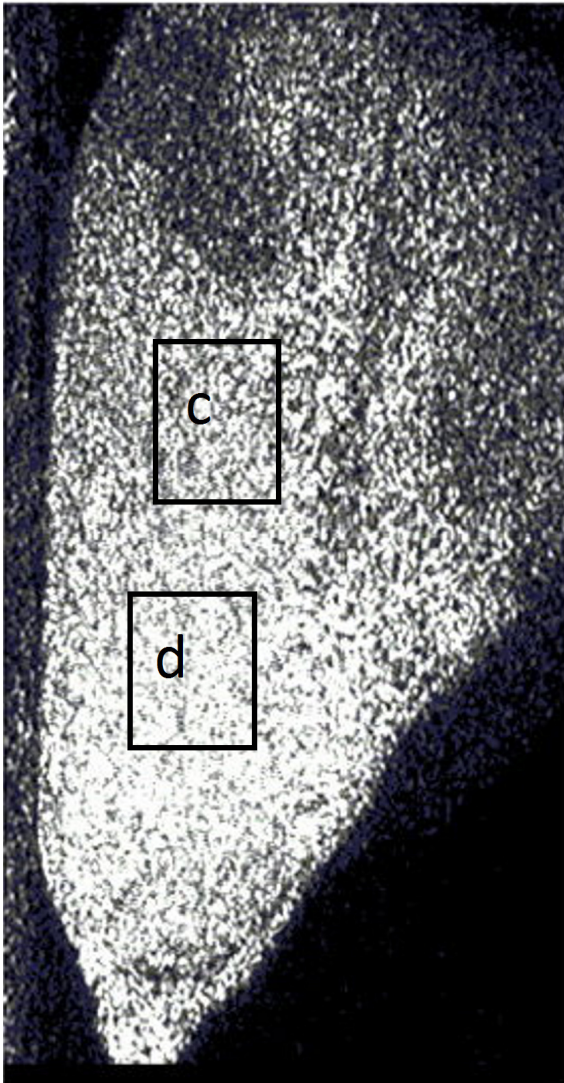


Figure 1.7: Image of a maize seed with two different regions of interest (*c* and *d*) on the seed. Reprinted from *Publication Optics and Lasers in Engineering*, 61, Junio Moreira, R. R. Cardoso, R. A. Braga, *Quality test protocol to dynamic laser speckle analysis*, Pages No. 8-13, Copyright (2014), with permission from Elsevier.

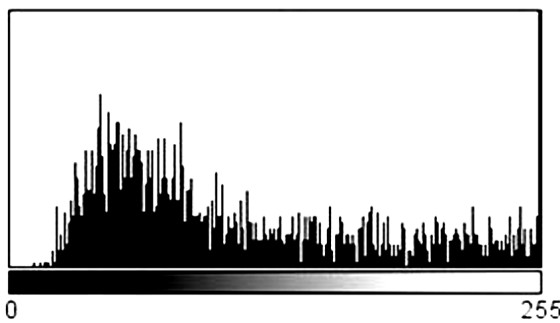


Figure 1.8: Histogram of the region of interest tagged by the letter *c* where the distribution of the levels of gray does not present over-exposition of the laser on the seed. Reprinted from *Publication Optics and Lasers in Engineering*, 61, Junio Moreira, R. R. Cardoso, R. A. Braga, *Quality test protocol to dynamic laser speckle analysis*, Pages No. 8-13, Copyright (2014), with permission from Elsevier.

values greater than 250 are considered saturated. The saturation and under-exposition tests finally return a picture with saturated regions filled with intensity values equal to 255 (or red), and dark regions filled with zero (or blue), as shown in Figure 1.11.

In this case, the area with saturation is exactly in a very important area of the seed, thus compromising the analysis.

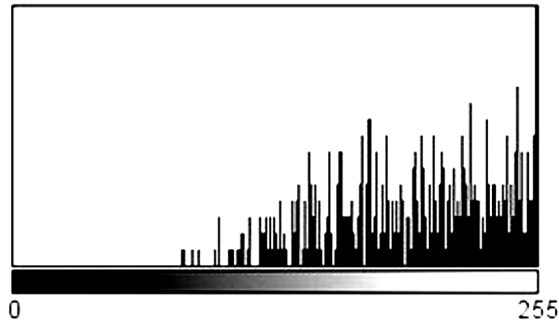


Figure 1.9: Histogram of the region of interest tagged by the letter *d* where the distribution of the levels of gray presents over-exposition of the laser on the seed. Reprinted from *Publication Optics and Lasers in Engineering*, 61, Junio Moreira, R. R. Cardoso, R. A. Braga, *Quality test protocol to dynamic laser speckle analysis*, Pages No. 8-13, Copyright (2014), with permission from Elsevier.

0	0.11	0	0.55	0.55	1.11	0	0
1.44	0	0	2.11	6.77	2.44	1.88	0
0.88	2.22	1.22	0.55	6.55	6.44	2.11	0.44
0	6.88	6.22	1.33	7	9.55	7.22	1.44
0.44	12.44	16.11	16.33	23.44	11	9.22	0.66
4.44	11.44	23.11	27.88	24.55	12.77	10.77	1.44
3.88	15.77	21.33	32.11	24.33	16.33	7.44	0.66
6.22	27.33	30.88	35.88	32.55	15.77	13.22	2.77
5.33	35.44	44.11	46.77	35	31.88	25.22	5.88
11.55	44.66	56.44	55.55	55.22	41.55	26.22	4
13.11	57.11	59.88	62.33	60.66	43.44	35.11	0
19	65.22	69	67.77	60	18.44	0	0
15.33	60.88	60.55	52.22	40.44	3.11	0	0
8.77	46.77	43.55	29.55	7.22	0	0	0
0	41.11	23.88	13.55	0.22	0	0	0
0.11	24.55	35.11	0.88	0	0	0	0

Figure 1.10: Saturation test map with values of gray levels presenting the over-exposition of the laser on the seed in the region of interest in gray. The numbers mean the pixels over 50% over the upper limit. Reprinted from *Publication Optics and Lasers in Engineering*, 61, Junio Moreira, R. R. Cardoso, R. A. Braga, *Quality test protocol to dynamic laser speckle analysis*, Pages No. 8-13, Copyright (2014), with permission from Elsevier.

1.5.2 Contrast Test

The second test of quality is the contrast (specifically, the spatial image contrast). It defines the compatibility between the velocity of the camera and the phenomenon observed, or the focus in the picture. The way to carry out this test is based on the contrast test proposed by J. Briers (1975), where the contrast of a *ROI* in one speckle pattern can be expressed by the Equation (1.1),

$$C = \frac{\sigma}{\mu}. \quad (1.1)$$

So that $\sigma = \sqrt{\langle (I - \mu)^2 \rangle}$ and $\mu = \langle I \rangle$ which represent the spatial standard deviation and the spatial mean value respectively. Both values are related to the intensities I inside of a *ROI*. Thus, C defines the degree of difference or opposition between intensities. A high contrast C is

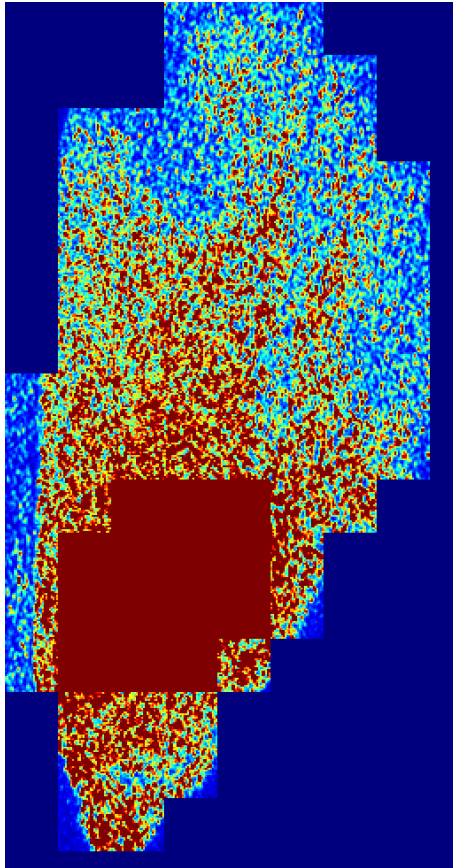


Figure 1.11: Saturation and under-exposition tests presenting the over-exposition of the laser on the seed.

characterized by a *ROI* with a high definition of the grains, whereas a low contrast is characterized by a *ROI* with blurring.

In Figure 1.12, one speckle pattern of a paint drying over time presents the aim of this test. Just 5 minutes after painting, it is possible to observe how the speckle pattern is blurred as a result of the high speed of the boiling, if compared with the speed of the camera. Therefore, after 15 minutes one can see the speckle pattern with well-defined grains, and then *the ideal condition to carry on the biospeckle analysis*. If you decide to acquire data of this phenomenon at 5 minutes your results will be compromised by the aliasing.

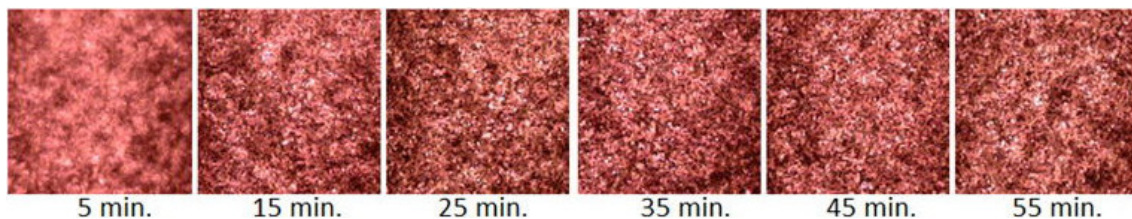


Figure 1.12: Speckle patterns in different instants presenting the blurring at the first instant. Reprinted from Publication *Optics and Lasers in Engineering*, 61, Junio Moreira, R. R. Cardoso, R. A. Braga, *Quality test protocol to dynamic laser speckle analysis*, Pages No. 8-13, Copyright (2014), with permission from Elsevier.

The contrast test applied to the first image of the maize biospeckle data (BRAGA, 2015), Figure 1.13, presents an area of low contrast (blue zone) in the middle of the embryo, at the same place as

the saturation area, meaning the interference of the over-exposure of light in the contrast of speckle grains.

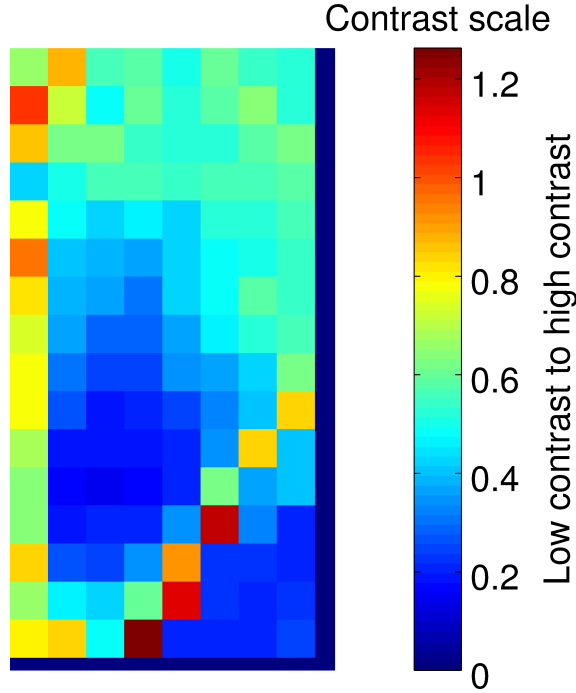


Figure 1.13: Contrast outcome test in a maize seed on analysis windows of 30 by 30 pixels.

1.5.3 Homogeneity Test

The third test is the homogeneity, which calculates the spatial homogeneity of any indicator that measure the activity of a biospeckle phenomenon. This test is relevant when one needs to carry on a numerical analysis avoiding transition areas within the sample. Any indicator known in the literature can be used to this purpose; for example can be used the Inertia Moment (ARIZAGA; TRIVI; RABAL, 1999) or the *AVD* (BRAGA et al., 2011).

The homogeneity test divides the images of $nN \times mM$ pixels, in the analysis windows $W(i, j)$ of $n \times m$ pixels, for all $1 \leq i \leq N$ and $1 \leq j \leq M$, applies the chosen indicator on these data, getting an activity indicator value $A(i, j)$ for the window $W(i, j)$. An example of activity indicators on analysis windows, the Figure 1.14 shows the result of applying the Inertia Moment indicator on the maize biospeckle data (BRAGA, 2015). The homogeneity value H (Equation 1.2), of window $W(i_0, j_0)$, can be calculated as the complement of spatial contrast C_a (Equation 1.3). So that

$$H = 100 \left(1 - \frac{C_a}{C_{max}} \right) \quad (1.2)$$

and

$$C_a = \sigma_a / \mu_a, \quad (1.3)$$

where $\mu_a = \langle a \rangle$ and $\sigma_a = \sqrt{\langle (a - \mu_a)^2 \rangle}$ are the mean value and standard deviation the set of values $a = \{A(i_0 - 1, j_0), A(i_0, j_0 - 1), A(i_0, j_0), A(i_0, j_0 + 1), A(i_0 + 1, j_0)\}$ respectively. The

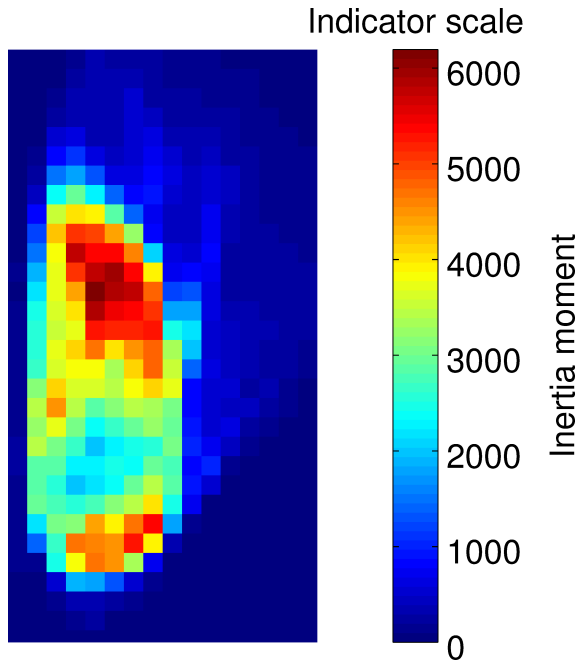


Figure 1.14: Result of applying the Inertia Moment indicator on analysis windows of 16 by 16 pixels.

value C_{max} represents the maximum value of spatial contrast in all windows $W(i, j)$ in the activity indicator.

The Figure 1.15 represents the outcome of applying the homogeneity test on the maize biospeckle data (BRAGA, 2015) for mapping the areas that have the same activity, in order to choose the best *ROI* to conduct numerical analysis. Therefore, you can elect one *ROI* with high homogeneity avoiding areas that presents transitions. This test is useful when the sample cannot be considered as homogeneous such as in the case of the maize, with many distinct areas in the biospeckle point of view.

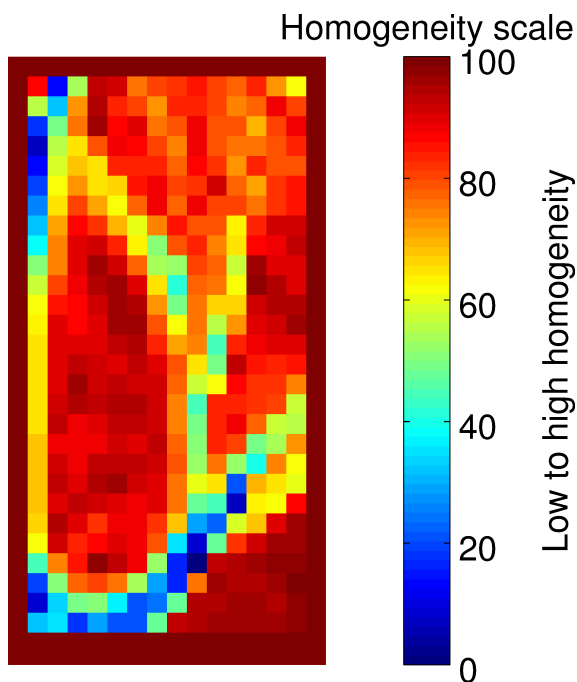


Figure 1.15: Homogeneity outcome test in a maize seed.

1.6 Methods of Data Analysis - On-line Methods

The on-line procedures associated to the biospeckle laser have their first reference linked to the *LASCA* software. The acronym *LASCA* means Laser Speckle Contrast Analysis and it was proposed by J. D. Briers; Webster (1996).

The on-line analysis was the only way to overcome the kinking (micro-tremors) of human beings, or alive animals, during the biospeckle analysis of blood micro-circulation, also known as perfusion.

In this case, just one image was acquired and then the blur was caused by the activity in the speckle pattern measured by means of the contrast, see Equation (1.1). The drawback of this approach is the low resolution of the final image representing the map of activity, and, in addition, it presents the challenge of tailoring the shot features of the camera related to the phenomenon observed.

As an alternative to that, we have developed a procedure based on the Motion History Image (*MHI*) approach (GODINHO et al., 2012; DAVIS, 2001) in order to maintain the same resolution of the prime images, as well as creating a collection of adjustments to use more than one image and avoiding the critical adjustment of the camera's shot. The *MHI* also represents an useful way to evaluate the phenomenon on-line (On-line adjustments in Figure 1.4) before its assembling and submission to accurate analysis, usually by off-line procedures.

1.6.1 MHI

We present in this book the OCTAVE and MATLAB procedures to run the *MHI*; however, it also can be used by means of an executable free software Speckle Tool at (BRAGA et al., 2014).

The basic idea to compute *MHI* will be presented in the next steps, and we will use a collection of pictures of a student lifting his arm to illustrate it. In the case of Figure 1.16, the arm is in its final position. Notice that the student has on his chest a paper with the identification of the university and of our center.



Figure 1.16: Image of a person with his arm lifted after moving it. *Reprinted from Optics and Lasers in Engineering*, 50, 3, R. P. Godinho, M. M. Silva, J. R. Nozela, R. A. Braga, *On-line biospeckle assessment without loss of definition and resolution by motion history image*, Pages 366-372, Copyright (2012), with permission from Elsevier.

The first step of *MHI* procedure is the subtraction of two subsequent 8-bit images, I_l and I_{l-1} , getting a matrix S_l as

$$S_l = I_l - I_{l-1}. \quad (1.4)$$

The result of subtraction can be seen in Figure 1.17, where S_l is the silhouette of two consecutive images.



Figure 1.17: Subtraction of two consecutive images of a person moving his arm. *Reprinted from Optics and Lasers in Engineering, 50, 3, R. P. Godinho, M. M. Silva, J. R. Nozela, R. A. Braga, On-line biospeckle assessment without loss of definition and resolution by motion history image, Pages 366-372, Copyright (2012), with permission from Elsevier.*

The image S_l resulted from the subtraction must be transformed in binary (Equation 1.5), and therefore a threshold must be proceeded for each pixel (i, j) , so that

$$T_l(i, j) = \begin{cases} 1 & \text{if } |S_l(i, j)| > Z \\ 0 & \text{if } |S_l(i, j)| \leq Z \end{cases} \quad (1.5)$$

and the result of this action on Figure 1.17 can be seen in Figure 1.18. Where the matrix T_l is the thresholded image of S_l using a Z limit.

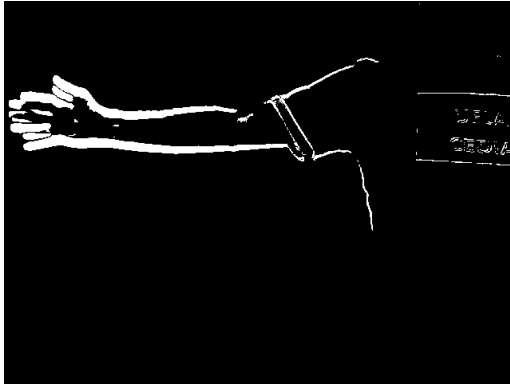


Figure 1.18: Threshold of the subtraction of two consecutive images of a person moving his arm. *Reprinted from Optics and Lasers in Engineering, 50, 3, R. P. Godinho, M. M. Silva, J. R. Nozela, R. A. Braga, On-line biospeckle assessment without loss of definition and resolution by motion history image, Pages 366-372, Copyright (2012), with permission from Elsevier.*

The history of the motion of the arm is therefore recorded by the *MHI* procedure, in the instant l , through the next equations,

$$MHI_l = 255 \sum_{k=0}^{N-1} T_{l-k} h_k, \quad (1.6)$$

$$h_k = \frac{N-k}{M}, \quad (1.7)$$

where the h_k value is the weighting that is based on the image's age and $M = N(N+1)/2$ is equal to a summation of N first natural numbers, so that the summation of all h_k is ever 1.0. Thus, the

last motions receive the highest gray level, then it is possible to follow the movement while the student lifts his arm.

In Figure 1.19 it is possible to see the *MHI* of an arm being lifted from its first position on the bottom to the top position. The light values of gray means movement, where the higher values address the latest changes in the positioning. See also the sensitivity of the technique that offers the ability to read the acronyms in the student's chest.

One needs to observe that the *MHI* procedure can be adjusted by means of at least three factors, the number of images that will be part of the buffer, the limit of the threshold and the weighting of the subtractions. The adjustments of those factors will let you bias your on-line observation with respect to the phenomenon speed and as well as tailor the sensitivity of the method.

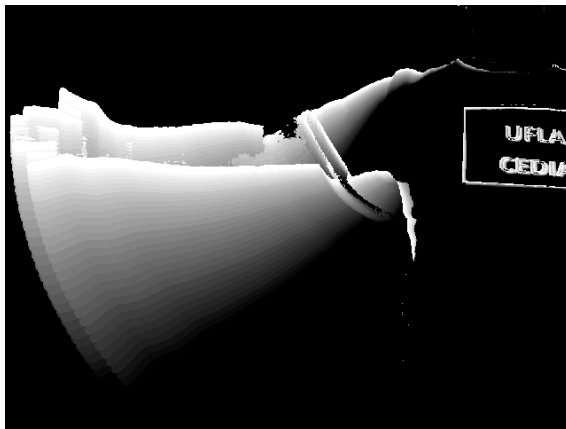


Figure 1.19: Motion history of consecutive images presenting the evolution of the movement of an arm from bottom to top, thus presenting the last movement as clear gray. Reprinted from *Optics and Lasers in Engineering*, 50, 3, R. P. Godinho, M. M. Silva, J. R. Nozela, R. A. Braga, *On-line biospeckle assessment without loss of definition and resolution by motion history image*, Pages 366-372, Copyright (2012), with permission from Elsevier.

Finally, the gray level can be enhanced by a pseudo-color image as can be seen in Figure 1.20.

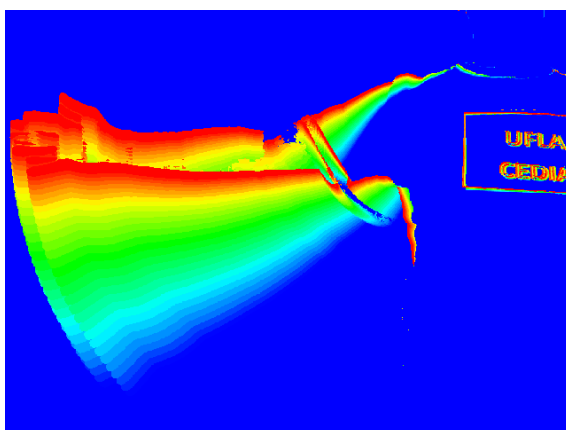


Figure 1.20: Motion history of consecutive images presenting the evolution of the movement of an arm from bottom to top, thus presenting the last movement as red. Reprinted from *Optics and Lasers in Engineering*, 50, 3, R. P. Godinho, M. M. Silva, J. R. Nozela, R. A. Braga, *On-line biospeckle assessment without loss of definition and resolution by motion history image*, Pages 366-372, Copyright (2012), with permission from Elsevier.

The same idea is adopted in the speckle pattern images and you can see one example in Figure 1.21 when a maize seed, illuminated by a laser, is observed on-line by the *MHI*. In our laboratories, we adjust the experimental configuration using the *MHI* on-line before assembling the first collection of images to proceed the Quality Test procedure (see Figure 1.4).

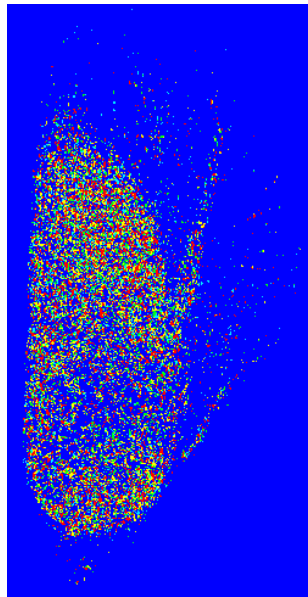


Figure 1.21: Motion history of consecutive images of the maize seed.

1.7 Methods of Data Analysis - Off-line Methods

The off-line analyses are usually a relevant way to process the biospeckle data, and as you could see in Figure 1.4 they can be done by means of graphical or numerical approaches. The off-line analysis also allows the adoption of frequency domain approach, since in on-line approaches the low number of frames are prohibitive in this domain. The information about the methods will be presented here (Part1) only to let you know the main concepts of each one, since the details can be seen in Parts 2 and 3 of this guide book.

1.7.1 Graphic Methods

An illustration of graphical analysis outcome of biospeckle data can be seen in Figure 1.22, where the original photo of a coffee seedling had its root illuminated by a laser and the images were analyzed using a graphical approach. Usually, graphical outcomes are presented in gray levels (8 bits), or alternatively in pseudo-color, with the activity represented from blue, low activity, to red, high activity. If it were gray, the outcome would be from black, low activity, to white, high activity. The main outcome of the graphical analysis is the map of activity, and it is adopted in heterogeneous samples where one desires to identify distinct areas. It is possible to get a numerical figure from these graphs finding the mean values in a Region Of Interest (*ROI*). The extraction of the numerical values from the graphical outcomes is not the best approach when one need comparisons since they are restrict to the levels of gray (0 to 255).

One can list some methods to compute the graphical outcomes, but the most used in the literature are three:

- Average Difference (*AD*) also called Fujii method (FUJII et al., 1987; FUJII; ASAKURA, 1975);
- Generalized Differences (*GD*) (ARIZAGA et al., 2002);
- Temporal speckle Standard Deviation (*SD*) (BLOTTA et al., 2011).

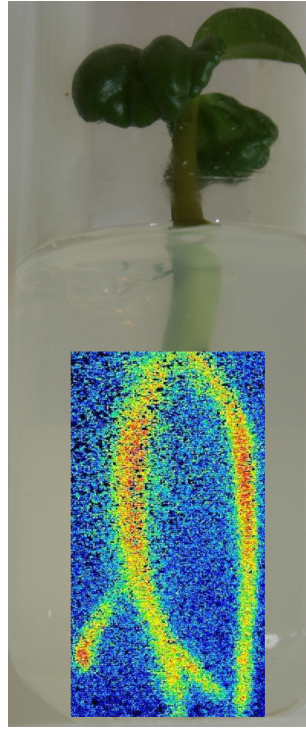


Figure 1.22: Mounted image of a coffee seedling with roots analyzed by the biospeckle laser presenting the map of activity. In the map, the pseudo-colors mean the level of activity from blue, low activity, to red, high activity.

The Average Difference - the Fujii Method

Fujii method (FUJII et al., 1987; FUJII; ASAKURA, 1975), originally called Average Difference (*AD*), is the first method known and it provides an outcome with a relative value, since the presence of the denominator in Equation (1.8) makes each difference relative to the intensities observed. Thus, $\mathbf{I}_k$ represents an image (intensity matrix) taken in the instant k , which is presented in Equation (1.8)

$$AD = \sum_k \frac{|\mathbf{I}_k - \mathbf{I}_{k-1}|}{\mathbf{I}_k + \mathbf{I}_{k-1}}. \quad (1.8)$$

One consequence of this is the ability of the Fujii method to enhance the differences in areas of the image that are dark, thereby the final image is clearer than the other graphical methods, and it is possible to see in Figure 1.23. In the figure, the Fujii method was applied in the maize seed and the smoothness in the final image can be observed.

The Generalized Difference Method

Generalized Difference (*GD*) method (ARIZAGA et al., 2002) in turn, as its name says, provides the difference of each image with respect to all the others. The absence of the normalization as occurred in Fujii method creates a sharp outcome with the areas of activity well delimited, see Figure 1.24 and compare the result with the Figure 1.23. One issue concerning the *GD* method is the high computer time-consuming regarding the generalized differences. In Equation (1.9) it is possible to see the method.

$$GD = \sum_k \sum_l |\mathbf{I}_k - \mathbf{I}_{k-l}| \quad (1.9)$$

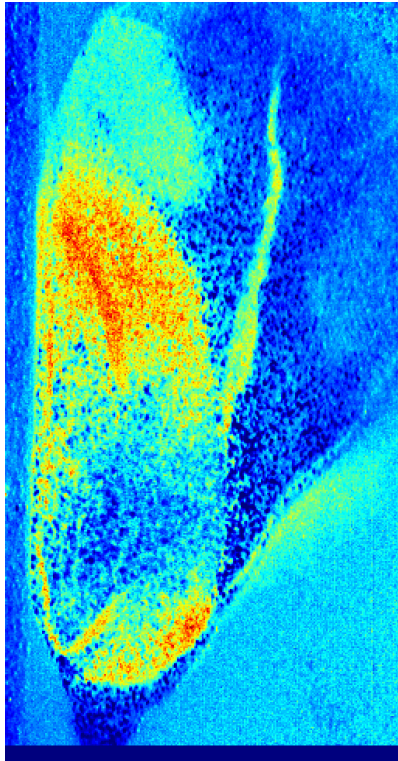


Figure 1.23: Map of activities on a maize seed using the Fujii method.

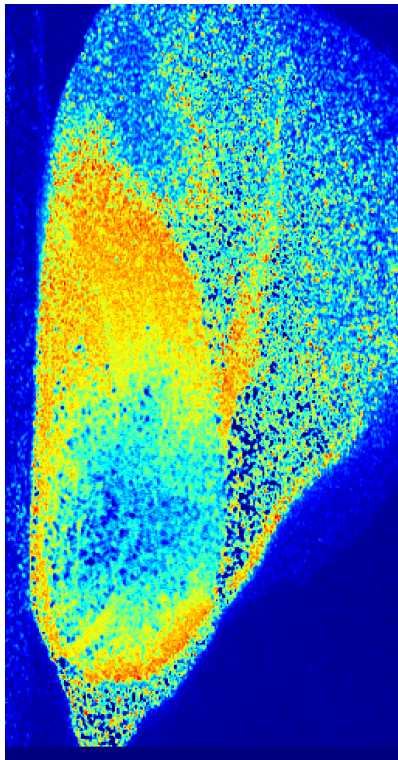


Figure 1.24: Map of activities in a maize seed using the DG method.

Where the image $\mathbf{I}_k$ is compared to all the images $\mathbf{I}_{k-l}$ of the collection.

The Temporal Speckle Standard Deviation

The *SD* method (BLOTTA et al., 2011) provides quite the same outcome of the *GD* method, though with lower computer time-consuming, thereby we normally adopt the *SD* method instead of *GD* in

our laboratories, Equation (1.10), where the $\mathbf{I}_k$ matrix represents the image taken in the instant k , $E[\mathbf{I}_k]$ means the temporal mean value of $\mathbf{I}_k$, for all k , and N means the number of images analyzed,

$$SD = \sqrt{\frac{1}{N} \sum_k |\mathbf{I}_k - E[\mathbf{I}_k]|^2}. \quad (1.10)$$

1.7.2 Numerical Methods

The numerical methods are adopted when one desires to quantify the level of changes in a speckle pattern in time, which is also known as level of activity. The main assumption to use a numerical approach is that the area analyzed is homogeneous. It is possible to get the level of changes from the graphical outcomes, however there are some particular approaches to provide the values related to the level of changes in a speckle pattern. The numerical and graphical methods adopt the same statistical and mathematical approaches, and thus they can be associated (BRAGA et al., 2012). The importance of the Quality Test plays a key point here, particularly the homogeneity.

Time History Speckle Pattern

Oulamara; Tribillon; Duvernoy (1989a) constructed what we know as Temporal History of the Speckle Pattern (*THSP*), and opened the way to analyze the phenomenon by means of numerical outcomes, in a *ROI* within a supposed homogeneous area, though monitoring one line, of M pixels, in time through N samples.

Therefore, the construction of the *THSP* traditionally adopts one line or column in time; however, when the homogeneity level of the *ROI* is not high, the adoption of a line or a column can compromise the results. An alternative approach that we are using with success is the construction of a *THSP* by means of a collection of M points randomly chosen in the first image and with the same points monitored in time. In Figure 1.25, it is possible to see a *THSP* of an active sample and, in Figure 1.26, the *THSP* of a non-active sample, thus the *THSP* can be a good graphical information of the level of activity. Compare the images and observe the lines better defined in Figure 1.26 than in Figure 1.25. The visual comparisons work well with different activities, but it is not the case when we have levels much closer than the ones of the example, thereby the need of numerical outcomes.

The *THSP* images are the base of some methods to obtain the numerical outcomes such as the autocorrelation (XU; JOENATHAN; KHORANA, 1995), the Inertia Moment (*IM*) (ARIZAGA; TRIVI; RABAL, 1999), and the Absolute Value of the Differences (*AVD*) (BRAGA et al., 2011), which is a variation of the inertia moment. The contrast (J. BRIERS, 1975) that is linked to the on-line approaches is another way to get numerical values, and as well as the autocorrelation of recorded images (without using the *THSP*) (KURENDA; ADAMIYAK; ZDUNEK, 2012). Other variations linked to the *IM* and *AVD* tried to enhance their robustness, particularly regarding the normalization (CARDOSO; BRAGA, 2014). In this guide you will find the way to understand how the *IM* and the *AVD* are implemented, including their variations.

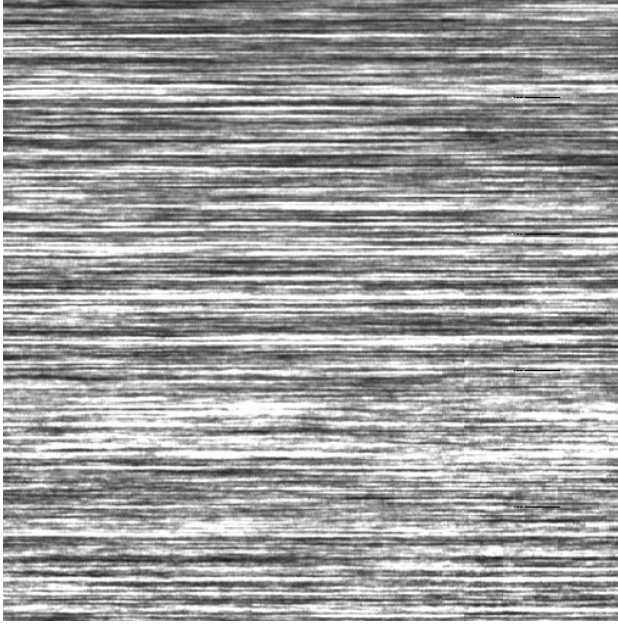


Figure 1.25: Time History Speckle Pattern (*THSP*) formed from a collection of images of an animal sperm with high activity. The center line of each image was picked up, rotated and allocated side by side forming the *THSP* with the columns representing the history of the pixels along the time.

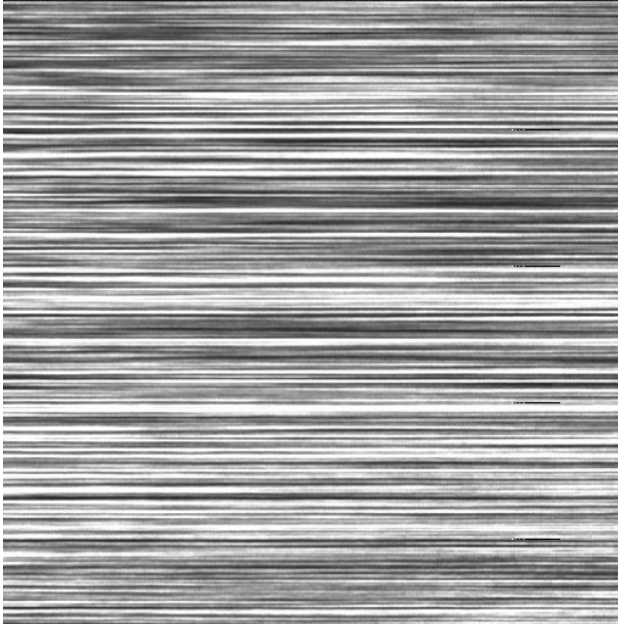


Figure 1.26: Time History Speckle Pattern (*THSP*) formed from a collection of images of an animal sperm with low activity. The center line of each image was picked up, rotated and allocated side by side forming the *THSP* with the columns representing the history of the pixels along the time.

Co-occurrence Matrix

The Co-occurrence Matrix (*COM*)(ARIZAGA; TRIVI; RABAL, 1999) is an intermediary matrix that we use to evaluate the dispersion of consecutive pixels in a *THSP* of M points monitoring through N samples, as shows the Equation (1.11),

$$COM(i, j) = \sum_{m=1}^M \sum_{n=1}^{N-1} \begin{cases} 1, & \text{if } THSP(m, n) = i \text{ and } THSP(m, n+1) = j \\ 0, & \text{otherwise} \end{cases} \quad (1.11)$$

where the *COM* matrix represents a transition histogram of intensity. Thus, $COM(i, j)$ has the number of times a transition of an intensity level i to j happens.

In Figure 1.27, one can see the co-occurrence matrix of the *THSP* presented by Figure 1.25, that is, of a animal sperm with high activity. It is possible to observe that the points are spread along the main diagonal, but with higher dispersion than the *COM* from a *THSP* representing low activity (Figure 1.28), such as the *THSP* of Figure 1.26. The *COM*, such as the *THSP*, are graphical options to evaluate the level of activity of a sample; however, the numerical outcome like inertia moment or *AVD* is relevant to statistical analysis and comparisons between close stages.

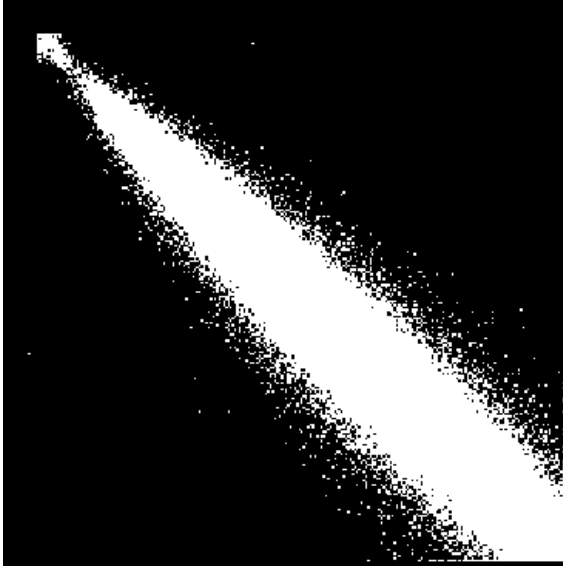


Figure 1.27: Co-occurrence matrix of a *THSP* formed from a collection of images of an animal sperm with high activity.

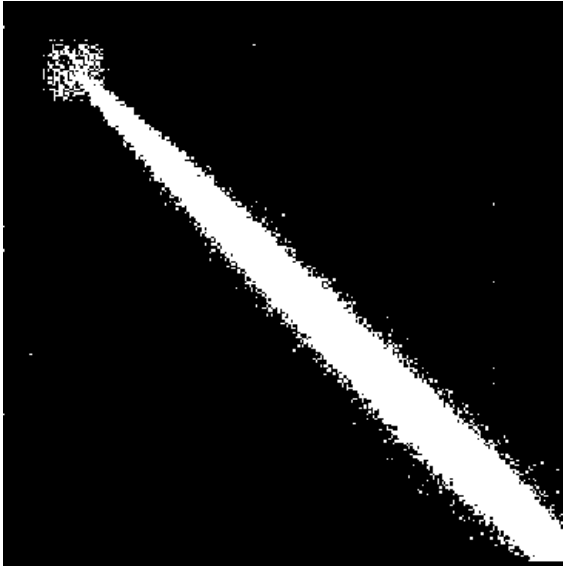


Figure 1.28: Co-occurrence matrix of a *THSP* formed from a collection of images of an animal sperm with low activity.

Correlation Measurement of the Speckle Pattern

The Equation (1.12), proposed by Xu; Joenathan; Khorana (1995), is the correlation of the time history speckle pattern, *THSP*, proposed by Oulamara; Tribillon; Duvernoy (1989a).

The memory of the phenomenon is tested using only the M pixels analyzed by the $THSP$.

$$C_{il} = \frac{\sum_{m=1}^M THSP(m, i) THSP(m, i+l)}{\sqrt{\sum_{m=1}^M THSP^2(m, i) \sum_{m=1}^M THSP^2(m, i+l)}}. \quad (1.12)$$

Finally, the correlation value to a lag “ l ” is calculated as the Equation (1.13)

$$C_l = \frac{1}{N/2} \sum_{i=1}^{N/2} C_{il}, \quad (1.13)$$

where N represents the number of columns of a $THSP$ matrix.

The memory can also be tested providing the Pearson correlation between all the pixels in subsequent images (ZDUNEK et al., 2007; KURENDA; ADAMIAK; ZDUNEK, 2012), as it can be seen in the Equation (1.14),

$$C_{il} = \frac{< (\mathbf{I}_i - \mu_i)(\mathbf{I}_{i+l} - \mu_{i+l}) >}{\sigma_i \sigma_{i+l}}, \quad (1.14)$$

where $\mu_a = < \mathbf{I}_a >$ and $\sigma_a = \sqrt{< (\mathbf{I}_a - \mu_a)^2 >}$ are the spatial mean and standard deviation of all pixels of an image $\mathbf{I}_a$ taken in the instant a .

Inertia Moment

The Inertia Moment (IM) is based on the construction of the Time History Speckle Pattern ($THSP$) first proposed by Oulamara; Tribillon; Duvernoy (1989a). Of generic way, the calculus of inertia moment is presented in the Equation (1.15),

$$IM = \sum_i \sum_j \frac{COM(i, j)}{Normalization} |i - j|^2 \quad (1.15)$$

The normalization shown in this equation can be the one proposed by Arizaga; Trivi; Rabal (1999), which makes the sum of values in each line of the COM equal to one, or other normalizations such as the one proposed by Cardoso; Braga (2014) in order to reduce the effect of inhomogeneities in the analyzed images.

Average Value of Difference

The Average Value of Difference (AVD) in the Equation (1.16), with difference of IM , replaces the square operation by the absolute value, being a way to overcome the effect of the square that increases the perception of high changes in the $THSP$, rather than the low changes, thus distorting the outcomes (CARDOSO; BRAGA, 2014).

$$AVD = \sum_i \sum_j \frac{COM(i, j)}{Normalization} |i - j| \quad (1.16)$$

Alternative Outcomes of *IM* and *AVD*

It is possible to have many variations of *IM* and *AVD* regarding some changes in their equations, for instance, related to normalization of the data and the way the outcomes are presented. The outcomes can be used apart or together in order to best evaluate the phenomenon. More details can be read in the other two sections of the book.

1.7.3 Frequency Analysis

The frequency analysis of the biospeckle laser data is an useful way to isolate in the signal of each pixel in time the components that can match to a particular biological phenomenon. There are some techniques to transform the signal from time to frequency domain. We will present here a library based on filter banks to deal with the biospeckle signal. This filter bank can be interpreted as

- A step of a filtering in frequency bands,
- A step of discrete wavelet transform, or
- A step of continuous wavelet transform of a discrete signal.

The aim is usually the transformation of the signal to the frequency domain in order to manipulate it easily. The elimination of some frequency components is the common action before the reconstruction of the signal in the time domain using the inverse transformation. The elimination allows numerical or graphical analysis of the biospeckle data without some non-desirable components that are assumed to be linked to a particular biological phenomenon. For instance, if you want to isolate the respiration of a tissue, you can eliminate all the components related to other phenomena, and then analyze the data only with the supposed components linked to the respiration process (ALVES; BRAGA; VILAS-BOAS, 2013).

Filtering in Frequency Bands

It is possible to construct a scheme of filtering in frequency bands using a filter bank. A filter bank step separates the information in two complementary parts so that the summation of outputs is equal to the input (Figure 1.29a). Where a signal DATA is divided in the output signals DATA1, DATA01, DATA001 and DATA000. DATA represents the information, for all samples k , of a pixel $I_k(x,y)$ or a set of pixels $\mathbf{I}_k$. The Figure 1.29c shows the frequency response for the system in the Figure 1.29a. Each one of the output signals, with sampling frequency F_s , has the information of only a frequency band and it can be processed with all methods of biospeckle analysis seen in the previous sections. The reconstruction of original signal DATA can be reached summing all the signals. The variables H_0 , H_1 , G_0 , G_1 , W_0 and W_1 represent filters (see Equation (1.17)), so that the filters in cascade can be replaced by an equivalent, as in the Figure 1.29b, where the operator “*” represents the convolution. Then, for example, in the case presented here it is right to say that

$$\begin{aligned}
 \text{DATA1} &= \text{DATA} * H_1 \\
 \text{DATA01} &= \text{DATA} * H_0 * G_1 \\
 \text{DATA001} &= \text{DATA} * H_0 * G_0 * W_1 \\
 \text{DATA000} &= \text{DATA} * H_0 * G_0 * W_0
 \end{aligned} \tag{1.17}$$

In the scheme of Figure 1.29, when DATA contains N samples, the outputs DATA1, DATA01, DATA001 and DATA000 also contain N samples.

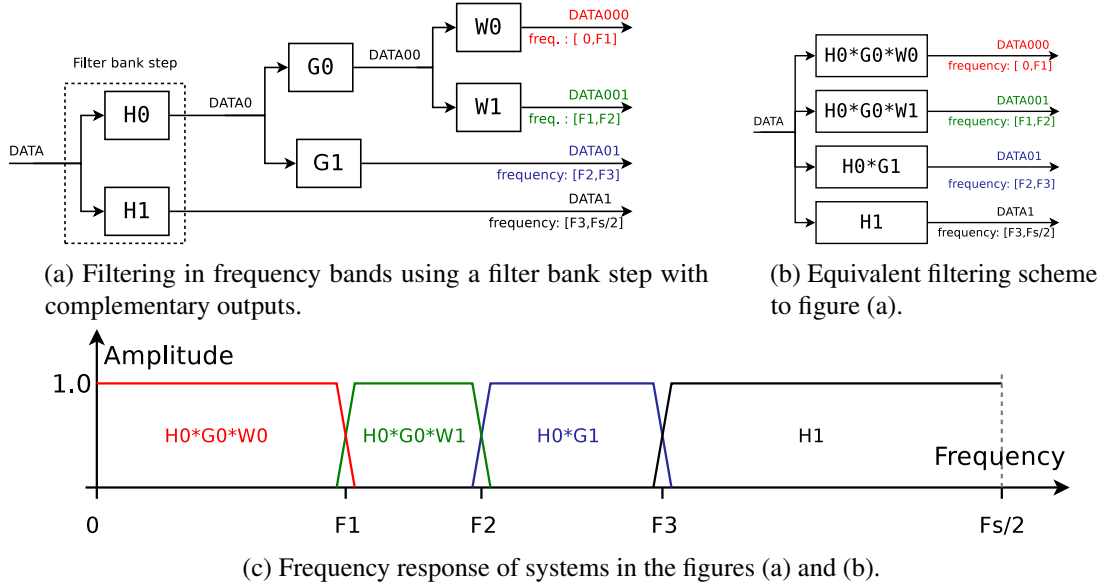


Figure 1.29: Filtering in frequency bands using a filter banks.

Discrete Wavelet Transform

When we decompose a signal, with N samples, in frequency bands using a Discrete Wavelet Transform (*DWT*) (VETTERLI; HERLEY, 1992), our intention is to transform the signal to a new space where only a reduced number of samples (coefficients) is necessary to represent the information of a specific frequency band from the original signal. Here, it is important to remember that these coefficients don't have an immediate relation with the original signal, so that to get the original signal it is necessary to make the Inverse Discrete Wavelet Transform (*IDWT*).

The objective of making the *DWT*, in the context of a biospeckle analysis, is to separate the signal in coefficients in order to be easy to eliminate frequency bands before the reconstruction of the signal, and thus apply all the methods of biospeckle analysis seen in the previous sections. Alternatively, it is possible to make a *IDWT* reconstruction of the coefficients of a specific frequency band and to make an analysis similar to the filtering in frequency bands using a filter bank.

In the Figure 1.30a, it can be seen an implementation of a *DWT* using blocks of a Quadrature Mirror Filter (*QMF*) (AGRAWAL; SAHU, 2013) in analysis mode. Each *QMF* analysis block has a filter pair arranged in quadrature, so that H0 is a low pass filter with cut-off frequency at $1/4$ of sampling frequency F_s , and H1 is a high pass filter with cut-off frequency at $F_s/4$. The filters are called in quadrature because the frequency response of H1 filter is a mirror image of frequency response of H0 filter with respect to $F_s/4$ (or $\pi/2$ for a sampling frequency normalized to 2π). Finally, before delivering the output, the *QMF* filters have two down-samplers by 2 (D) (DISTEFANO; STUBBERUD; WEHBRING, 2011). Thereby, each *QMF* analysis block separates your input data in two parts (set of coefficients), where each one has the information representing half of the input bandwidth. The output variable with the same name of input variable plus a 0

represents the low part of frequency whereas the variable that adds a 1 represents the high part of frequency.

Figure 1.30a shows the signal $DATA$ with N samples, which represents the information for all samples k of a pixel $I_k(x, y)$ or a set of pixels $\mathbf{I}_k$; being decomposed in the coefficients $DATA00$, $DATA01$, $DATA10$ and $DATA11$, with $\lceil N/4 \rceil$ samples (being $\lceil a \rceil$ the ceil function of a). In the scheme of decomposition, Figure 1.30a, the coefficients represent the information of frequency bands with the same bandwidth as can be seen in the Figure 1.30c.

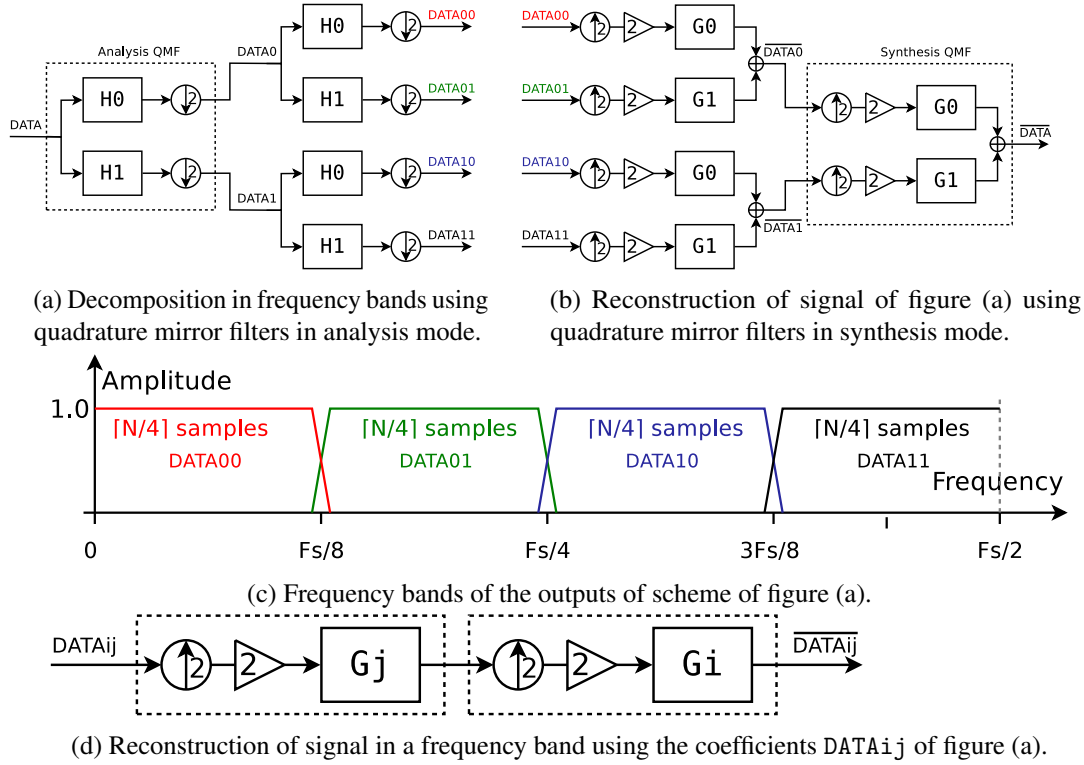


Figure 1.30: Decomposition in frequency bands and reconstruction using DWT .

Figure 1.30b shows the reconstruction $\overline{DATA}$ of the signal $DATA$ from the coefficients obtained in the scheme of Figure 1.30a using quadrature mirror filters in synthesis mode. Similarly to analysis mode, and in reverse, the synthesis mode uses two up-samplers ($\uparrow 2$) (DISTEFANO; STUBBERUD; WEHBRING, 2011) and two filters $G_0 = H_0$ and $G_1 = -H_1$, additionally the synthesis mode needs a gain stage of 2 ($\times 2$). An important rule to perfect reconstruction demands that the Z transform, $H_0(Z)$, of H_0 filter should satisfy the Equation (1.18),

$$H_0^2(Z) - H_0^2(-Z) = a_0 Z^{a_1}, \quad (1.18)$$

a_0 and a_1 being real values. Specifically, to the case of synthesis, for the block proposed here, to have a perfect reconstruction with a similar amplitude of input signal, it needs that $a_0 = 1$.

In some cases, we need to analyze only some frequency bands, and we want to save processing time reconstructing only these bands. For these cases, it is convenient to use a reconstruction scheme as showed in the Figure 1.30d, where the coefficients $DATA_{ij}$, for any $i, j \in \{0, 1\}$, are

used to get $\overline{\text{DATA}i j}$ that represents a portion of $\overline{\text{DATA}}$ in the frequency band ij (Equation 1.19). Therefore, it is correct to say that

$$\overline{\text{DATA}} = \sum_{i,j} \overline{\text{DATA}i j}. \quad (1.19)$$

Continuous Wavelet Transform of a Discrete Signal

We can decompose a discrete signal $x(k)$ in components $w_a(k)$ (Equation 1.20) both with N samples, using a Continuous Wavelet Transform (CWT) (POPOV; ZHUKOV, 2009). Therefore, we analyze the signal in a new space where each scale factor, relative to a , represents the behavior of a frequency band for each sample k . Thus, the general expression of CWT of a discrete signal is

$$w_a(k) = \sum_i x(i) \psi_a(i - k), \quad (1.20)$$

where all the wavelets $\psi_a(k)$ (Equation 1.21) follow the same formation ruler using the scale factor s_a ,

$$\psi_a(k) = \frac{1}{\sqrt{s_a}} \psi_0\left(\frac{k}{s_a}\right) \wedge s_0 = 1, \quad (1.21)$$

the function ψ_0 being called mother wavelet, which is chosen at the discretion of each user. In the Equation (1.21), $\psi_a(k)$ is considered a function whose domain is a set of integers. Therefore, a discrete function $\psi_a(k)$ can be considered as a wavelet (POPOV; ZHUKOV, 2009) if it fulfills the Equation (1.22)

$$1 = \sum_k |\psi_a(k)|^2 \wedge 0 = \sum_k \psi_a(k). \quad (1.22)$$

Established the rules of a CWT, if we consider $\psi_a(k)$ as an even function, the discrete signal $w_a(k)$ (Equation 1.23), in the instant k , can be interpreted as the convolution of a signal $x(k)$ with a wavelet $\psi_a(k)$. So that

$$w_a(k) = x(k) * \psi_a(k). \quad (1.23)$$

In this sense we can say that the CWT, $w_a(n)$, represents a filtered part in a non-causal form of the signal $x(k)$, see Figure 1.31a. The considerations presented in the Equation (1.22) guarantee that $\psi_a(k)$ is a band pass filter with an equal mean spectral energy for any scale s_a .

Whereas previously explained, in the context of biospeckle analysis, the Figure 1.31b shows the signal DATA, that represents the information for all samples k of a pixel $I_k(x, y)$ or a set of pixels $\mathbf{I}_k$, being decomposed in $\{\text{DATA0}, \text{DATA1}, \text{DATA2}, \dots, \text{DATAM}\}$ using a CWT. The Figure 1.31c shows an example of frequency responses of a system in the CWT decomposition (Figure 1.31b), using a Morlet wavelet (MORLET et al., 1982a; MORLET et al., 1982b) as mother, for the case of scale factors with form $s_a = 2^a$. Therefore, as it can be seen, the work with CWT gives us a restriction of filter family, but in the other hand, it gives us a wide freedom to use the scale factor. For this

reason when we use a *CWT* we should be careful in choosing a mother wavelet to have a frequency response of the system that fulfills our requirements.

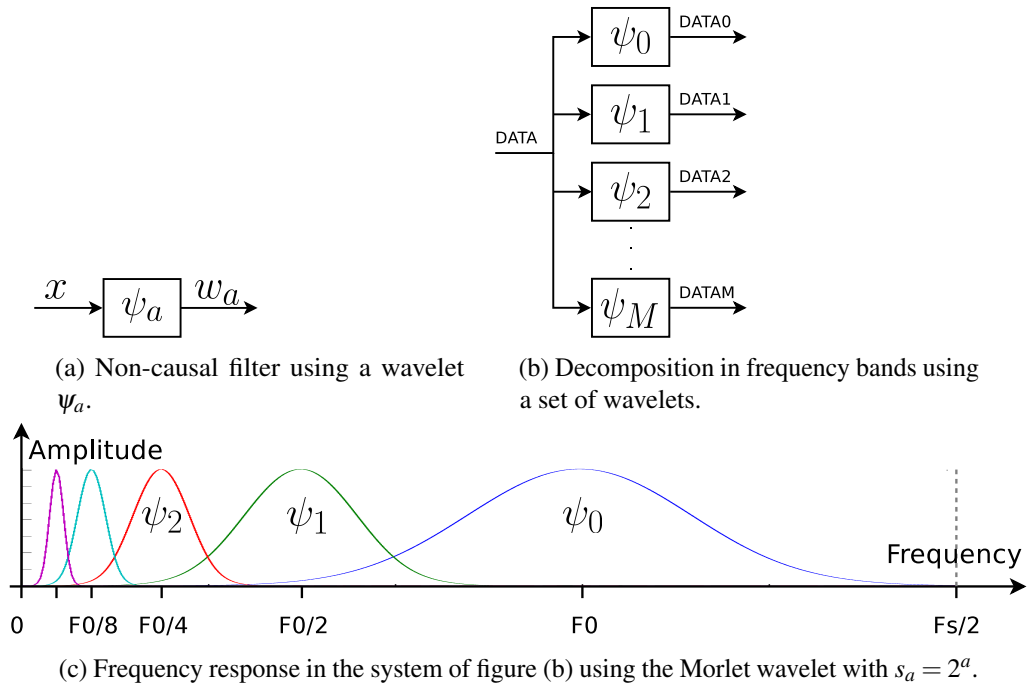


Figure 1.31: Decomposition in frequency bands using *CWT*.

1.8 Frequently Asked Questions - FAQ

1.8.1 How Should I Mount my Experiment?

First of all, you should have a stable table placed in a dark room if possible. The laser is your second step, and it must be a stable laser, which means it must maintain its intensity over time, in order to not affect the speckle pattern formed. The way you will be illuminated using the laser will vary, and it can be done by forward or back-scattering. The laser should form the speckle pattern upon the sample in order you can see the grains with a good contrast, so that avoiding saturation or under-exposition. Place the camera in order to get the best image of the speckle pattern avoiding grains too small, i.e. smaller than the size of a pixel.

Should it be Forward or Back-scattering?

If I were you, I would start doing test by means of the back-scattering approach, even if you have a transparent sample.

Must the Laser be of HeNe?

No. We are using diode lasers with stable power sources, and the results are reliable. The mode-hopping phenomenon has not compromised our analysis so far.

How Should the Grains Appear?

They should have a good contrast, and at least you have to image a grain bigger than the size of a pixel.

How Should the Camera be Chosen?

If I were you, I would try a *USB* camera. It is easier to control. But pay attention to the automatic zoom provided by webcams. You must disable the automatic function before using this sort of cameras to acquire images. Yes, webcams can be used, though they do not offer conditions to attach a macro to magnify the images and, in some cases, their resolution are very low.

Does the Presence of Water Interfere in the Results?

The water in biological material is the main feature to provide the observation of movements. The water activity is thus what the biospeckle laser can monitor since it is highly linked to the moisture of a tissue. For example, a seed with 11 % of water content does not present any expression of activity under the biospeckle. In order to observe the biological activity in a seed one needs to imbibe it to 18 % at least.

In addition, you have to be sure of the absence of free water over the surfaces, since it will produce evaporation and thus a high intense activity observed by the biospeckle.

Does the External Light Interfere in the Results?

Yes, the external light will interfere in your images, thereby reducing the quality of the speckle pattern.

Does the Mechanical Vibration in the Environment Interfere in the Results?

Yes! This is a key point in the use of biospeckle, since it is very sensitive of movements.

1.8.2 How Should I Analyze my Data?

Does the On-line Analysis Help?

Yes! Nowadays the on-line analysis is our first step, and it helps a lot to have the first view of the challenges we will face to acquire and analyze the data.

What is the Need of the Quality Test?

The Quality Test that we suggest is a collection of three procedures you should take in order to avoid saturation of your images, under-exposure of light, incompatibility of the camera speed with the phenomenon, and the numerical analysis in transient regions.

Numerical or Graphical? What Sort of Analysis do I Have to Use?

It depends. If your sample is homogeneous, i.e. the activity should be the same in the *ROI*, the numerical analysis allows you to get information to carry on the statistical analysis. It is relevant to provide statistical analysis when you deal with biological data, in order to know the level of variation of the data. However, if you want to map the sample, thus if you want to check the relative levels of activity within your image, the graphical approach is the one.

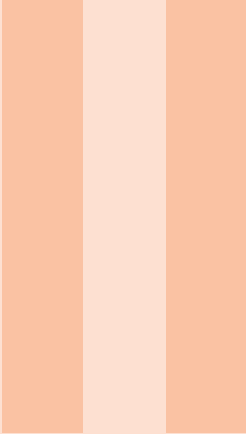
Is the Frequency Analysis Necessary?

It depends. Most of the time you can get the information you want directly from the data. However, if there are some phenomena affecting the final outcome, you will probably need to isolate the phenomenon you want by means of frequency approaches.

1.9 Exercises

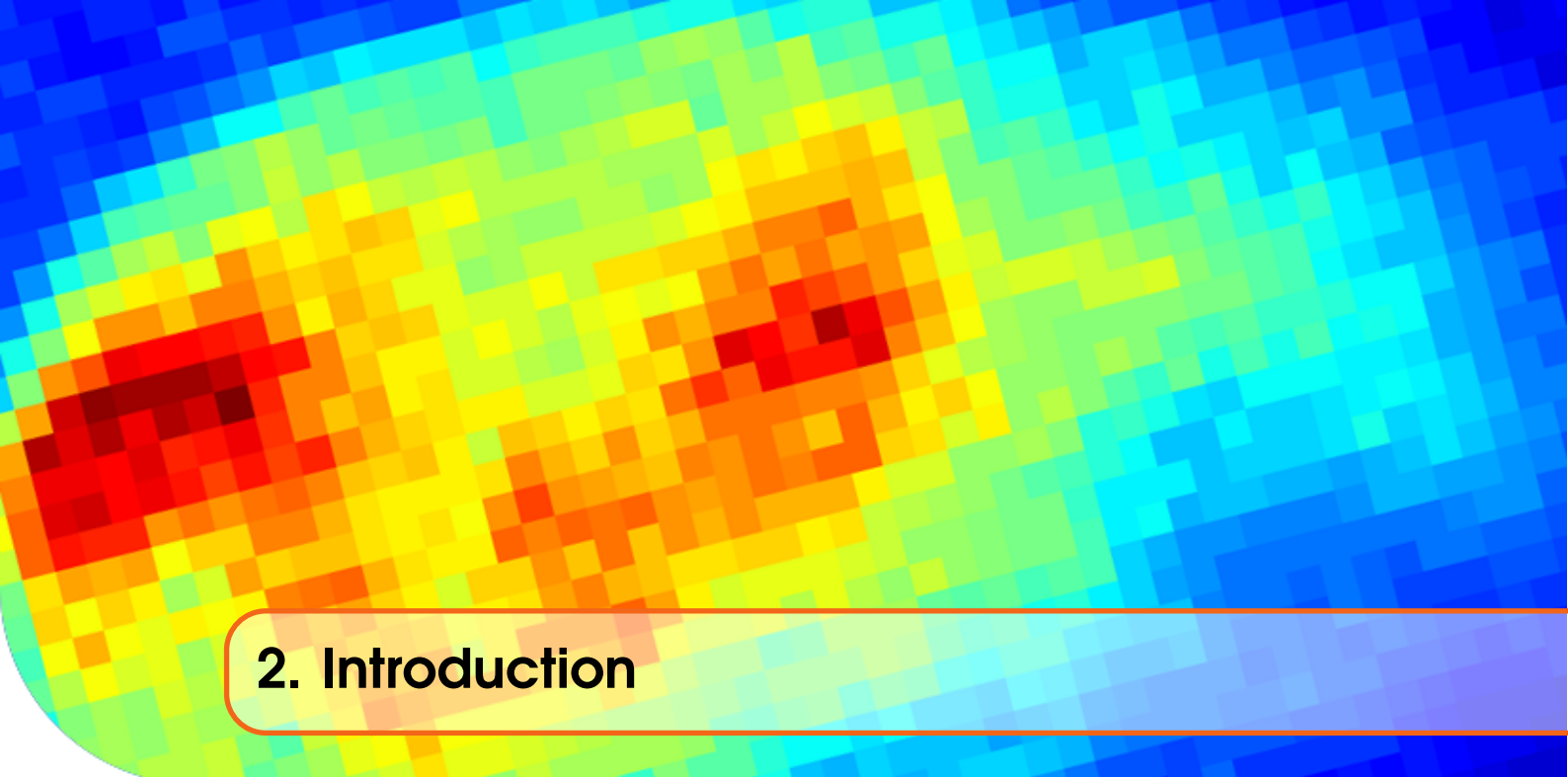
What do we Expect you Should Know About the Theme Before Use the Biospeckle

- You should know about laser features.
- You should know about coherent light.
- You should know about scattering of light.
- You should know about electromagnetic wave interference.
- You should know about Fourier analysis.
- You should know about statistical analysis of biological samples.
- You should know basic programming.



Part 2: BSL Tool Library User Manual

2	Introduction	57
2.1	Overview of BSL Tool Library	
2.2	News	
3	Getting Started with BSL Tool Library ..	59
3.1	User and System Requirements	
3.2	Good Programming Practices	
3.3	Installing BSL Tool Library	
4	BSL Tool Library <i>cookbook</i>	69
4.1	Image Datapack	
4.2	Time History Speckle Pattern	
4.3	Co-Occurrence Matrix	
4.4	Probability Mass Function	
4.5	Numerical Methods	
4.6	Graphic Methods	
4.7	Quality Tests	
4.8	Frequency Analysis	
4.9	Extra Functions	
5	Additional Informations	109
5.1	Frequently Asked Questions	
5.2	Interesting Links	
5.3	BSL - Picture Packages	



2. Introduction

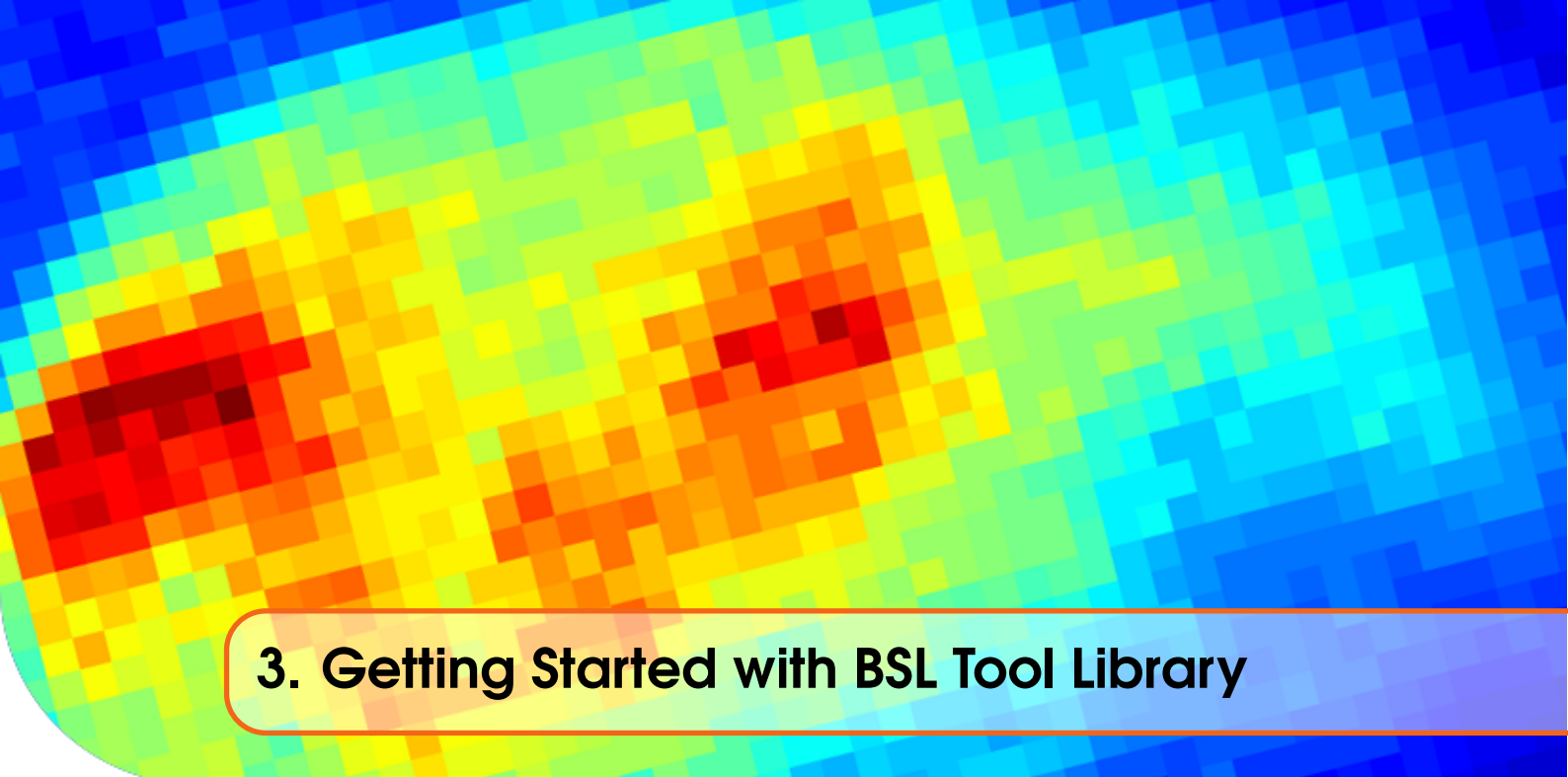
2.1 Overview of BSL Tool Library

The Bio-Speckle Laser (*BSL*) phenomenon can be studied by means of many technics, and in the literature there exists many works treating this topic with numeric (BRAGA et al., 2011; ARIZAGA; TRIVI; RABAL, 1999) or graphics (FUJII; ASAKURA, 1975; ARIZAGA et al., 2002; NOTHDURFT; YAO, 2005) methods. In this sense, the *BSL Tool Library* (*BSLTL*) (BSLTL-PROJECT-GROUP, 2016) is presented here as a free software solution based on digital image processing of speckle pattern images from *BSL* applications.

In the next sections, all image collections containing the time history of speckle patterns (OULAMARA; TRIBILLON; DUVERNOY, 1989b) will be called “datapack”.

2.2 News

- Functions to load in a package a set of images.
- Functions to deal with many time history speckle patterns.
- Functions to analyse numerically the activity of *BSL*. For example, AVD (BRAGA et al., 2011), inertia moment (ARIZAGA; TRIVI; RABAL, 1999), etc.
- Functions to create maps of activity of *BSL*. For example Fujii (FUJII; ASAKURA, 1975), generalized differences (ARIZAGA et al., 2002), standard deviation (NOTHDURFT; YAO, 2005), etc.
- Functions to test the quality (CARDOSO; BRAGA; RABAL, 2012) of a collection of images.



3. Getting Started with BSL Tool Library

3.1 User and System Requirements

In order to use the *Biospeckle Laser Tool Library* it is important to have some previous knowledges and an equipment with minimal characteristics, in the next subsections we will show the necessary requirements.

3.1.1 User Requirements

When a user starts the study of *BSL* phenomenon and the use of *BSL Tool Library*, he will perceive the need to know some concepts. Next we will enumerate a list that, in our view, includes everything a user must know.

Speckle laser theory: To use the *BSL Tool Library* it is necessary a basic knowledge of *BSL* phenomenon; for example, knowing in which cases to use an activity index or another, or what procedures make a specific technique. In any case, the documentation of library functions has references to the scientific paper that defines or describes the technique.

Algorithm: The *BSL Tool Library* was written to be easy to use, with a modular focus and imperative programming style. Thus, an user only needs a basic knowledge of algorithm to begin working with the library.

Knowledge of OCTAVE or MATLAB: Given that the library is written in the code style and syntax of OCTAVE and MATLAB, it is necessary that the user has a minimal knowledge of some of this two interpreters. In this sense, the library works well in both, and this is due to the high compatibility between OCTAVE and MATLAB.

3.1.2 System Requirements

At this point, it is assumed that the user has basic abilities to manipulate the libraries in OCTAVE or MATLAB softwares, and also he has a set of images to test the *BSL* phenomenon. Thus, the user only need process your images. However, we have images of *BSL* application in seeds (maize and coffee) in a public repository, which will help you start only using the software first. The download paths are presented in section 5.3. With this consideration, to use the *BSL Tool Library* it is needed the next computational requirements:

- **Associated software needed:** Given that the library is written in the code style and syntax of OCTAVE and MATLAB, it is necessary to have any of these two interpreters installed.
- **Operating system needed:** OCTAVE and MATLAB interpreter programs can be installed in many operating systems; for example the Microsoft Windows family or all based in GNU with Linux kernel.

Finally, we list below a detailed description of the requirements of the programs aforementioned.

MATLAB requirements: Any Intel or AMD x86 processor supporting SSE2 instruction set with free disk space of 1 GB, 3–4 GB for a typical installation, 2 GB of RAM and any of these operating systems: Windows 10, Windows 8.1, Windows 8, Windows 7 Service Pack 1, Windows Vista Service Pack 2, Windows XP Service Pack 3, Windows XP x64 Edition Service Pack 2, Windows Server 2012, Windows Server 2008 R2 Service Pack 1, Windows Server 2008 Service Pack 2, Windows Server 2003 R2 Service Pack 2, Ubuntu 14.04 LTS and 14.10, Red Hat Enterprise Linux 6 and 7, SUSE Linux Enterprise Desktop 11.3+, Debian 7.x. Here, it is important to note that currently MATLAB 2015a “*do not support architectures of 32 bits for operating systems based in GNU/Linux*”.

OCTAVE requirements: Given that OCTAVE is distributed both as binary and source code it can be founded in many operating system. Currently (OCTAVE 4.0) exist compiled binary versions of OCTAVE for operating systems of Microsoft Windows family, operating system based in GNU/Linux and for Android. In the case of source code, OCTAVE requires approximately 400 MB of disk storage to unpack and compile without debugging symbols (significantly greater, 1.4 GB, if you compile with debugging symbols). Once installed, OCTAVE requires approximately 350MB of disk space (and considerably less, 70 MB, if you don’t build shared libraries or if the binaries and libraries do not include debugging symbols).

3.2 Good Programming Practices

In this section it will be seen a set of rules needed to work with libraries (packages) written for working on OCTAVE or MATLAB.

When working with a new software project it is necessary to follow some rules, since in many occasions, a same project makes use of many libraries, which are generally written by different developers without connection and communication between themselves. Therefore, the function names, dependencies and versions numbers may have conflicts. Thereby, it is important to follow a set of rules such as:

- Have a place to download the libraries ordered by versions in directories.

- Have a place to download new software projects ordered by directories.
- Have a method to load the libraries, in order to have just what you need.

In the next subsections it will be presented a series of guidelines that can help you to follow these rules.

3.2.1 Where Must I Place my Libraries?

For a best order in the use of libraries in our projects it is recommended that all of them are located in a known directory. All libraries are commonly located in a directory called “lib”, with the “lib” usually located in the user’s home directory (HOME). The complete path of HOME depends of operating system; for example, in operating systems based in GNU/Linux, regularly HOME = '/home/username'; whereas in Microsoft Windows Vista, 7 and 8, regularly HOME = 'C:\Users\username'. In the former cases, “username” is the name defined by the user. Thus, in the next subsections it will be shown how to sort the directories of libraries according to the used computer operating system.

Recommendations in Computer Operating Systems Based on GNU/Linux

The first thing that we need to know before to sort our libraries is to locate the HOME path. It can be found using (in OCTAVE or MATLAB) the function `getenv()` with the next code:

```
HOME=getenv('HOME');
```

Instruction 3.1: Getting HOME path in *GNU/Linux*

This returns the value of the system environment variable HOME (for example, HOME = '/home/username'). It is important to highlight that, in computer operating systems based in GNU with kernel Linux, the HOME directory is also represented by the symbol "~". This directory has a mix of user files and user application configuration files. Thus, we propose that the application libraries be inside “lib” directory in HOME; for example, the libraries that only run over OCTAVE should be in the path

```
~/lib/octave/
```

The same way to the libraries that only run over MATLAB should be in

```
~/lib/matlab/
```

By other side, if the libraries can be run in OCTAVE and MATLAB, it is better to have the libraries in a directory such as:

```
~/lib/octmat/
```

Thus, the "octmat" directory will have a subdirectory tree such as:

```
~/lib/octmat/mylibrary1
~/lib/octmat/myLibrary2
...
~/lib/octmat/libraryOfMyBrother
```

```
~/lib/octmat/libraryOfMyTeacher
...
~/lib/octmat/bsl1
```

where each name that follows "octmat" represents one library.

Recommendations in Computer Operating Systems as Microsoft Windows Vista, 7 and 8

As in the previous subsection, libraries in Microsoft operating system should be grouped in a known directory such as HOME . It can be found using (in OCTAVE or MATLAB) the function `getenv()` with the next code:

```
HOME=getenv('USERPROFILE');
```

Instruction 3.2: Getting HOME path in Microsoft Windows Vista, 7 and 8

This returns, in the variable HOME, the value of the system environment variable *USERPROFILE* (for example, HOME='C:\Users\username'). Once we know this value, we suggest that the application libraries should be in the "lib" directory in HOME ; for example, using the value of HOME assumed before, the libraries that only run over OCTAVE should be in the path: 'C:\Users\username\lib\octave\'. But, for simplify the notation, here we use

```
lib\octave\
```

Thus we understand that the paths are always inside HOME directory. Following the same notation, the libraries that only run over MATLAB should be in

```
lib\matlab\
```

and the libraries that can be interpreted in OCTAVE and MATLAB would be in

```
lib\octmat\
```

So that, the "octmat" directory will have a subdirectory tree such as

```
lib\octmat\mylibraryA
lib\octmat\mylibraryB
...
lib\octmat\libraryOfMyFriend
lib\octmat\libraryOfMyTeacher
...
lib\octmat\bsl1
```

Where each subdirectory inside "octmat" represents a library (package). In the similar way, the directories "octave" and "matlab" also have a subdirectory tree with libraries sorted by name and if necessary by version.

3.2.2 How to Make a New Project?

In the same way we did in the Section 3.2.1, it is recommended to group all the projects inside a known directory. Here, we recommend that all new software projects should be located in HOME inside a directory called “projects” (The way of getting HOME path can be seen in the Instruction 3.1 and 3.2). For example, if we want to create a new code project called “MyCodeTest1”, which can be interpreted in OCTAVE and MATLAB, in computer operating systems based in GNU/Linux, the directory of project “MyCodeTest1” should be located inside the next projects path

```
~/projects/octmat/
```

Thus, the new code will be in the directory '`~/projects/octmat/MyCodeTest1`'. However, if the computer operating system used is Microsoft Windows Vista, 7 or 8, the project directory (following the path notation used in Section 3.2.1) should be located inside

```
projects\octmat\
```

Thereby, if for example HOME='`C:\Users\username`', the code project will be located in the directory '`C:\Users\username\projects\octmat\MyCodeTest1`'.

The projects' path follows the same rule specified in Section 3.2.1, where projects that run in OCTAVE and MATLAB should be inside a directory called “octmat”, projects that only run over OCTAVE should be inside a directory called “octave” and projects that only run over MATLAB should be inside a directory called “matlab”.

3.2.3 How to Load my Libraries?

Once a library has been properly stored (as in the Subsection 3.2.1), it is necessary to have a method so that a new code project, such as the ones defined in the Subsection 3.2.2, can locate and load the libraries. It is important to note that when we use the term “load”, we are talking about the action to access the routines where they are located, typically adding new locations to OCTAVE/MATLAB search path. In the next subsections we show how to load libraries, using OCTAVE or MATLAB.

Loading Manually my Libraries

When loading manually a library we should have in consideration how the routines files are grouped in a library; for example, if we have the routines in a unique directory represented by the variable MY_LIBRARY1_DIR such as

```
MY_LIBRARY1_DIR='C:\Users\username\lib\octmat\mylibrary1';
```

we can add the path MY_LIBRARY1_DIR to OCTAVE/MATLAB search path with:

```
addpath(MY_LIBRARY1_DIR);
```

Instruction 3.3: Adding the directory pointed by variable MY_LIBRARY1_DIR to OCTAVE/MATLAB search path.

In turn, if we have a variable `MY_LIBRARY2_DIR` that represent a library path where the routines are grouped in subdirectories,

```
MY_LIBRARY2_DIR='C:\Users\username\lib\octmat\mylibrary2';
```

the command that adds the path `MY_LIBRARY2_DIR` and the subdirectories, to OCTAVE/MATLAB search path is

```
addpath(genpath(MY_LIBRARY2_DIR));
```

Instruction 3.4: Adding the directory pointed by variable `MY_LIBRARY2_DIR` and subdirectories to OCTAVE/MATLAB search path.

In the last command, the function `genpath()` returns a list of all subdirectories in `MY_LIBRARY2_DIR`.

Automatically Loading my Libraries in MATLAB

The Instructions 3.3 and 3.4 showed the routines to load a library, but to automate this work in MATLAB, we can make use of “startup.m” file. This file is read and interpreted when starting MATLAB, and it is the file to write the routines for loading a specific library. In MATLAB, to load the startup.m file automatically it is necessary to have startup.m in the search path. If you want to know the search path, you can use the command “`path`”. This will return a list with the paths where MATLAB is searching routines (functions and procedures); any path in this list can be used to store the startup.m file, but it is recommended the “user path”, since it is a special MATLAB search path and it is located in user HOME directory. In order to know the actual user path it is necessary to run the code:

```
USER_PATH= userpath();
```

If necessary you can change the user path, for example `'C:\Users\username\anyaddress'`

```
userpath('C:\Users\username\anyaddress');
```

thus, we can write the Instruction 3.3 or 3.4 in the startup.m file, located in `'USER_PATH'` directory, to automatically load a library.

Automatically Loading my Libraries in OCTAVE

In OCTAVE there is a file called “.octaverc”, which is read and interpreted when starting OCTAVE and is used to make personal changes in the OCTAVE environment. In order to locate this file in OCTAVE, it is necessary to store “.octaverc” in some particular paths (for details, test the command: `doc .octaverc`). We will talk now about two of these directories.

~/.**.octaverc** The first place where OCTAVE searches for the `.octaverc` file is in the user HOME directory (a way of to know the location of HOME can be seen in the Instruction 3.1 and 3.2). This file can be used to make personal changes to the default OCTAVE environment, being applied to all projects.

.octaverc The second place where OCTAVE searches for the `.octaverc` file is in the initial startup directory (the current work directory). Usually the startup directory is also the HOME directory,

but depending on how OCTAVE is initiated or configured, the location can be different. This file can be used to make changes in the OCTAVE default environment to adopt in a particular project.

Thus, in order to load some libraries automatically, as showed in Instruction 3.3 and 3.4, we can add these routines in any of these *.octaverc* files.

Installing and loading a Package in OCTAVE

OCTAVE provides an additional method for shared libraries, when these are grouped in a package. An OCTAVE package is a compressed file with a specific internal structure, as it is described in the OCTAVE documentation (EATON et al., 2015) (test the command: `doc pkg`). The package installation is designed to be very simple; for example if we have a library called “mylibrary1” compressed in a package with filename “MyLibrary1Package.tar.gz”, then the installation in a path as “~/lib/octmat” is made with the following OCTAVE code.

```
pkg prefix ~/lib/octmat
pkg install MyLibrary1Package.tar.gz
```

Thus, each time OCTAVE starts we need to use the next command to load the mylibrary1 library/-package.

```
pkg load mylibrary1
```

3.2.4 Other Recommendations

Graphics with Latex Support in OCTAVE 4.0

For rendering graphics with latex code notations, use the next OCTAVE code

```
graphics_toolkit("gnuplot");
```

only “gnuplot” graphics toolkit supports latex rendering in OCTAVE 4.0; such as “fltk” and “qt” are not yet supported. Use the function `available_graphics_toolkits()` to see the complete list of graphics toolkits

```
available_graphics_toolkits();
```

Print Text During Loops in OCTAVE

In order to show information while the OCTAVE scripts are running or immediately after the function `disp()`, we can use the next OCTAVE code

```
page_screen_output(0);
page_output_immediately(1);
```

where the function `page_screen_output(0)` disables the option that sends the output through of the pager, when this is very long. The function `page_output_immediately(1)` enables the output print as soon as it is available.

3.3 Installing BSL Tool Library

In the next subsections three methods to install the *BSL Tool Library* it will be shown (BSLTL-PROJECT-GROUP, 2016). In the two last methods, it is assumed that the user has already downloaded the last version of package from <http://www.nongnu.org/bsltl/> (BSLTL-PROJECT-GROUP, 2016) or at least the first version from <http://repositorio.ufla.br/handle/1/11116> (BRAGA; PUJAICO RIVERA; MOREIRA, 2016). The filename of the compressed file containing the *BSL Tool Library* may change between versions; but the procedures for installing them will be the same (here the current previous version 1.0.2 will be used as example). Once you have downloaded the library, we provide three methods to install it. The first and second are only oriented only for OCTAVE users, and the third method works independently if you are using OCTAVE or MATLAB.

3.3.1 Method 1: Online - Only in OCTAVE

The next OCTAVE code installs the last version of BSLTL package directly from the [octave-forge](#) website(OCTAVE-FORGE-DEVELOPERS, 2016) in the default install directory; thus, it is necessary to have an internet connection during the installation process.

```
pkg install -forge bsltl
```

With this method, the package is configured to be unloaded by default and it should be loaded each time that OCTAVE starts; the command will be:

```
pkg load bsltl
```

After loading, all the functions of *BSL Tool Library* will be available to be used.

3.3.2 Method 2: Offline - Only in OCTAVE

The next OCTAVE code installs the *BSL Tool Library* in the directory '[~/lib/octmat](#)', in its package form (this means it is a tar.gz compressed file), with the filename [bsltl-1.0.2.tar.gz](#) (BSLTL-PROJECT-GROUP, 2016) (or similar).

```
pkg prefix ~/lib/octmat
pkg install bsltl-1.0.2.tar.gz
```

The package is configured to be unloaded by default and it should be loaded each time that OCTAVE starts. The command will be:

```
pkg load bsltl
```

After loading, all the functions of *BSL Tool Library* will be available.

3.3.3 Method 3: In MATLAB or OCTAVE

If *BSLTL* is [uncompressed](#) in a directory such as '[C:\Users\username\lib\octmat\bsltl](#)', it is necessary to add the next code in the top (before the code) of our main project source file.

```
BSLTL_DIR='C:\Users\username\lib\octmat\bsltl\inst';  
addpath(genpath(BSLTL_DIR));
```

The subdirectory '*inst*' is added to the path to load only the functions of the library and not the configuration files. The function `genpath()` generates a list with the directories and sub directories of *BSLTL* package. The function `addpath()` add all directories of *BSLTL* to the search path. More details about automatically loading the library can be seen in the Subsection 3.2.3.

4. BSL Tool Library *cookbook*

In this chapter, you will find a set of examples of *BSL Tool Library (BSLTL)*. All examples assume that there exists an image data package in the directory `IMAGESDIR`; for example this directory can be

```
IMAGESDIR='~/data/cafe-biospeckle/sem1';
```

in operating systems based in *GNU/Linux*, or

```
IMAGESDIR='D:\data\cafe-biospeckle\sem1';
```

in operating systems of Microsoft Windows family.

It is also assumed in all examples that we have at least 128 images called: “1.bmp”, “2.bmp”, “3.bmp”, ... and “128.bmp”. Thus, when we use the examples in the next sections, the variable `IMAGESDIR` should be replaced by the appropriate directory. The image data package used in the examples of this chapter can be downloaded from the [repository](#) (VIVAS; BRAGA, 2015), and they are related to a collection of speckle pattern images from a coffee bean illuminated by a HeNe laser beam.

4.1 Image Datapack

The *BSL Tool Library* calls “datapack” a collection of images grouped in a 3-dimensional matrix, where the third dimension is related to the time domain. The next code, highlighted, uses *BSLTL* to create a datapack of 128 images in bmp format, loaded from image “1.bmp” to image “128.bmp” (VIVAS; BRAGA, 2015).

```
DATA=datapack(IMAGESDIR, ' ', 1, 128, 'bmp');
```

Another interesting example is the loading process of 128 images in “bmp” format, loaded from image “cafe1.bmp” to image “cafe128.bmp”. The word “cafe” means coffee in Portuguese.

```
DATA=datapack(IMAGESDIR, 'cafe', 1, 128, 'bmp');
```

In both cases, the datapack DATA is a 3-dimensional matrix with NLIN lines

```
NLIN=size(DATA, 1);
```

NCOL columns

```
NCOL=size(DATA, 2);
```

and NTIMES=128 elements (matrices).

```
NTIMES=size(DATA, 3);
```

Sometimes we see that the images in a datapack DATA have unwanted regions, thus in order to create a new datapack without these regions, it is possible to crop the DATA with `datacut()` function.

```
DATACUT=datacut(DATA);
```

This function opens a window (Figure 4.1) for interactive graphic selection of two points to define a new region.

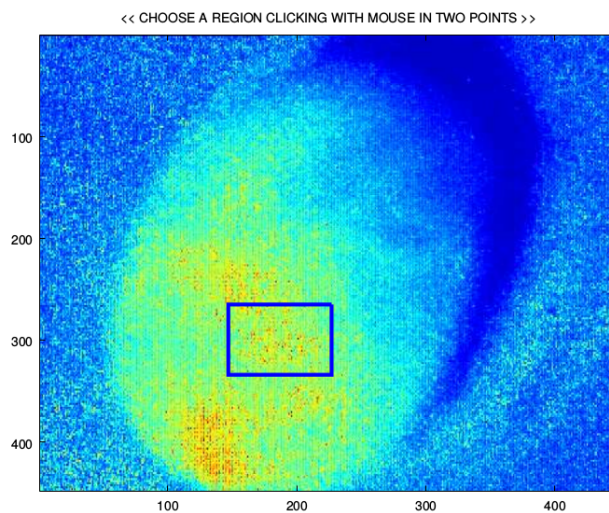


Figure 4.1: Cropping one image of a datapack.

The datapack DATACUT is also a 3-dimensional matrix with NTIMES=128 images, but NLIN and NCOL are reduced.

Another interesting function is `datapack_to_gif()`. This function creates a “gif” file with some characteristics as observed in the example

```
datapack_to_gif(DATA, 'image.gif', 10, jet, 1);
```

This function creates, from a datapack DATA, an image with the filename “image.gif”, 10 frames, and color map “jet” with a delay of 1 frame per second. To see other options of the color map you can use the OCTAVE/MATLAB comand: `help colormap;`.

4.2 Time History Speckle Pattern

The creation of a time history speckle pattern from a datapack DATA (OULAMARA; TRIBILLON; DUVERNOY, 1989b; XU; JOENATHAN; KHORANA, 1995), can be performed using the following code:

```
DATA=datapack(IMAGESDIR, '', 1, 128, 'bmp');
THSP=thsp(DATA, 1, 240);
```

where the function `thsp()` makes the time history speckle pattern of one line or column. In this case, “1” indicates that a line will be used (to select a column you must use the value “2”), and the selected line is “240”. The result is a 2-dimensional matrix called THSP with the number of lines equal to the number of columns in the speckle pattern image and the numbers of columns equals to the number of images of DATA, in this case 128 images, meaning the time. Other examples of function `datapack()` can be found in the Section 4.1.

To see the content of THSP matrix, it is possible to use the next code:

```
imagesc(THSP);colormap gray;grid;
xlabel('k');
ylabel('Points in the image datapack');
title('THSP');
```

The function `imagesc()` shows the THSP matrix using all color map levels. Here we used the gray scale color map. The result can be seen in the Figure 4.2.

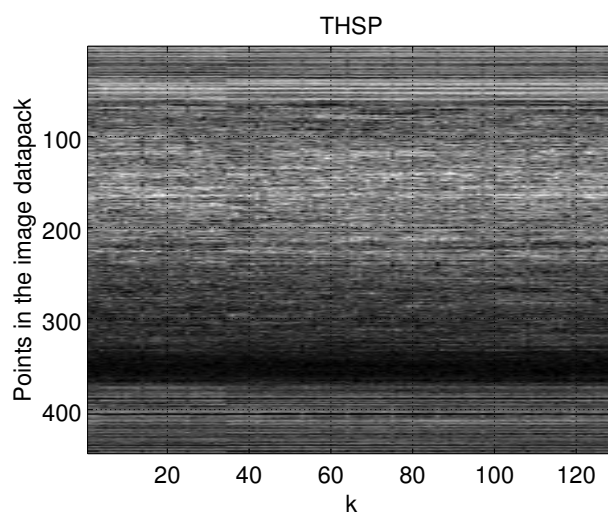


Figure 4.2: Time history speckle pattern of a line from a biospeckle collection of images in a datapack.

Additionally, if you want to make a time history speckle pattern of a set of random uniformly selected points (PAPOULIS, 1984), you can use the next code:

```
DATA=datapack(IMAGESDIR, '', 1, 128, 'bmp');
[THSP POINTS]=thsp_random(DATA, 200);
```

where the function `thsp_random()` uses 200 points to create the time history speckle pattern in the THSP matrix; optionally, the function returns the used points.

Another random selection function is the `thsp_gaussian()`, which uses a Gaussian distribution (PAPOULIS, 1984) around one point. The next code

```
DATA=datapack(IMAGESDIR, '', 1, 128, 'bmp');
[THSP POINTS]=thsp_gaussian(DATA, 200, 25, [250, 250], 'on');
```

uses a datapack called DATA, to get the time history speckle pattern from 200 points randomly selected around the $(line, column) = (250, 250)$; these data are loaded in the variables THSP and POINTS respectively. The points follow a Gaussian distribution of mean zero and standard deviation equal to 25. The optional function parameter 'on' enable the graphic output of selected points as can be seen in the Figure 4.3.

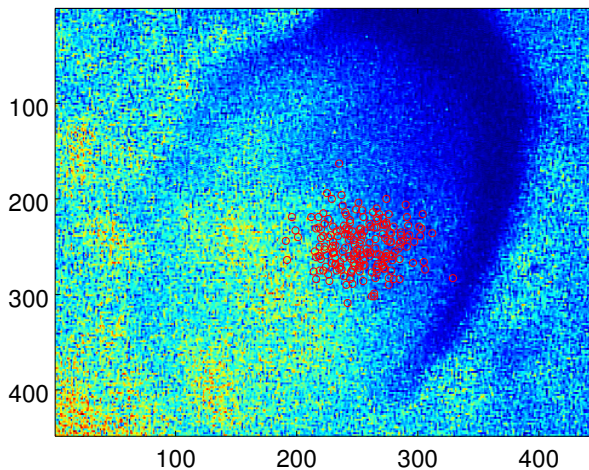


Figure 4.3: Randomly selected points following a Gaussian distribution.

4.3 Co-Occurrence Matrix

In order to get a co-occurrence matrix, COM, from a set of images with 8 bits by pixel, we use the `coom()` function. This function follows the procedure described in (ARIZAGA; TRIVI; RABAL, 1999). The next code

```
DATA=datapack(IMAGESDIR, '', 1, 128, 'bmp');
THSP=thsp(DATA, 1, 240);
COM=coom(THSP);
```

shows how to get the COM matrix from the time history speckle pattern, THSP, of a line in a collection of images. The line used in this example is the number 240 of datapack DATA. Examples of functions `datapack()` and `thsp()` can be seen in the Sections 4.1 and 4.2.

If you want to plot the COM matrix you can follow the next code:

```
imagesc(COM);colorbar;grid;
xlabel('j');
ylabel('i');
title('COM');
```

The graphic seen in the Figure 4.4 represents the dispersion of subsequent pixels in the lines of the THSP matrix, i.e. the jump of intensity level “ i ” to “ j ” of two consecutive intensity samples within a THSP matrix. The function `imagesc()` shows the COM matrix in color, representing the amount of times the jumps of $i \rightarrow j$ happens.

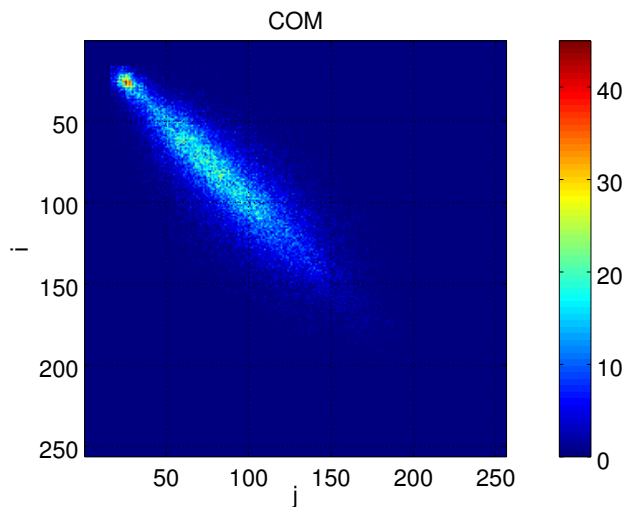


Figure 4.4: COM of a line in image datapack with the color bar representing the amount of times the jumps occurred.

4.4 Probability Mass Function

In the previous section, we have seen the co-occurrence matrix, which can be interpreted as a histogram of transitions between two consecutive instants of a same pixel in the *THSP*, i.e. presented in Figure 4.2 with the transitions along each line. Thereby, with some arithmetic changes regarding the normalization, the COM matrix can be interpreted as probability mass function. In this sense, in the next sections we will see two types of normalizations used in *BSL Tool Library*.

4.4.1 Probability Mass Function of Absolute Difference

One way to interpret the co-occurrence matrix is calculating the probability mass function of absolute difference of the intensities between two consecutive instants of a same pixel in a *THSP*. Thereby $Pr(|i - j| = z)$ represents the probability of an intensity jump from level “ i ” to “ j ” so that $|i - j| = z$. In order to calculate this probability we can use the function `pmfad()`. The next code shows an example of using this function.

```
DATA=datapack(IMAGESDIR, ' ', 1, 128, 'bmp');
THSP=thsp(DATA, 1, 240);
COM=coom(THSP);
```

```
[Pr Z]=pmfad(COM);
```

The vectors `Pr` and `Z` have 256 elements, representing the probability $Pr(|i - j| = z)$ and the values of z restricted to $0 \leq z \leq 255$, being z an integer; both vectors are evaluated from the co-occurrence matrix `COM`. In this case, the matrix `COM` is calculated from the time history speckle pattern (THSP) at line number 240, which was formed by images from the directory pointed by the `IMAGESDIR` variable. Additionally, examples using functions `datapack()`, `thsp()` and `coom()` can be found in the Sections 4.1, 4.2 and 4.3 respectively.

In order to see the `Pr` vector content, it is possible to use the next code:

```
plot(Z,Pr,'--o'),grid
xlabel('|i-j|=z');
ylabel('Probability');
title('Probability Mass Function');
xlim ([0 50]);
```

The result can be seen in the Figure 4.5. Here, it is easy to note that the highest occurrence probability is $Pr(|i - j| = 1)$.

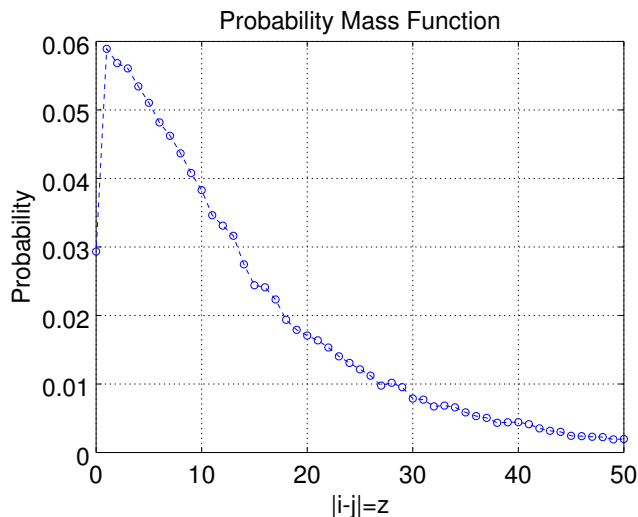


Figure 4.5: Absolute probability mass function of a line from a collection of images.

4.4.2 Probability Mass Function of Regular Difference

Another way to interpret the co-occurrence matrix is calculating the probability mass function of a regular difference of the intensities between two consecutive instants of a same pixel in a *THSP*. This will be called here as $Pr((j - i) = w)$, and represents the probability to have an intensity jump from level “ i ” to “ j ” so that $(j - i) = w$. To calculate this probability, we can use the function `pmfrd()` as shown in the next code.

```
DATA=datapack(IMAGESDIR, ' ', 1, 128, 'bmp');
THSP=thsp(DATA, 1, 240);
COM=coom(THSP);
[Pr W]=pmfrd(COM);
```


The vectors Pr and W have 511 elements, representing the probability $\text{Pr}((j-i) = w)$ and the values of w restricted to $-255 \leq w \leq 255$, being w an integer. Both vectors are evaluated from the co-occurrence matrix COM , which in turn is calculated from the time history speckle pattern (THSP) at the line number 240 of a datapack that was formed by images from the directory pointed by the `IMAGESDIR` variable. Additionally, examples using the functions `datapack()`, `thsp()` and `coom()` can be found in the Sections 4.1, 4.2 and 4.3 respectively.

The probability mass function Pr is shown in the next code:

```
plot(W,Pr, '--o'), grid on
xlabel('(j-i)=w');
ylabel('Probability');
title('Probability mass function');
xlim ([-50 50]);
```

The result is shown in Figure 4.6. In the image, it is easy to see that the mean value is $w = w^*$, being $w^* \approx 0$. It is very interesting to analyze the case when $w^* > 0$ and $w^* < 0$; the first case indicates that the light intensities are growing and will come into saturation. In the second case, the light intensities are decreasing and will come into darkness.

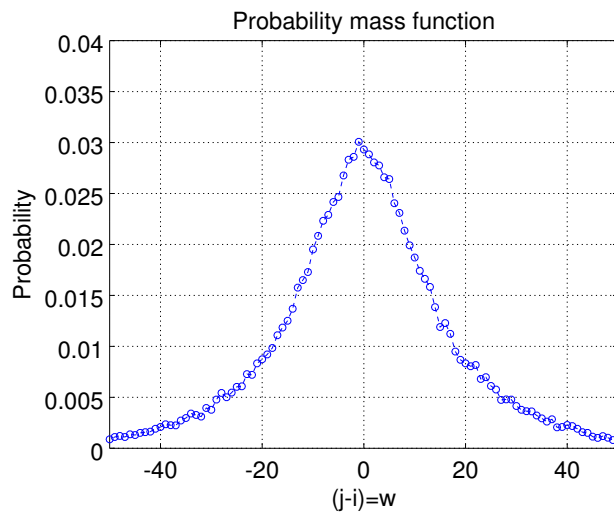


Figure 4.6: Regular probability mass function of a line from a collection of images.

4.5 Numerical Methods

4.5.1 Inertia Moment

The next code shows a way to calculate the inertia moment (ARIZAGA; TRIVI; RABAL, 1999) with two types of normalizations of the co-occurrence matrix (CARDOSO; BRAGA, 2014; ARIZAGA; TRIVI; RABAL, 1999).

```
DATA=datapack(IMAGESDIR, '', 1, 128, 'bmp');
THSP=thsp(DATA, 1, 240);
COM=coom(THSP);
[IM1 IM2]=inertiamoment(COM, 2)
```


The inertia moment is calculated by the function `inertiamoment()` from a co-occurrence matrix COM, which in turn is calculated using the time history speckle pattern (THSP). In this example, it we used the line number 240 of the datapack DATA that was formed by images from the directory pointed by the IMAGESDIR variable. Additionally, examples using the function `datapack()`, `thsp()` and `coom()` can be found in the Sections 4.1, 4.2 and 4.3 respectively. The previous code will return the information:

```

Loading images from:~/data/cafe-biospeckle/sem1
Please wait...
DATA Pack loaded... [OK]
IM1 = 358.57
IM2 = 1.0533e+05

```

By default, the function `inertiamoment()` returns the inertia moment IM1 with the normalization of the co-occurrence matrix proposed by Cardoso; Braga (2014); the parameter “2” in the function `inertiamoment()` indicates that the function will return a second parameter IM2 of type “2” (with the normalization of the co-occurrence matrix proposed by Arizaga; Trivi; Rabal (1999)).

It is interesting to note that, if we define the expectation operator $E[.]$ and use the Cardoso’s normalization, then the inertia moment $IM1$ is the second moment $E[(i-j)^2]$ of the probability mass function $Pr((i-j)=w)$ returned by function `pmfrd()`, see Section 4.4.2. In this way, if we assume $E[i-j] \approx 0$, then the value $\sqrt{IM1} = 18.936$ can be considered as the standard deviation of the probability mass function returned by function `pmfrd()`. The next code generates the Figure 4.7, which shows these relations.

```

Pr=pmfrd(COM);
Z=[-255:255];

figure(1)
plot(Z,Pr, '--o', sqrt(IM1)*ones(1,17), [0:16]*0.05/16, '--s'),grid

hx=xlabel('(i-j)=w');
hy=ylabel('Probability');
ht=title('Probability Mass Function');
hl=legend('pmfrd()', 'sqrt(IM1)', 'location', 'northwest');

xlim([-30 30]);

```

4.5.2 Absolute Value of the Differences

The next code shows many ways to calculate the Absolute Value of the Differences (AVD) (BRAGA et al., 2011) with two normalizations of co-occurrence matrix (CARDOSO; BRAGA, 2014; ARIZAGA; TRIVI; RABAL, 1999).

```

DATA=datapack(IMAGESDIR, '', 1, 128, 'bmp');
THSP=thsp(DATA, 1, 240);

```

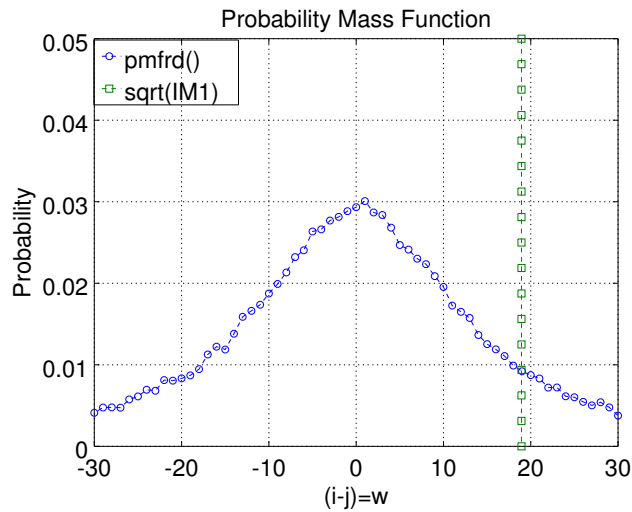


Figure 4.7: $\sqrt{IM1}$ and `pmfrd()` function of a line from a collection of images.

```
COM=coom(THSP);
[AVD1 AVD2 AVD3 AVD4]=avd(COM,2,3,4)
```

Where

AVD1 is the value of *AVD* with the normalized co-occurrence matrix proposed by Cardoso; Braga (2014), being $AVD1 = E[|i - j|]$.

AVD2 is the value of the second moment of *AVD* with the normalized co-occurrence matrix proposed by Cardoso; Braga (2014), being $AVD2 = E[|i - j|^2]$. This is similar to the inertia moment technique (ARIZAGA; TRIVI; RABAL, 1999) with the Cardoso's normalization.

AVD3 is the second central moment of *AVD* (variance) with the normalized co-occurrence matrix proposed by Cardoso; Braga (2014), being $AVD3 = E[|i - j|^2] - AVD1^2$.

AVD4 is the value of *AVD* with the normalized co-occurrence matrix proposed by Arizaga; Trivi; Rabal (1999).

Only AVD1 is an obligatory output parameter of function `avd()`, while the others are enabled by using of optional input parameters "2,3,4". This indicates that the function should also return the values: AVD2 for "2", AVD3 for "3" and AVD4 for "4".

All *AVD* outcomes in the previous code example are calculated from a co-occurrence matrix *COM*, which in turn is calculated from the time history speckle pattern (THSP) of the line number 240 of datapack *DATA* that was formed by images from the directory pointed by the *IMAGESDIR* variable. Additionally, examples using the functions `datapack()`, `thsp()` and `coom()` can be found in the Sections 4.1, 4.2 and 4.3 respectively. The code will return the next information.

```
Loading images from:~/data/cafe-biospeckle/sem1
Please wait...
DATA Pack loaded... [OK]
AVD1 = 13.764
AVD2 = 358.57
AVD3 = 169.12
AVD4 = 3208.5
```

So that $AVD1 = E[|i - j|] = 13.764$ represents the first moment of probability mass function $Pr(|i - j| = z)$ and $\sqrt{AVD3} = \sqrt{E[|i - j|^2] - AVD1^2} = 13.005$ represents the second central moment of probability mass function $Pr(|i - j| = z)$. The next code generates the Figure 4.8 which represents the relations between the *AVD* values with the probability mass function $Pr(|i - j| = z)$ returned by `pmfad()`, presented in the Section 4.4.1.

```
[Pr Z]=pmfad(COM);

figure(1)
plot(      Z,Pr,'--o', ...
      AVD1*ones(1,17),[0:16]*0.085/16,'--s', ...
      AVD1-sqrt(AVD3)*ones(1,17),[0:16]*0.065/16,'r--<', ...
      AVD1+sqrt(AVD3)*ones(1,17),[0:16]*0.065/16,'r-->'),grid
hx=xlabel('|i-j|=z');
hy=ylabel('Probability');
ht=title('Probability Mass Function');
hl=legend('pmfad()', 'AVD1', 'AVD1-sqrt(AVD3)', 'AVD1+sqrt(AVD3)');
xlim([0 40]);
```

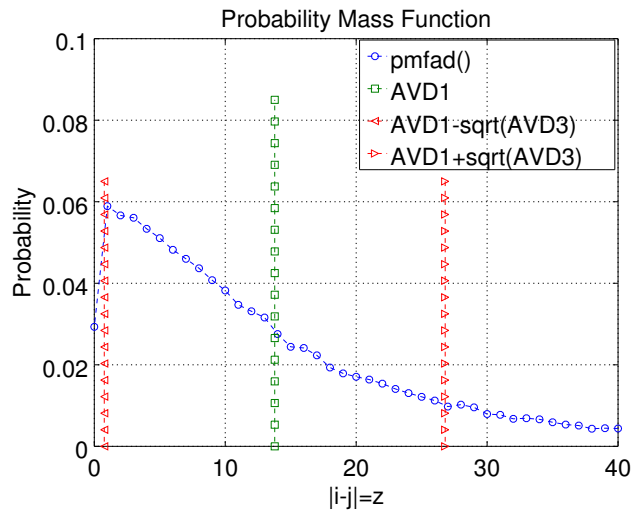


Figure 4.8: Graphic of relations between $AVD1$, $\sqrt{AVD3}$ and `pmfad()` function.

4.5.3 Regular Value of the Differences

The next code shows many ways to calculate the Regular Value of the Differences (*RVD*) with the normalization of the co-occurrence matrix proposed by Cardoso; Braga (2014) and Arizaga; Trivi; Rabal (1999).

```
DATA=datapack(IMAGESDIR, '', 1, 128, 'bmp');
THSP=thsp(DATA, 1, 240);
COM=coom(THSP);

[RVD1 RVD2 RVD3 RVD4]=rvd(COM, 2, 3, 4)
```

Where

RVD1 is the value of *RVD* with the normalized co-occurrence matrix proposed by Cardoso; Braga (2014), being $RVD1 = E[j - i]$.

RVD2 is the value of the second moment of *RVD* with the normalized co-occurrence matrix proposed by Cardoso; Braga (2014), being $RVD2 = E[(j - i)^2]$. This is similar to inertia moment technique (ARIZAGA; TRIVI; RABAL, 1999) with the Cardoso's normalization.

RVD3 is the second central moment of *RVD* (variance), with the normalized co-occurrence matrix proposed by Cardoso; Braga (2014), being $RVD3 = E[(j - i)^2] - RVD1^2$.

RVD4 is the value of *RVD* with the normalized co-occurrence matrix proposed by Arizaga; Trivi; Rabal (1999).

Only RVD1 is an obligatory output parameter of function `rvd()`, while the others are enabled by the use of the optional input parameters "2,3,4". This indicates that the function should also return the values: RVD2 for "2", RVD3 for "3" and RVD4 for "4".

All *RVD* outcomes in the previous code are calculated from a co-occurrence matrix *COM*, which in turn is calculated from the time history speckle pattern (THSP) of the line number 240 in the datapack *DATA* which was formed by images from the directory pointed by the *IMAGESDIR* variable. Additionally, examples using the functions `datapack()`, `thsp()` and `coom()` can be found in the Sections 4.1, 4.2 and 4.3 respectively. The code will return the next information.

```
Loading images from:~/data/cafe-biospeckle/sem1
Please wait...
DATA Pack loaded...[OK]
RVD1 = -0.0032340
RVD2 = 359.36
RVD3 = 359.36
RVD4 = -1267.6
```

So that $RVD1 = E[j - i] = 0.0095912$ represents the first moment of probability mass function $Pr((j - i) = w)$ and $\sqrt{RVD3} = \sqrt{E[(j - i)^2] - RVD1^2} = 18.936$ represents the second central moment of probability mass function $Pr((j - i) = w)$. The next code generates the Figure 4.9, which represents the relations between the *RVD* values with the probability mass function $Pr((j - i) = w)$ returned by the `pmfrd()` function, as presented in the Section 4.4.2.

```
Pr=pmfrd(COM);
N=length(Pr);
W=[0:N-1]-round((N-1)/2);

figure(1)
plot(W,Pr,'--o', ...
      RVD1*ones(1,17),[0:16]*0.05/16,'--s', ...
      RVD1-sqrt(RVD3)*ones(1,17),[0:16]*0.03/16,'r--<', ...
      RVD1+sqrt(RVD3)*ones(1,17),[0:16]*0.03/16,'r-->'),grid
hx=xlabel('(j-i)=w');
```

```

hy=ylabel('Probability');
ht=title('Probability Mass Function');
hl=legend('pmfrd()', 'RVD1', 'RVD1-sqrt(RVD3)', 'RVD1+sqrt(RVD3)');
xlim([-40 40]);

```

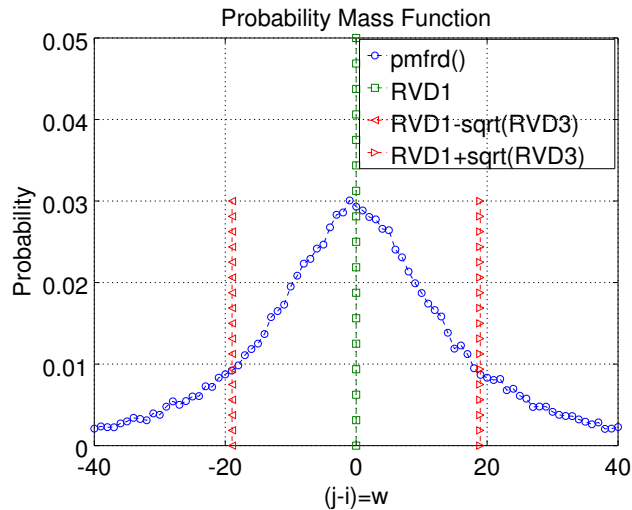


Figure 4.9: Graphic of relations between RVD1, $\sqrt{\text{RVD3}}$ and `pmfrd()` function.

4.5.4 Numerical Analysis of the Average Differences

The next code shows how to make a Numerical Analysis of the Average Differences (*NUMAD*) (REIS; RABAL; BRAGA, 2016) of the co-occurrence matrix, in order to reduce the influence of the illumination in the numerical outcomes.

```

DATA=datapack(IMGESDIR, '', 1, 128, 'bmp');
THSP=thsp(DATA, 1, 240);
COM=coom(THSP);
[NUMAD1 NUMAD2]=numad(COM, 2)

```

Where,

NUMAD1 is the first moment of *NUMAD* using the normalized co-occurrence matrix proposed by Cardoso; Braga (2014), where $\text{NUMAD1} = E[|i - j| / (i + j)]$.

NUMAD2 is the second moment of *NUMAD* using the normalized co-occurrence matrix proposed by Cardoso; Braga (2014), where $\text{NUMAD2} = E[|i - j|^2 / (i + j)^2]$.

Only NUMAD1 is an obligatory output parameter of the function `numad()`, while the other is enabled by using of optional input parameter “2”. This indicates that the function should also return the value NUMAD2.

All *NUMAD* outcomes in the previous code example are calculated from a co-occurrence matrix COM, which in turn is calculated from the time history speckle pattern (THSP) of the line number 240 in the datapack DATA, which was formed by images from the directory pointed by the IMGESDIR variable. Additionally, examples using the functions `datapack()`, `thsp()` and `coom()` can be found in the Sections 4.1, 4.2 and 4.3 respectively. The code will return the next information.

```

Loading images from:~/data/cafe-biospeckle/sem1

Please wait...
DATA Pack loaded...[OK]
NUMAD1 = 0.084686
NUMAD2 = 0.013135

```

So that $NUMAD1 = E[|i - j|/(i + j)] = 0.084686$ represents the first moment of *NUMAD* and $NUMAD2 = E[|i - j|^2/(i + j)^2] = 0.013135$ represents the second moment of *NUMAD*.

4.5.5 Time History Speckle Pattern to Correlation Coefficient

The next code shows the use of `thsp2corr()` function, which implements a method to analyze the correlation between pixels in the time history speckle pattern (XU; JOENATHAN; KHORANA, 1995).

```

DATA=datapack(IMAGESDIR, ' ', 1, 128, 'bmp');

THSP=thsp(DATA, 1, 240);

[C1 L1] = thsp2corr(THSP, 1);
[C2 L2] = thsp2corr(THSP, 2);

```

The vectors C1 and C2 have the correlation values of pixel inside of the THSP matrix of the line number 240 in the datapack DATA formed by images of the directory pointed by the IMAGESDIR variable. The difference between C1 and C2 lies in that C1 use the correlation proposed by Xu; Joenathan; Khorana (1995) and C2 uses the Pearson correlation (ZDUNEK et al., 2007). L1 and L2 represent the sample lag for the correlations C1 and C2 respectively.

The next code generates the Figure 4.10, which shows the two types of correlations returned by the `thsp2corr()` function, where it is possible to see the similarity between both results. It is also possible to note that the pixel informations slightly tends to be repeated every 11 samples.

```

[ax, p1, p2] = plotyy(L1, C1, L2, C2);

hx = xlabel('L1 and L2');
hy1 = ylabel(ax(1), 'C1');
hy2 = ylabel(ax(2), 'C2');
set(p1, 'Marker', 's');
set(p2, 'Marker', 'o');
legend([p1, p2], {'Pearson corr.', 'Xu et al. corr.'}, ...
       'Location', 'northoutside', ...
       'Orientation', 'horizontal');

```

Examples using the functions `datapack()` and `thsp()` can be found in the Sections 4.1 and 4.2 respectively.

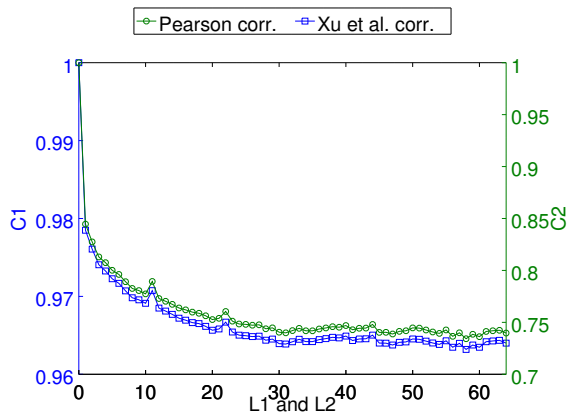


Figure 4.10: Time history speckle pattern to correlation coefficient.

4.5.6 Spatial-Temporal Speckle Correlation

The next code shows the use of the `stscorr()` function, which implements a method to analyze the spatial-temporal speckle correlation between matrices (ZDUNEK et al., 2007) .

```
DATA=datapack(IMAGESDIR, '', 1, 128, 'bmp');

tau=1;
[C T]=stscorr(DATA, tau, 42);
```

The C vector has the values of spatial-temporal speckle correlation between the images inside of the datapack DATA, which was formed by the images in the directory pointed by the IMAGESDIR variable. In this case the matrix (image) in the position 42 of the datapack (`DATA(:, :, 42)`) is correlated with all other matrices, so that in each correlation one element of the vector C is filled with the Pearson correlation coefficient. The `stscorr()` function considers that the images were obtained in intervals of $\tau = 1$ second. Thus the vector T is returned with these time intervals.

The Figure 4.11 shows the image returned by the `stscorr()` function, where it is possible to see a graphic of the C vector versus the T vector.

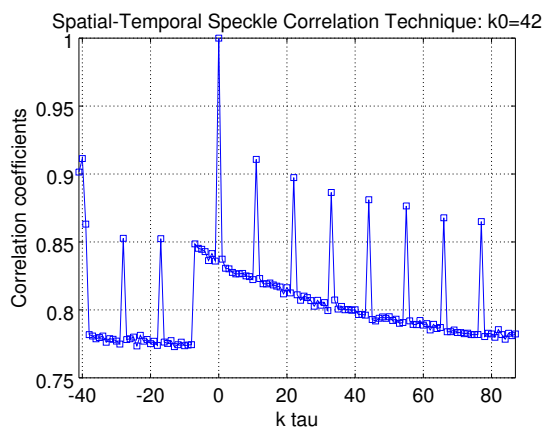


Figure 4.11: Spatial-temporal speckle correlation.

Examples using the `datapack()` function can be found in the Section 4.1.

4.6 Graphic Methods

4.6.1 Average Difference or Fujii Method

The next code shows the use of the `fujii()` function, which implements the graphic activity indicator, or map of activities, called Fujii technique (FUJII et al., 1987; FUJII; ASAKURA, 1975).

```
DATA=datapack(IMAGESDIR, '', 1, 128, 'bmp');
FUJII=fujii(DATA);
```

The FUJII matrix is the result of the Fujii technique application over a datapack DATA formed by 128 images in the directory pointed by the IMAGESDIR variable. Additionally, examples using the `datapack()` function can be found in the Section 4.1.

The Figure 4.12 shows the outcome of the FUJII matrix.

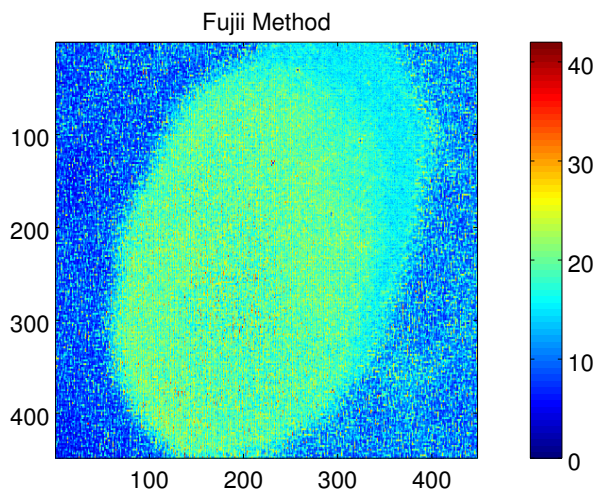


Figure 4.12: Map of activities obtained by the Fujii method on a collection of images of a coffee seed.

4.6.2 Generalized Difference Method - GD

The next code shows the use of the `gendiff()` function, which implements the graphic activity indicator called generalized difference method (ARIZAGA et al., 2002).

```
DATA=datapack(IMAGESDIR, '', 1, 128, 'bmp');
GENDIFF=gendiff(DATA);
```

The GENDIFF matrix is the result of the generalized difference method application on a datapack DATA formed by 128 images in the directory pointed by IMAGESDIR variable. Additionally, examples using the `datapack()` function can be found in the Section 4.1.

The `gendiff()` function has a considerable computational cost, thus it has a verbose output. Next we can see the outcome of `gendiff()`.

```
Loading images from:~/data/cafe-biospeckle/sem1
Please wait...
DATA Pack loaded...[OK]
```



```

Start G.D. method ...
10.156 %
20.312 %
30.469 %
40.625 %
50.781 %
60.938 %
71.094 %
81.25 %
91.406 %
[OK]

```

The Figure 4.13 shows the outcome of GENDIFF matrix of a collection of images of a coffee seed.

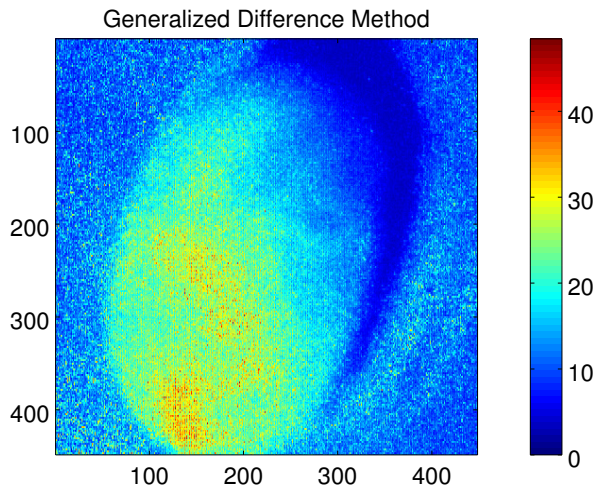


Figure 4.13: Map of activities obtained by the GD method on a collection of images of a coffee seed.

4.6.3 Temporal Speckle Contrast - Standard Deviation - Mean's Method

The next code shows the use of `stdcont()` function, which implements temporal speckle contrast matrix (NOTHDURFT; YAO, 2005). Additionally, the function returns the temporal speckle standard deviation matrix and temporal speckle mean matrix.

```

DATA=datapack(IMAGESDIR, '', 1, 128, 'bmp');

[C D E]=stdcont(DATA);

```

The C matrix is the temporal speckle contrast matrix created from a datapack DATA that was formed by 128 images in the directory pointed by the IMAGESDIR variable. Additionally, the function returns the matrices D and E which are the temporal speckle standard deviation matrix and temporal speckle mean matrix respectively. Examples using the `datapack()` function can be found in the Section 4.1.

The Figure 4.14 shows the outcome of matrices C, D and E.

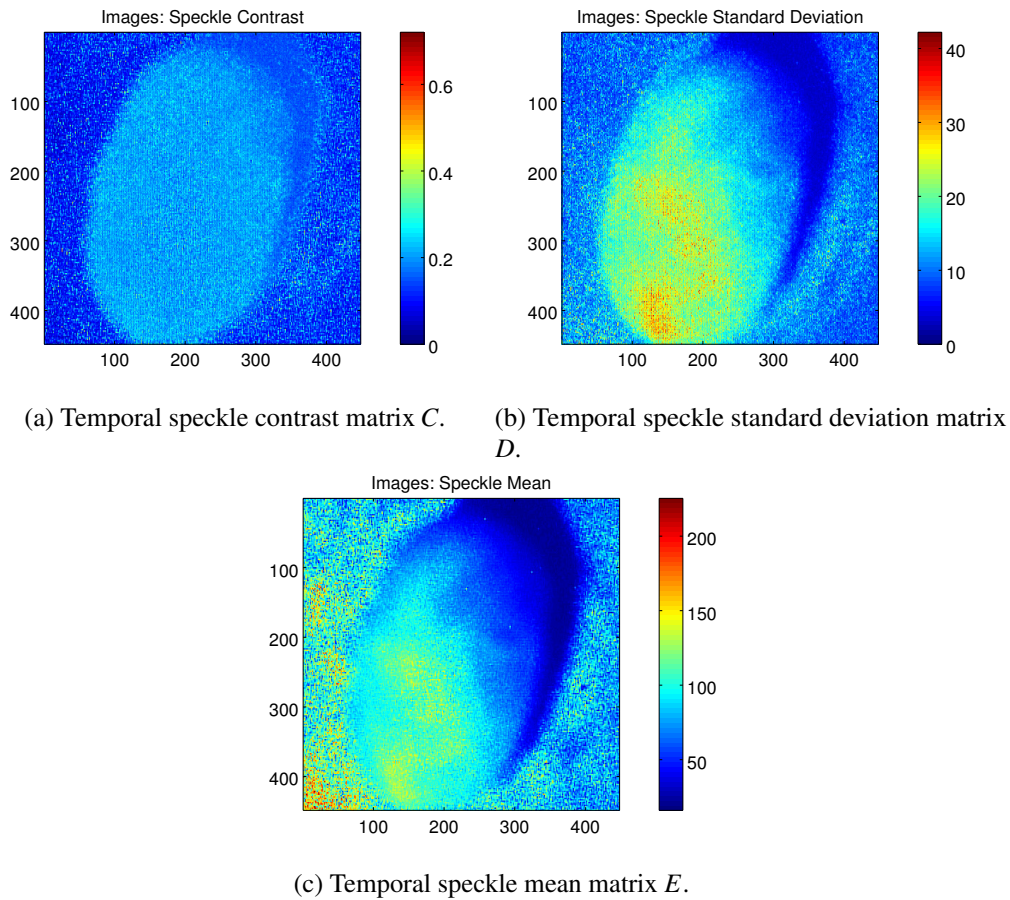


Figure 4.14: Matrices returned by `stdcont()` function, with (a) representing the Temporal speckle contrast matrix, named C , (b) representing the Temporal speckle standard deviation matrix, named D , and (c) representing the Temporal speckle mean matrix, named E .

4.6.4 Inertia Moment Method in Graphic Mode

The next code shows the use of `graphim()` function, which implements the inertia moment activity indicator (ARIZAGA; TRIVI; RABAL, 1999) for each pixel of the datapack, with the normalization of co-occurrence matrix proposed by Cardoso; Braga (2014).

```
DATA=datapack(IMAGESDIR, '', 1, 128, 'bmp');
GIM=graphim(DATA);
```

The GIM matrix is the result of applying the inertia moment technique on a datapack `DATA` that was formed by 128 images in the directory pointed by the `IMAGESDIR` variable. Additionally, examples using the `datapack()` function can be found in the Section 4.1.

The Figure 4.15 shows the outcome of GIM matrix of a coffee seed.

4.6.5 Absolute Value of the Differences Method in Graphic Mode

The next code shows the use of `graphavd()` function, which implements the *AVD* activity indicator (BRAGA et al., 2011) for each pixel of the datapack, with the normalization of co-occurrence

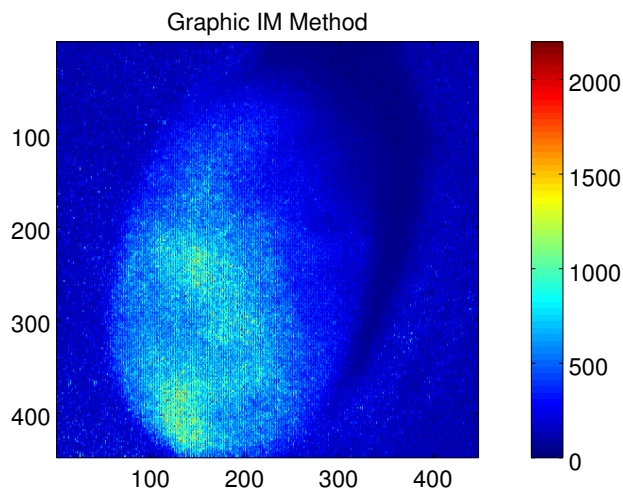


Figure 4.15: Graphic inertia moment method of a coffee seed.

matrix proposed by Cardoso; Braga (2014).

```
DATA=datapack(IMAGESEDIR, '', 1, 128, 'bmp');
GAVD=graphavd(DATA);
```

The GAVD matrix is the result of applying the *AVD* technique on a datapack *DATA* that was formed by 128 images in the directory pointed by the *IMAGESEDIR* variable. Additionally, examples using the `datapack()` function can be found in the Section 4.1.

The Figure 4.16 shows the outcome of GAVD matrix of a coffee seed.

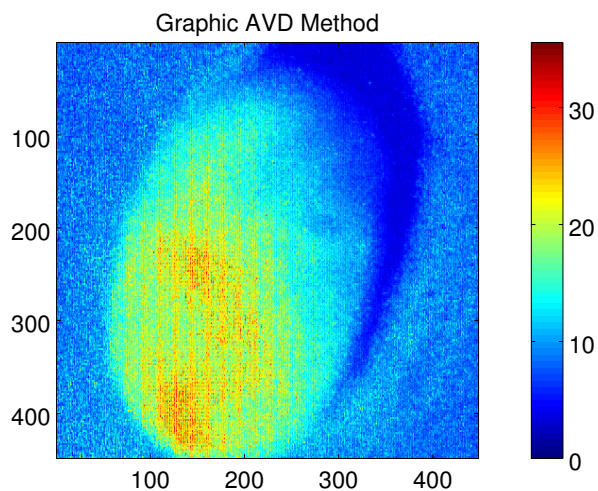


Figure 4.16: Graphic *AVD* method of a coffee seed.

4.6.6 Regular Value of the Differences Method in Graphic Mode

The next code shows the use of `graphrvd()` function, which implements the *RVD* indicator can be seen in Section 4.5.3 for each pixel of the datapack, with the normalization of co-occurrence matrix proposed by Cardoso; Braga (2014).

```
DATA=datapack(IMAGESEDIR, '', 1, 128, 'bmp');
```

```
GRVD=graphrvd(DATA);
```

The GRVD matrix is the result of applying the *RVD* technique on a datapack DATA that was formed by 128 images in the directory pointed by the IMAGESDIR variable. Additionally, examples using the `datapack()` function can be found in the Section 4.1.

The Figure 4.17 shows the outcome of GRVD matrix applied in a coffee seed.

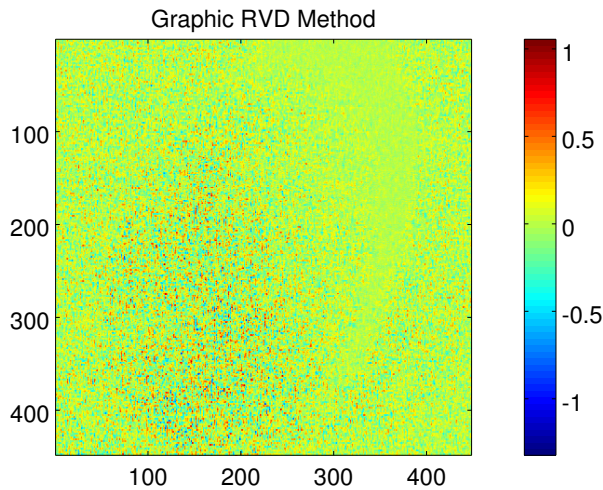


Figure 4.17: Graphic *RVD* method of a coffee seed.

4.6.7 Parametrized form of Temporal Difference Method

The next code shows the use of `graphptd()` function, which implements the graphic activity indicator called in the literature as *PTD* technique (MINZ; NIRALA, 2014).

```
DATA=datapack(IMAGESDIR, '', 1, 128, 'bmp');
GPTD=graphptd(DATA, 0.5);
```

The GPTD matrix is the result of the *PTD* technique over a datapack DATA that was formed by 128 images in the directory pointed by the IMAGESDIR variable. Additionally, examples using the `datapack()` function can be found in the Section 4.1.

The Figure 4.18 shows the outcome of the GPTD matrix applied in a coffee seed.

4.6.8 Temporal Speckle Kurtosis Matrix

The next code shows the use of `graphkurt()` function, which implements the kurtosis (SHESKIN, 2003) for each pixel of the datapack.

```
DATA=datapack(IMAGESDIR, '', 1, 128, 'bmp');

LIN=240;    LIN_WIDE=64;
COL=320;    COL_WIDE=64;
```

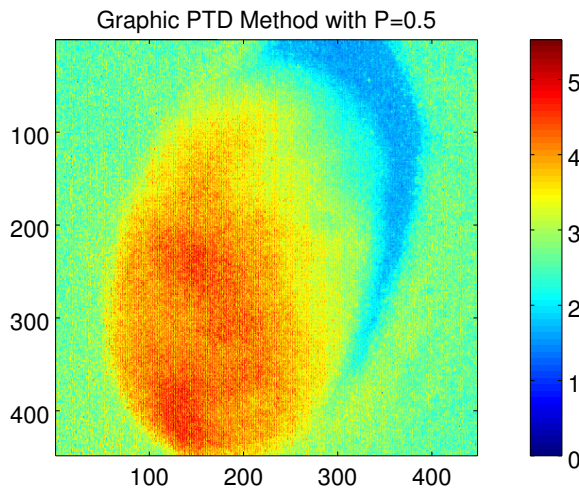



Figure 4.18: Parameterized form of Temporal Difference method.

```
PORTIONDATA=DATA(LIN+[0:LIN_WIDE-1],COL+[0:COL_WIDE-1],:);

[K D E]=graphkurt(PORTIONDATA);

MEAN=mean2(K)
STDDEV=std2(K)
```

The K matrix is the kurtosis of a datapack PORTIONDATA, which, in turn, is a portion of the datapack DATA that was formed by 128 images in the directory pointed by the IMAGESDIR variable. In this case, PORTIONDATA is a portion of 64×64 pixels from the position $(line, column) = (240, 320)$. In the last code, the `graphkurt()` function also returns the temporal speckle standard deviation matrix D and the temporal speckle mean matrix E (NOTHDURFT; YAO, 2005). Additionally, the spatial mean (MEAN) and the spatial standard deviation (STDDEV) of K matrix are calculated. See the results of the code presented above:

```
Loading images from:~/data/caf -biospeckle/sem1
Please wait...
DATA Pack loaded...[OK]
MEAN = 3.2152
STDDEV = 0.77433
```

Here, it is interesting to note that the kurtosis value of Gaussian random variable is 3.0. The last result shows that the kurtosis of random variables in PORTIONPACK is closed to a Gaussian type. The Figure 4.19 shows the content of K matrix in a graphic mode applied in a coffee seed.

Examples using the `datapack()` function can be found in the Section 4.1.

4.6.9 Temporal Speckle Skewness Matrix

The next code shows the use of `graphskew()` function, which implements the skewness (SHESKIN, 2003) for each pixel of the datapack.

```
DATA=datapack(IMAGESDIR, ' ', 1, 128, 'bmp');
```

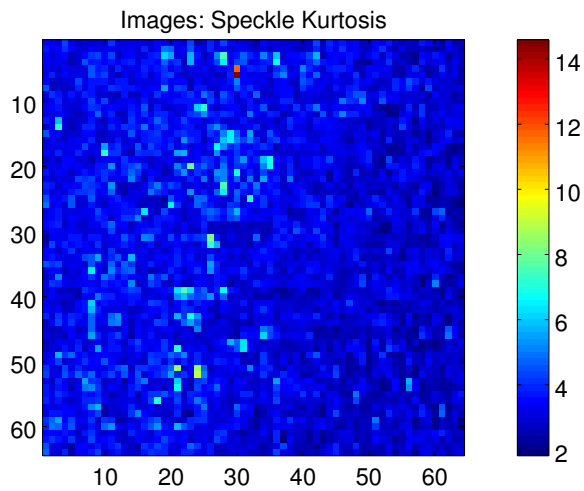


Figure 4.19: Temporal speckle kurtosis matrix of a coffee seed

```

LIN=240;   LIN_WIDE=64;
COL=320;   COL_WIDE=64;

PORTIONDATA=DATA(LIN+[0:LIN_WIDE-1],COL+[0:COL_WIDE-1],:);

[S D E]=graphskew(PORTIONDATA);

MEAN=mean2(S)
STDDEV=std2(S)

```

The S matrix is the skewness of a datapack `PORTIONDATA`, which, in turn, is a portion of the datapack `DATA` that was formed by 128 images in the directory pointed by the `IMAGESDIR` variable. In this case, `PORTIONDATA` is a portion of 64×64 pixels from the position $(line, column) = (240, 320)$. The `graphskew()` function also returns the temporal speckle standard deviation matrix D and the temporal speckle mean matrix E (NOTHDURFT; YAO, 2005). The last code also calculates the spatial mean (`MEAN`) and the spatial standard deviation (`STDDEV`) of S matrix, with the outcome presented highlighted:

```

Loading images from:~/data/cafe-biospeckle/sem1
Please wait...
DATA Pack loaded...[OK]
MEAN = 0.15805
STDDEV = 0.37181

```

Here, it is interesting to note that the skewness value of the Gaussian random variable is 0.0. The last result shows that the skewness of random variables in `PORTIONPACK` is closed to a Gaussian type. The Figure 4.20 shows the content of S matrix in a graphic mode applied in a coffee seed and the examples using the `datapack()` function can be found in the Section 4.1.

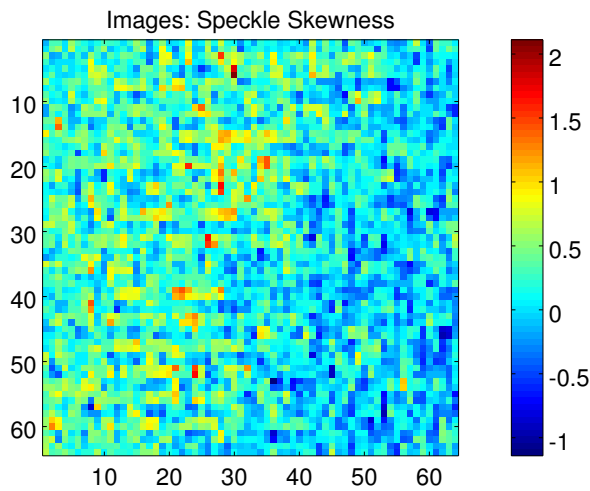


Figure 4.20: Temporal speckle skewness matrix of a coffee seed.

4.6.10 Motion History Image

The next code shows the use of `graphmhi()` function, which implements the motion history image technique (DAVIS, 2001).

```
DATA=datapack(IMAGESDIR, '', 1, 128, 'bmp');
MHI=graphmhi(DATA, 11);
```

The MHI matrix is the result of applying the motion history image technique on a datapack DATA that was formed by 128 images in the directory pointed by the IMAGESDIR variable. In this case, a threshold of 11 is used, and it means that only intensity values larger than 11 are considered activity changes. The Figure 4.21 shows the content of the MHI matrix. Examples using the `datapack()`

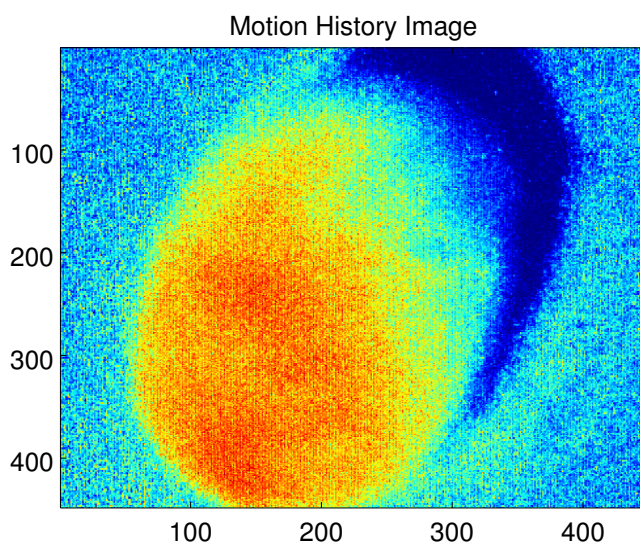


Figure 4.21: Motion history image.

function can be found in the Section 4.1.

4.7 Quality Tests

4.7.1 Analyzing the Saturation and the Sub-exposition of Light

The next code shows the use of `satdark()` function, that implements a method for the analysis of saturation and sub-exposition in an image (CARDOSO; BRAGA; RABAL, 2012).

```
DATA=datapack(IMAGESDIR, '', 1, 128, 'bmp');
IMAGE1=DATA(:, :, 1);

[Img S D]=satdark(IMAGE1, 8, 8, 20, 150, 50);
```

The `Img` matrix is an image with dark and saturated areas manually introduced in some regions of the speckle pattern, in this case over the first image `IMAGE1` of the datapack `DATA` of a coffee seed. The datapack was formed by the images in the directory pointed by the `IMAGESDIR` variable. In this case, windows with 8×8 pixels were used to analyze the saturation and the sub-exposition. The `satdark()` function considers that the windows are saturated when more than 50 percent of pixels are greater than the gray level 150. On the other hand, the function considers the windows in darkness or sub-exposed to laser light when more than 50 percent of pixels are lower than the gray level 20.

Additionally, the function returns the `S` and `D` matrix, which are matrices with values “0” or “1”. The `S` matrix has values “1” in the pixels with saturation, and The `D` matrix has values “1” in the pixels with darkness, in accordance with the levels offered as input. The Figures 4.22, 4.23a and 4.23b show the content of matrices `Img`, `S` and `D`, respectively.

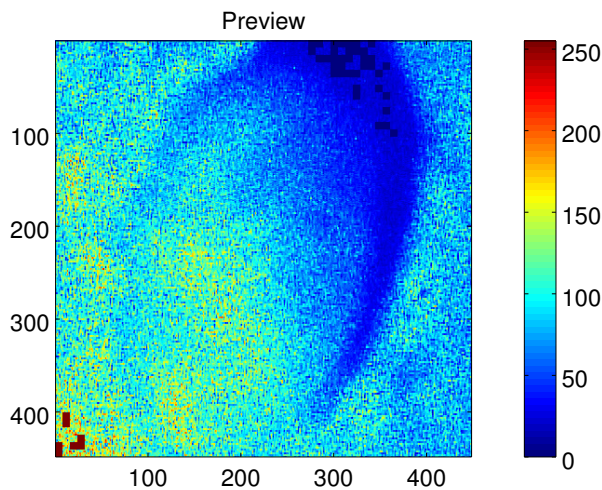


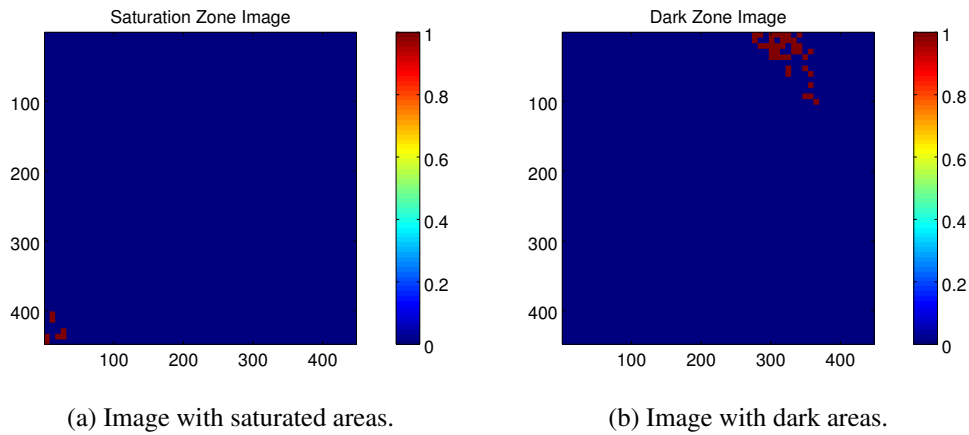
Figure 4.22: Image with dark or saturated areas of a coffee seed, in this case introduced manually.

Examples using the `datapack()` function can be found in the Section 4.1.

4.7.2 Spatial Speckle Contrast Window Method

The next code shows the use of `sscont()` function, that implements the spatial speckle contrast window method (CARDOSO; BRAGA; RABAL, 2012).

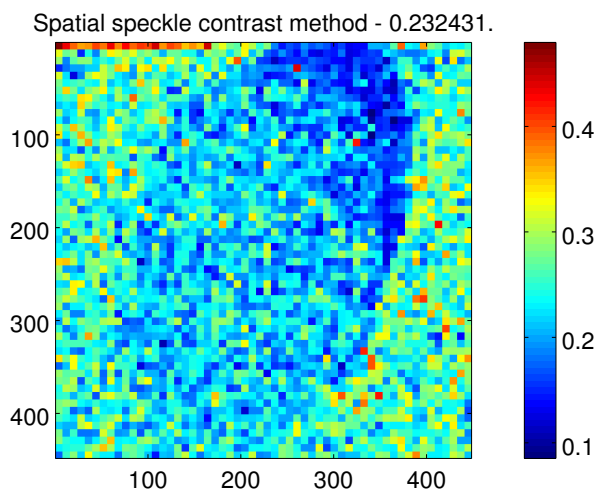
```
DATA=datapack(IMAGESDIR, '', 1, 128, 'bmp');
```


Figure 4.23: Matrices showed by the `satdark()` function.

```
IMAGE1=DATA(:, :, 1);

[C mC]=sscont(IMAGE1,8,8);
```

The C matrix is the windowed spatial speckle contrast (CARDOSO; BRAGA; RABAL, 2012) over the first image IMAGE1 of the datapack DATA. The datapack was formed by the images in the directory pointed by the IMAGESDIR variable. In this case, windows with 8x8 pixels were used. In each window, the spatial speckle contrast is calculated within the C matrix. This value indicates the relation between the spatial standard deviation and the spatial mean value. Additionally, the function returns the mC value, that is the mean value of the contrasts in all windows in C. The Figure 4.24 shows the content of C matrix.

Figure 4.24: Windowed spatial speckle contrast image, with $mC = 0.232431$ of a coffee seed.

Examples using the `datapack()` function can be found in the Section 4.1.

4.7.3 Homogeneity of Spatial Variability

The next code shows the use of `homogeneity()` function, which implements the homogeneity of spatial variability of a measured activity (CARDOSO; BRAGA; RABAL, 2012).

```
DATA=datapack(IMGESDIR, '', 1, 128, 'bmp');

[Y X]=homogeneity(DATA, 8, 8, 0);
```

The Y matrix represents the homogeneity (CARDOSO; BRAGA; RABAL, 2012) percentages in the windows of a datapack DATA that was formed by 128 images in the directory pointed by the IMGESDIR variable. In this case, windows of 8×8 pixels were created and the inertia moment method (ARIZAGA; TRIVI; RABAL, 1999) was applied in each one (Type 0). Additionally, the function returns the X matrix, where each element of X represents the spatial mean value of inertia moment in the windows of Y.

The `homogeneity()` function, in the last code, returns a graphical outcome with the inertia moment activity indicator (see Figure 4.25a) and the homogeneity Y matrix (see Figure 4.25b).

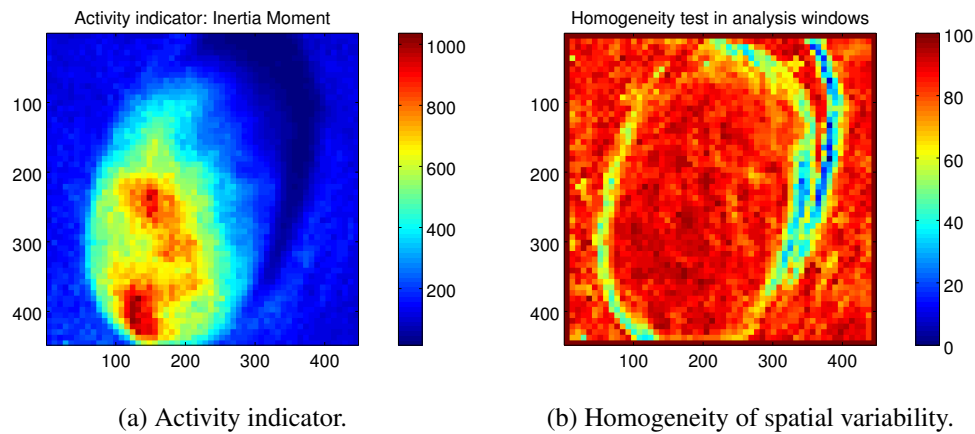


Figure 4.25: Matrices showed by the `homogeneity()` function.

Examples using the `datapack()` function can be found in the Section 4.1.

4.8 Frequency Analysis

4.8.1 Filtering in Frequency Bands

The next code shows the use of `firfilterbank()` function in complementary filter mode ('mode0'). Thereby the `firfilterbank()` function implements a filter bank step with one input and two outputs, so that the summation of outputs is equal to the input.

```
DATA=datapack(IMGESDIR, '', 1, 128, 'bmp');

ORDER=32;

HO = get_fir_filter(ORDER, 1/2, 'low'); % Cut-off at 1/2 of Fs/2.
GO = get_fir_filter(ORDER, 1/4, 'low'); % Cut-off at 1/4 of Fs/2.
WO = get_fir_filter(ORDER, 1/8, 'low'); % Cut-off at 1/8 of Fs/2.

[DATA0 DATA1] = firfilterbank(DATA, HO, 'mode0');
```

```
[DATA00 DATA01] = firfilterbank(DATA0,G0,'mode0');
[DATA000 DATA001] = firfilterbank(DATA00,W0,'mode0');
```

The code implements a filtering scheme in frequency bands, the same as shown in the Figure 4.26, where a datapack DATA that was formed by 128 images in the directory pointed by the IMAGESDIR variable is separated in datapacks, { DATA1, DATA01, DATA001, DATA000 } with signals, each one pertaining to a specific frequency band. In the first step we used a low pass H0 *FIR* filter, with

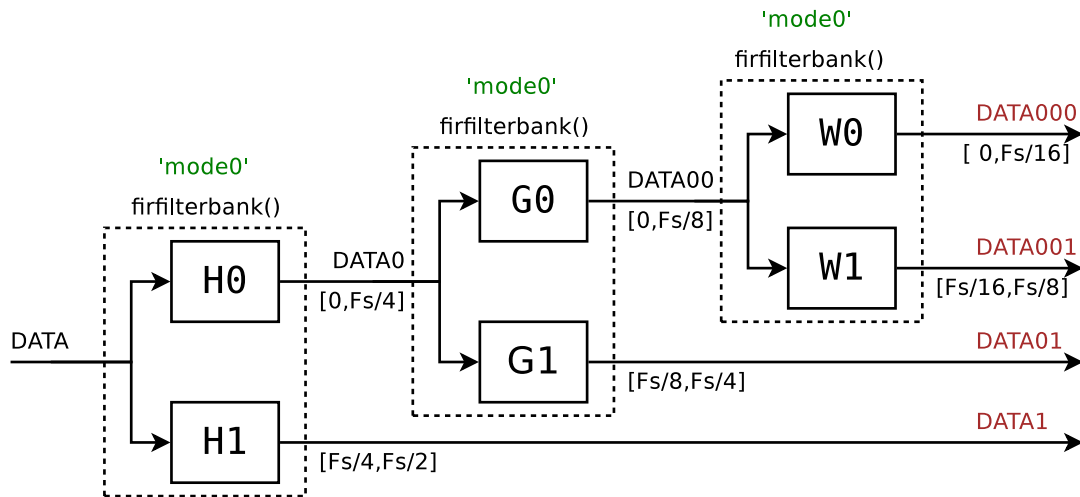


Figure 4.26: Scheme of filtering in frequency bands.

cut-off at $1/2$ of $F_s/2$ (half the sampling frequency), and the signal in the datapack DATA is divided into the datapacks DATA0 and DATA1, with the frequency bands $[0, F_s/4]$ and $[F_s/4, F_s/2]$ respectively. In the second step, using a low pass G0 *FIR* filter, with cut-off at $1/4$ of $F_s/2$, the signal in the datapack DATA0 is divided into the datapacks DATA00 and DATA01 with the frequency bands $[0, F_s/8]$ and $[F_s/8, F_s/4]$ respectively. And finally, similarly to the last cases a low pass W0 *FIR* filter with cut-off at $1/8$ of $F_s/2$ is used to separate DATA00 in DATA000 and DATA001, with the frequency bands $[0, F_s/16]$ and $[F_s/16, F_s/8]$ respectively.

At this point, we have a set of datapacks separated in frequency bands, in which we can apply any of the biospeckle analysis methods seen in the last sections. For example, the next code uses the *AVD* method in graphic mode on the datapacks { DATA1, DATA01, DATA001, DATA000 }.

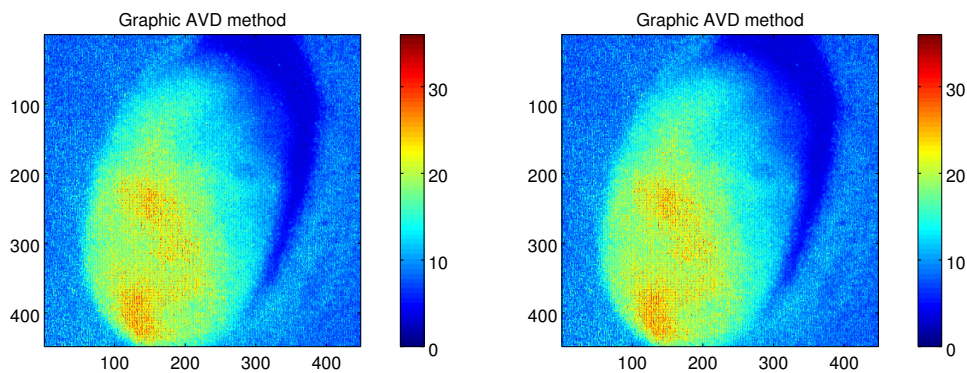
```
GAVD =graphavd(DATA,'off');
GAVD_ =graphavd(DATA1+DATA01+DATA001+DATA000,'off');

GAVD1 =graphavd(DATA1,'off');
GAVD01 =graphavd(DATA01,'off');
GAVD001=graphavd(DATA001,'off');
GAVD000=graphavd(DATA000,'off');
```

The next code generates the Figure 4.27, which shows the results of comparing the matrices GAVD and GAVD_, that means the result of the *AVD* graphic activity indicator over the datapack DATA and its homologue evaluated from a datapack formed by summing DATA1+ DATA01+ DATA001+ DATA000.

```
figure(1);
imagesc(GAVD,[0 36]);colorbar;
title('Graphic AVD method');

figure(2);
imagesc(GAVD_,[0 36]);colorbar;
title('Graphic AVD method');
```



(a) *AVD* method in graphic mode of a datapack DATA.

(b) *AVD* method in graphic mode of a datapack formed by summing DATA1+ DATA01+ DATA001+ DATA000.

Figure 4.27: Comparison between the *AVD* graphic activity indicator of the datapack DATA and its homologue evaluated from a datapack formed by the reconstructed signal.

Finally, the next code generates the Figure 4.28, which shows the result comparing the matrices GAVD1, GAVD01, GAVD001 and GAVD000 with the same color map. The matrices are the results of the *AVD* graphic activity indicator on the datapacks DATA1, DATA01, DATA001 and DATA000, respectively.

```
figure(3);
imagesc(GAVD1,[0 36]);colorbar;
title('Graphic AVD method');

figure(4);
imagesc(GAVD01,[0 36]);colorbar;
title('Graphic AVD method');

figure(5);
imagesc(GAVD001,[0 36]);colorbar;
title('Graphic AVD method');

figure(6);
imagesc(GAVD000,[0 36]);colorbar;
title('Graphic AVD method');
```

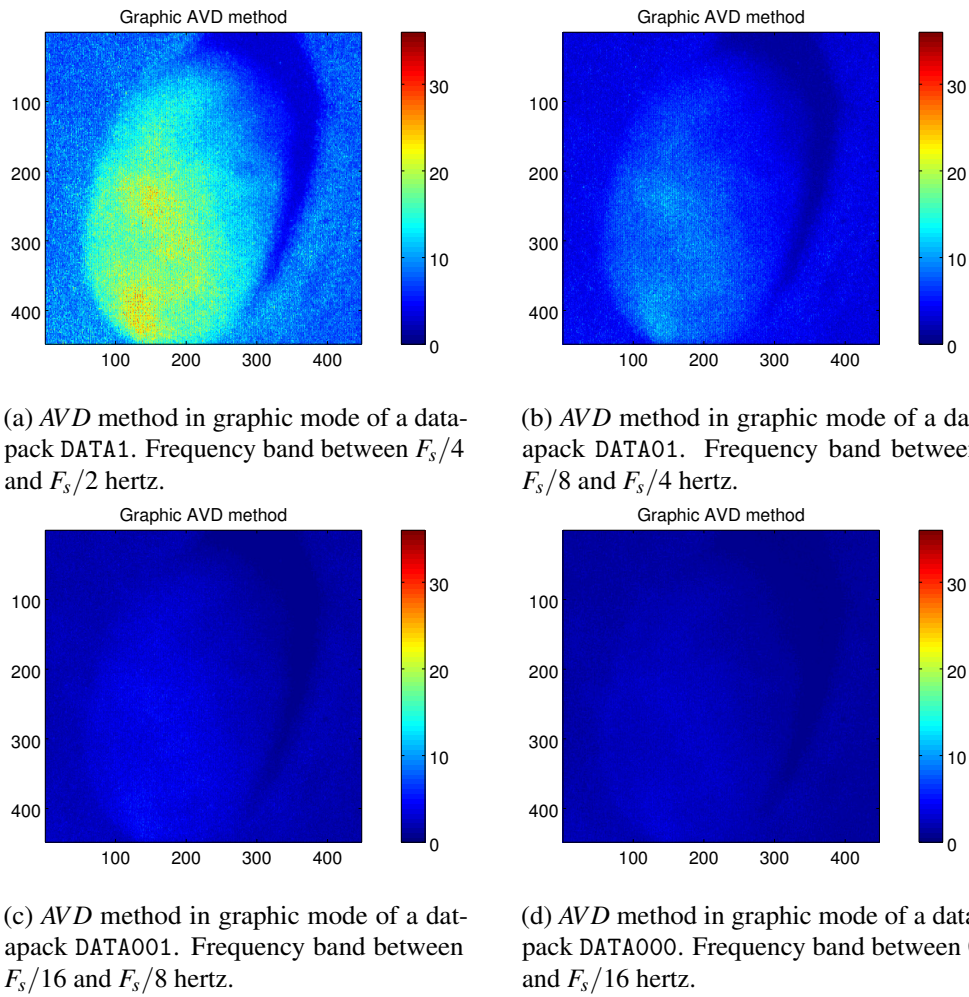


Figure 4.28: Comparison between the AVD graphic activity indicator of datapacks in different frequency bands.

4.8.2 Decomposition Using the Discrete Wavelet Transform

The next code shows the use of `firfilterbank()` function in quadrature mirror filter ('mode2'), that implements a discrete wavelet transform step with one input and two outputs, so that if the input signal has L samples the outputs have $\lceil L/2 \rceil$ samples respectively, and these contain the information of half part of the maximum possible input bandwidth. Additionally we presented the `qmfmaker()` function, which implements a *FIR* filter with cut-off frequency at $F_s/4$, so that it fulfills the requirements with the Quadrature Mirror Filter (*QMF*) providing perfect reconstruction.

```
DATA=datapack(IMAGESDIR, ' ', 1, 128, 'bmp');

ORDER=33;
HO=qmfmaker(ORDER);

[DATA0 DATA1] = firfilterbank(DATA, HO, 'mode2');
[DATA10 DATA11] = firfilterbank(DATA1, HO, 'mode2');
[DATA00 DATA01] = firfilterbank(DATA0, HO, 'mode2');
```

The code implements a decomposition scheme in coefficients with information of frequency bands, same as shown in the Figure 4.29.

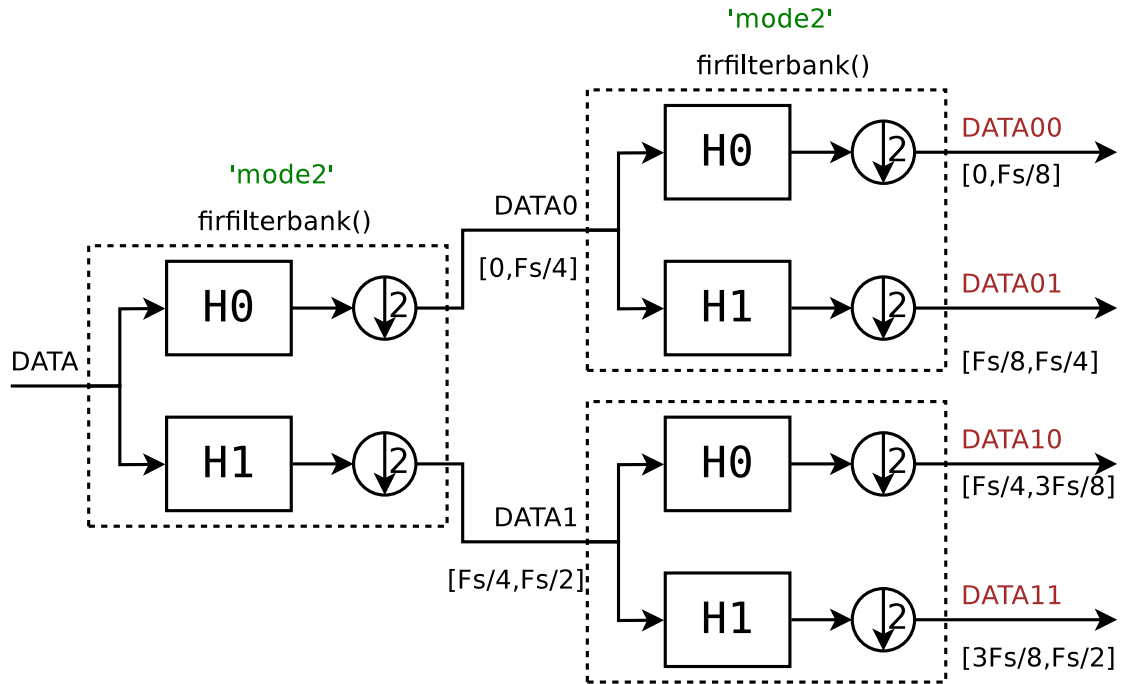


Figure 4.29: Scheme of decomposition in coefficients with information of frequency bands.

The datapack `DATA` was formed by $L = 128$ images (samples) in the directory pointed by `IMAGESDIR` variable and it is separated in datapacks, $\{DATA11, DATA10, DATA01, DATA00\}$, with informations that pertain, each one, at a specific frequency band. In the first step we use a *QMF* filter, and the signal in the datapack `DATA`, with L samples is decomposed in the datapacks (coefficients) `DATA0` and `DATA1`, with informations of frequency bands $[0, F_s/4]$ and $[F_s/4, F_s/2]$ respectively, both outputs having $\lceil L/2 \rceil$ images. In the second step two *QMF* filters are used, and the datapacks `DATA0` and `DATA1` are decomposed in the coefficients $\{DATA00, DATA01\}$ and $\{DATA10, DATA11\}$, with informations of frequency bands $\{[0, F_s/8], [F_s/8, F_s/4]\}$ and $\{[F_s/4, 3F_s/8], [3F_s/8, F_s/2]\}$ respectively, all outputs having $\lceil L/4 \rceil$ images.

At this point it is important to note that the coefficients $\{DATA11, DATA10, DATA01, DATA00\}$ do not have an immediate relation with the datapack `DATA`, and to reconstruct this signal it is necessary to apply the Inverse Discrete Wavelet Transform (*IDWT*) to the coefficients.

4.8.3 Reconstruction Using the Inverse Discrete Wavelet Transform

The next code shows the use of `firsynthesisbank()` function, which implements a step of *IDWT*. This is a *QMF* filter in synthesis mode with two inputs and one output, so that if the inputs have L samples then the output has $2L$ samples, which contain the reconstructed spectral information of both inputs.

```
DATA0_ = firsynthesisbank(DATA00, DATA01, H0);
DATA1_ = firsynthesisbank(DATA10, DATA11, H0);
```

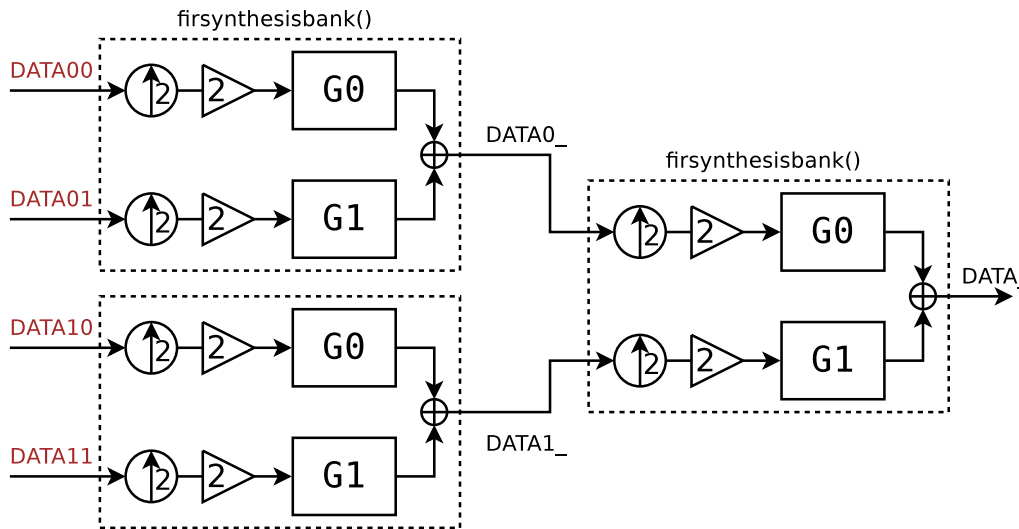


Figure 4.30: Reconstruction scheme of the coefficients obtained in a *DWT*.

```
DATA_ = firsynthesisbank(DATA0_,DATA1_,H0);
```

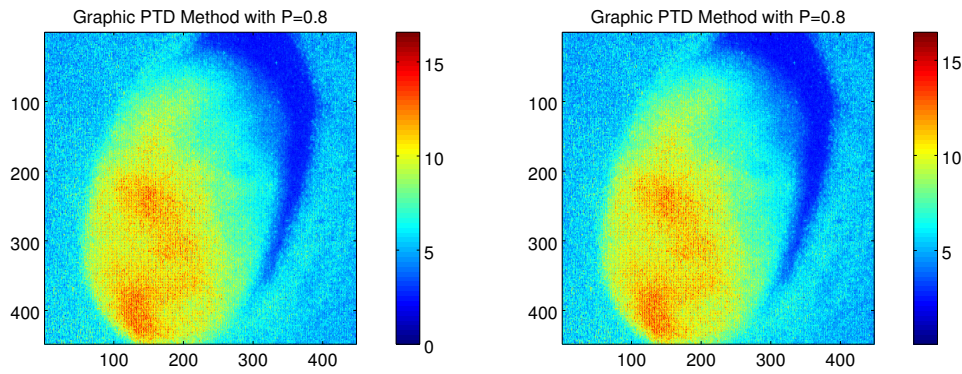
The code implements a reconstruction scheme as shown in the Figure 4.30, where the coefficients $\{DATA00, DATA01, DATA10, DATA11\}$ were obtained with a decomposition as presented in Figure 4.29. All the *QMF* filters in synthesis mode implemented with the `firsynthesisbank()` function receive as input parameter a *H0* *FIR* filter, with cut-off frequency at $F_s/4$, which is the same used in the decomposition scheme shown in the Figure 4.29. Thus, *H0* is used to calculate the filters *G0* and *G1* used in the synthesis blocks in the Figure 4.30.

In the first step of the reconstruction scheme, the coefficients $\{DATA00, DATA01\}$ and $\{DATA10, DATA11\}$, with $L = 32$ samples each one, are used by two *QMF* filters in synthesis mode to get the reconstructed coefficients *DATA0_* and *DATA1_*, with $2L$ samples, homologous to the coefficients *DATA0* and *DATA1* in the Figure 4.29. In the second step, the coefficients *DATA0_* and *DATA1_* are used by *QMF* filter in synthesis mode, and a datapack *DATA_* is recovered, which is the reconstructed version of *DATA* seen in the scheme of Figure 4.29.

We can compare the original datapack and the reconstructed version using any of the biospeckle analysis method seen in the last sections; for example, we can use the *PTD* method in graphic mode:

```
GPTD = graphptd(DATA ,0.8);
GPTD_ = graphptd(DATA_ ,0.8);
```

The Figure 4.31 shows the matrices *GPTD* and *GPTD_*, that are the results of the *PTD* method in graphic mode on the datapack *DATA* and your homologue *DATA_*.



(a) *PTD* method in graphic mode of a data-pack DATA.

(b) *PTD* method in graphic mode of a data-pack DATA_.

Figure 4.31: Comparison between the *PTD* activity indicator of datapack DATA and DATA_.

4.8.4 Partial Reconstruction Using The Inverse Discrete Wavelet Transform

The next code shows the use of `firsynthesispath()` function, that implements a branch of *IDWT* scheme, being this a set, in cascade, of *QMF* filters in synthesis mode following a path. So that if the input datapack (coefficient) has L samples then the output has ML samples, being M the number of steps used in the decomposition scheme of an input. Thus, the function reconstructs partially the original signal, obtaining from each coefficient a signal with the information of only a frequency band.

```
DATA00_ = firsynthesispath(DATA00,H0,[0 0]);
DATA01_ = firsynthesispath(DATA01,H0,[0 1]);
DATA10_ = firsynthesispath(DATA10,H0,[1 0]);
DATA11_ = firsynthesispath(DATA11,H0,[1 1]);
```

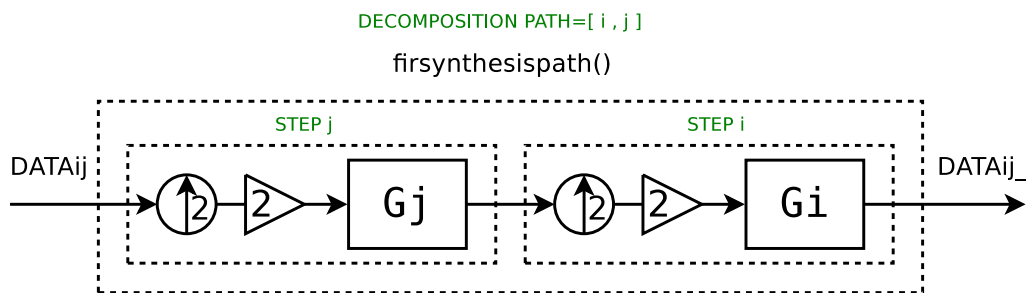


Figure 4.32: Reconstruction scheme of a coefficient with the analysis path $[i, j]$.

The code implements, for each coefficient obtained in the decomposition scheme of Figure 4.29, a reconstruction branch as shown in the Figure 4.32. Where, following the name notation used in the decomposition scheme, the end part of the coefficient names indicates the decomposition path used in their creation. For example, the datapack $DATA_{ij}$ follows the decomposition path $[i, j]$. Thus, all the steps implemented with the `firsynthesispath()` function follow an inverse path of the decomposition. Finally, the `firsynthesispath()` function reconstructs the information in the

coefficient $DATA_{ij}$ using a H_0 FIR filter and the decomposition path $[i\ j]$. H_0 is used to calculate the filters G_0 and G_1 , being H_0 a filter with cut-off frequency at $F_s/4$, and it is the same used in the decomposition scheme shown in the Figure 4.29. The datapacks $\{DATA_{00_}, DATA_{01_}, DATA_{10_}, DATA_{11_}\}$ represent the reconstructed signals of the coefficients $\{DATA_{00}, DATA_{01}, DATA_{10}, DATA_{11}\}$ respectively. So that, each signal $DATA_{ij_}$ has the information of the frequency band ij . Thus, we can say that, $DATA_$, the sum of all $DATA_{ij_}$ signals, is equivalent to the reconstructed signal $DATA_$ seen in the Section 4.8.3.

$$DATA_ = \sum_{i,j} DATA_{ij_}. \quad (4.1)$$

Having the datapacks $\{DATA_{00_}, DATA_{01_}, DATA_{10_}, DATA_{11_}\}$, we can compare them using any of the biospeckle analysis methods seen in the last sections; by example we can use the *AVD* method in graphic mode:

```
GAVD11_ = graphavd(DATA11_, 'off');
GAVD10_ = graphavd(DATA10_, 'off');
GAVD01_ = graphavd(DATA01_, 'off');
GAVD00_ = graphavd(DATA00_, 'off');
```

The next code generates the Figure 4.33, which shows the matrices $\{GAVD11_ , GAVD10_ , GAVD01_ , GAVD00_ \}$ with the same color map.

```
figure(3);
imagesc(GAVD11_, [0 27.062]);colorbar;
title('Graphic AVD method');

figure(4);
imagesc(GAVD10_, [0 27.062]);colorbar;
title('Graphic AVD method');

figure(5);
imagesc(GAVD01_, [0 27.062]);colorbar;
title('Graphic AVD method');

figure(6);
imagesc(GAVD00_, [0 27.062]);colorbar;
title('Graphic AVD method');
```

4.8.5 Filtering Using the Continuous Wavelet Transform of a Discrete Signal

The first step to work with the Continuous Wavelet Transform (*CWT*) of a discrete signal is to choose the mother wavelet. In this sense, the next code shows a way to get a set between the domains $\{-3.5, +3.5\}$, of Morlet wavelets (MORLET et al., 1982a; MORLET et al., 1982b), according to the *CWT* scheme shown in the Figure 4.34b, where a datapack *DATA* is decomposed

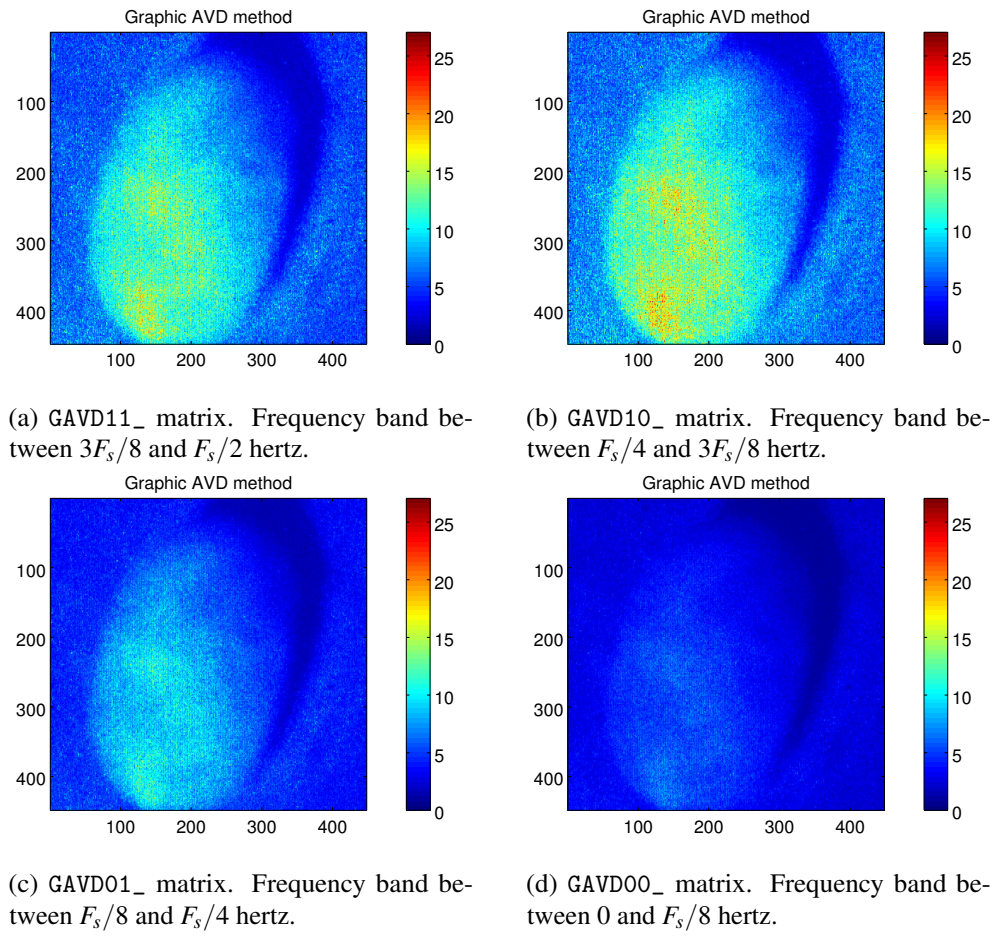


Figure 4.33: AVD method in graphic mode of the datapacks $\{\text{DATA00_}, \text{DATA01_}, \text{DATA10_}, \text{DATA11_}\}$.

using the wavelets based in the mother wavelet ψ_0 .

```

SIDEPOINTS=16;           % Side points of mother wavelet.

PSI0 = morlet(-3.5,3.5, 2*SIDEPOINTS+1);
PSI1 = morlet(-3.5,3.5, 4*SIDEPOINTS+1);
PSI2 = morlet(-3.5,3.5, 8*SIDEPOINTS+1);

PSI0=PSI0/sum(PSI0.^2);
PSI1=PSI1/sum(PSI1.^2);
PSI2=PSI2/sum(PSI2.^2);

```

As it can be seen, the code gets the vectors PSI0, PSI1 and PSI2, that represent the wavelets ψ_0 , ψ_1 and ψ_2 with 33, 65 and 129 points respectively; so that the wavelet ψ_i is a 2^i scaled function of mother wavelet ψ_0 , and all wavelets use their middle element to represent the sample in the time zero. The code also normalize the amplitude to fulfill the wavelet condition (POPOV; ZHUKOV, 2009).

Once the creation of the wavelets is completed, we can see the frequency response using the `freqmod()` function, which has two input parameters, the first is a vector that represents a wavelet to analyze and the second parameter is the resolution (number of points) of the frequency response. The function returns two vectors, the first is the modulus of Fourier transform of the studied wavelet, and the second is the normalized frequency for each element of the first vector. The normalization is performed so that the value 2 represents the frequency sampling value F_s . Thus, the next code shows the use of the `freqmod()` function over the vectors PSI0, PSI1 and PSI2.

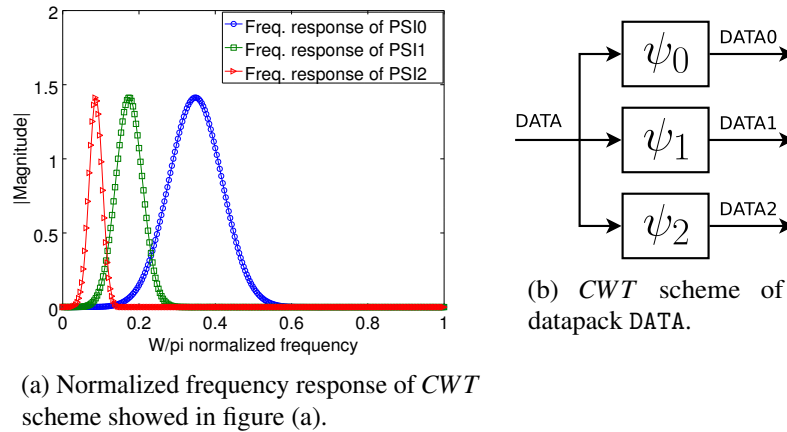


Figure 4.34: Scheme of continuous wavelet transform of a discrete signal.

Figure 4.34a shows graphically the frequency response of wavelets. It is easy to see that, by using of a scaled factor 2^i in the creation of wavelets, if the wavelet ψ_0 is centered in the frequency F_0 , then the wavelet ψ_i will be centered in the frequency $F_0/2^i$.

```
RESOLUTION = 256; % Graphic resolution.

[FR0 F0] = freqmod(PSI0,RESOLUTION);
[FR1 F1] = freqmod(PSI1,RESOLUTION);
[FR2 F2] = freqmod(PSI2,RESOLUTION);

figure(1);
plot(F0,FR0,'-o',F1,FR1,'-s',F2,FR2,'->');
legend('Freq. response of PSI0', ...
       'Freq. response of PSI1', ...
       'Freq. response of PSI2');
ylabel('Magnitude');
xlabel('W/\pi normalized frequency');
```

Once the wavelets are defined, we can decompose a datapack DATA (signal of Figure 4.34b) convolving it with the wavelets. The next code shows the use of the `datapack_conv()` function, which implements the convolution of two signals, where the first input parameter is a datapack DATA, which represents the input signal and the second is a vector, which represents the impulse response of the convolver block. Thus, the convolution is performed on the scope of the first parameter, and

the number of samples that returns the function is the same as in the DATA. The datapack DATA was formed by 128 images (samples) in the directory pointed by the IMAGESDIR variable.

```
DATA = datapack(IMAGESDIR, '', 1, 128, 'bmp');

DATA0 = datapack_conv(DATA, PSI0);
DATA1 = datapack_conv(DATA, PSI1);
DATA2 = datapack_conv(DATA, PSI2);
```

The datapacks DATA0, DATA1 and DATA2 conform to the *CWT* of datapack DATA, and their signals represent DATA portions according to the frequency response shown in the Figure 4.34a. The next code shows the analysis of these datapacks using the *AVD* method in graphic mode.

```
GAVD0 = graphavd(DATA0, 'off');
GAVD1 = graphavd(DATA1, 'off');
GAVD2 = graphavd(DATA2, 'off');
```

The next code generates the Figure 4.35, which shows graphically the comparison of the matrices GAVD0, GAVD1 and GAVD2, using the same color map.

```
figure(2);
imagesc(GAVD0, [0 18.226]);colorbar;
title('Graphic AVD method');

figure(3);
imagesc(GAVD1, [0 18.226]);colorbar;
title('Graphic AVD method');

figure(4);
imagesc(GAVD2, [0 18.226]);colorbar;
title('Graphic AVD method');
```

4.9 Extra Functions

4.9.1 Mean Values in Analysis Windows

The next code shows the use of the `mwindowing()` function, which implements a method to get the mean values in analysis windows over a matrix.

```
DATA=datapack(IMAGESDIR, '', 1, 128, 'bmp');

GAVD=graphavd(DATA, 'off');

WLines=7;
WColumns=7;
GAVDW = mwindowing(GAVD, WLines, WColumns);
```

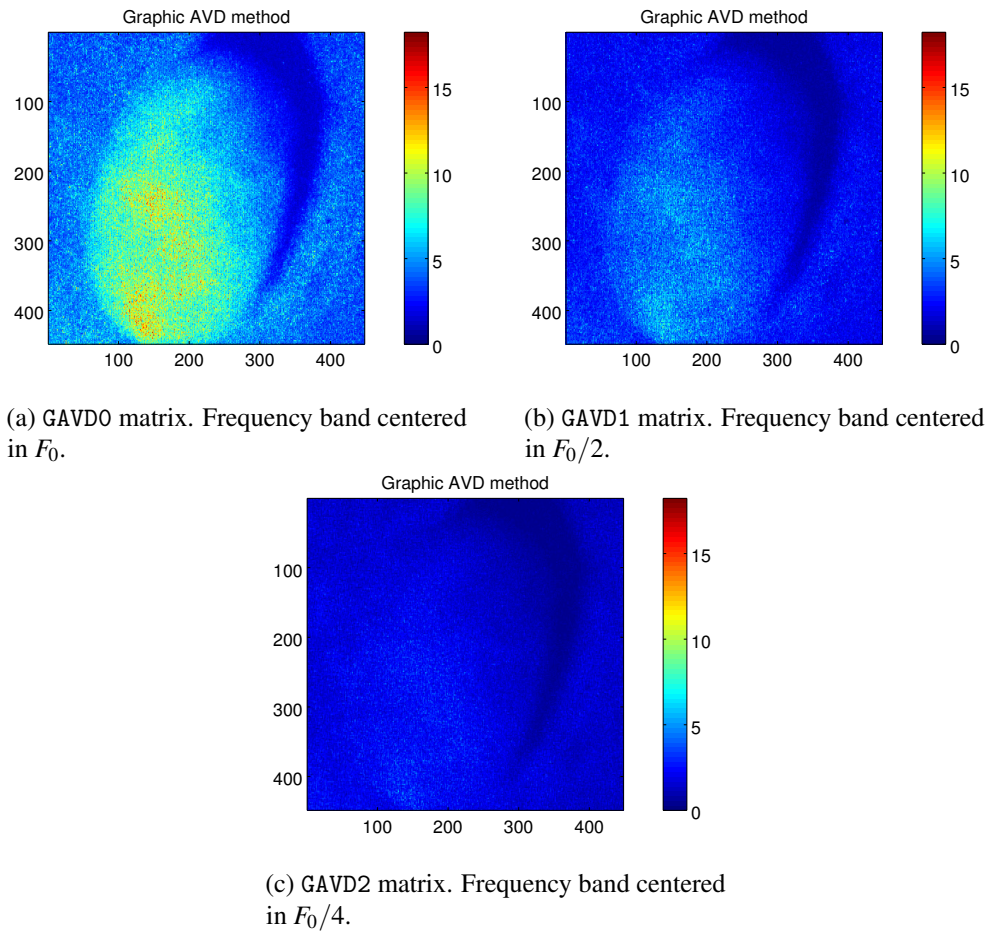


Figure 4.35: AVD method in graphic mode of the datapacks DATA0, DATA1 and DATA2.

The code returns the GAVDW matrix, filled with the mean values of the analysis windows in the GAVD matrix; being that, the GAVD matrix was segmented in windows of $W_{Lines} \times W_{Columns}$ pixels. This matrix is the result of applying the AVD method in graphic mode over a datapack DATA, that was formed by 128 images in the directory pointed by the IMAGESDIR variable. Additionally, examples using the functions `datapack()` and `graphavd()` can be found in the Section 4.1 and Section 4.6.5, respectively.

The next code generates a set of graphics comparing the matrices GAVD and GAVDW. The `figure(1)` and `figure(2)` show the matrices GAVD and GAVDW, respectively. In both cases the same color map are used. By other side, the `figure(3)` shows GAVDW matrix; so that, the blue and red colors represent the minimum and maximum values in the matrix. All these graphics can be seen in the Figure 4.36.

```
figure(1);
imagesc(GAVD, [0,35.654]);
colorbar;
title('Graphic AVD method');

figure(2);
```

```

imagesc(GAVDW,[0,35.654]);
colorbar;
title('Mean values in the graphic AVD method');

figure(3);
imagesc(GAVDW);
colorbar;
title('Mean values in the graphic AVD method');

```

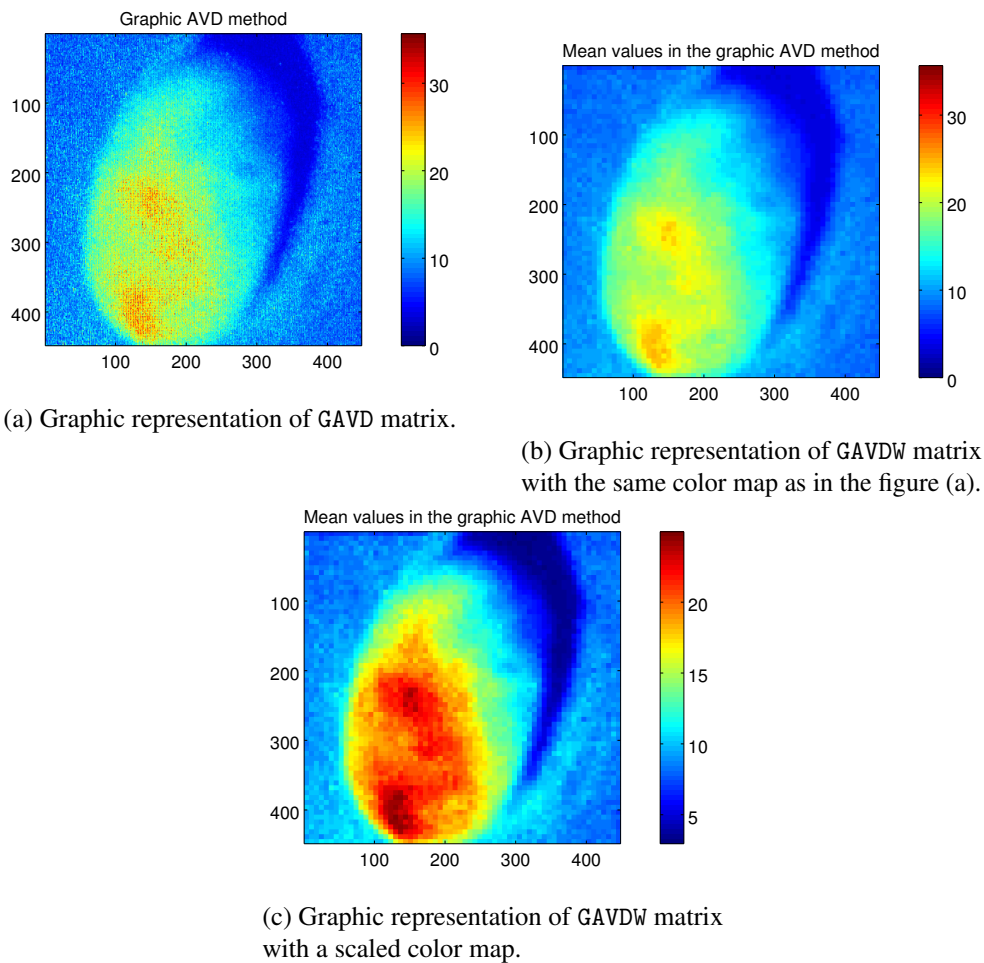


Figure 4.36: Comparison between the matrices GAVD and GAVDW.

4.9.2 Matrix Threshold

The next code shows the use of `threshold2d()` function, which implements a threshold function.

```

M = [ [0 10 20 30]; ...
      [30 20 20 10]; ...
      [0 10 2 0]; ...
      ]

```

```
[A B]=threshold2d(M,20)
```

The matrix A is a truncated version of matrix M , so that any value of A is greater than 20 and all values greater than 20 are replaced by 20. Additionally, the `threshold2d()` function returns the B matrix so that $A + B = M$, as presented highlighted.

```
M =
    0    10    20    30
   30    20    20    10
    0    10     2     0

A =
    0    10    20    20
   20    20    20    10
    0    10     2     0

B =
    0     0     0    10
   10     0     0     0
    0     0     0     0
```

4.9.3 Binary Entropy of Probability Mass Function

The `hbpmf()` function implements the binary entropy from a probability mass function. For example, the next code calculates the binary entropy $H \equiv H_b(A, B)$ of two random variables A and B ; with possible values $\{A_0, A_1, A_2\}$ and $\{B_0, B_1, B_2, B_3\}$ respectively; and with a joint probability mass function $P \equiv Pr(A, B)$ as shown in the Table 4.1.

$Pr(A, B)$	B_0	B_1	B_2	B_3
A_0	0.0	0.1	0.2	0.1
A_1	0.1	0.2	0.0	0.1
A_2	0.0	0.1	0.1	0.0

Table 4.1: Joint probability mass function of random variables A and B .

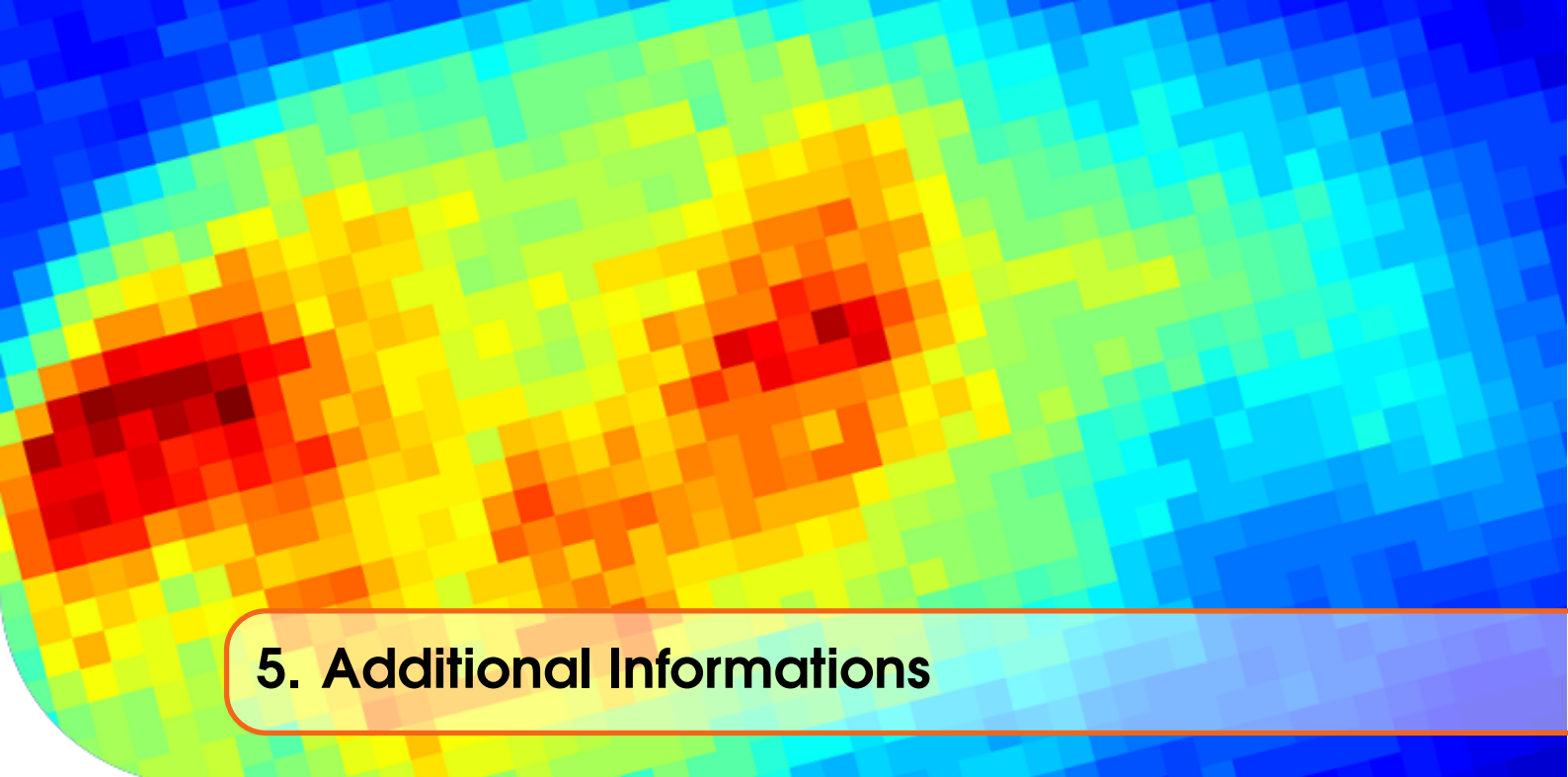
```
P = [ [0.0 0.1 0.2 0.1]; ...
      [0.1 0.2 0.0 0.1]; ...
      [0.0 0.1 0.1 0.0]; ...
    ];

SUM=sum(sum(P))

H=hbpmf(P)
```

Here, it is important to note that the sum of all elements in P should be 1.0 given that this is a probability. The result can be seen highlighted:

```
SUM = 1  
H = 2.9219
```

5. Additional Informations

5.1 Frequently Asked Questions

1. **What do I need to proceed when this message appears: Warning: your version of GraphicsMagick limits images to 8 bits per pixel**

This message is only a warning, and it means that GraphicsMagick can only work with images of 8 bits per pixel. The *BSLTL* package was written for working with 8 bits, so there is no problem in the *BSLTL* applications. However, if this message is annoying you, the problem can be solved by compiling the last stable version of GraphicsMagick (currently v.1.3.21). Before compiling the GraphicsMagick, all dependencies should be installed. In GNU/Linux, you can do it using this instruction in the Terminal:

```
sudo apt-get build-dep libgraphicsmagick++3
```

2. **Which operating systems support the *BSLTL* package?**

Given that the *BSLTL* package is an interpreted software, it can be run on any platform that support the interpreter OCTAVE or MATLAB. Moreover, some of this platforms have been already tested, for example:

- (a) Ubuntu 14.04 (32 bits)
- (b) Ubuntu 16.04 (64 bits)
- (c) Debian 8 (64 bits)
- (d) Android 6.0 (MPB24.65-34-3)
- (e) Microsoft Windows 7 (32 bits)

3. **What license does the *BSL Tool Library* use?**

This project use the General Public License for all source code. You can download a copy at <http://www.gnu.org/licenses>. One copy of license is also included in the *BSLTL* package.

4. How can I contribute to the *BSL Tool Library*?

All comments, suggestions or requests regarding new features are welcome, and they can be sent to the project mailing list (see next section). We promise to read all carefully and, if necessary, we will make the required changes. In case you want to include a new function in the library, submissions will only be accepted with the source code documented and accompanied by a tutorial (all these should be under General Public License or any compatible).

5.2 Interesting Links

Home page: <http://www.nongnu.org/bsltool/>

This is the official website of the project, where it is possible to access many features: source code, download, mailing list, bug reports, etc.

Version control repository: <http://www.nongnu.org/bsltool/code.html>

Here you can find the previous source code version of the *BSL Tool Library* before a official release. In order to get the last official release version, please go to the download page.

Download page: <http://www.nongnu.org/bsltool/download.html>

This page has the last official release of *BSL Tool Library*.

Mailing list: <http://www.nongnu.org/bsltool/mailling.html>

This mailing list is the address to comments, suggestions or requests for new features.

Bug reports: <http://www.nongnu.org/bsltool/bugs.html>

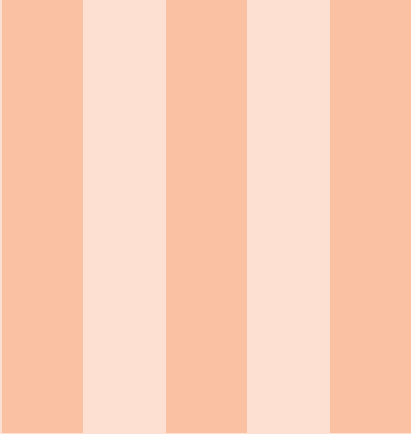
Here you can report any detected error.

5.3 BSL - Picture Packages

In order to study the biospeckle laser phenomenon, it is necessary to have the equipment to acquire a set of speckle images produced by a laser. However, here we list a set of links with picture packages for free download and use.

Data from coffee seed: <http://repositorio.ufla.br/jspui/handle/1/10619>

Data from maize seed: <http://repositorio.ufla.br/jspui/handle/1/10560>



Part 3: BSL Tool Library Reference Manual

6	Functions' Descriptions	113
6.1	Loading the Datapack	
6.2	Creating the Time History Speckle Pattern	
6.3	Creating and Working with the Co-occurrence Matrix	
6.4	Calculating the Inertia Moment	
6.5	Calculating the Absolute Value of the Differences	
6.6	Calculating the Regular Value of the Differences	
6.7	Numerical Analysis of the Average Difference Method	
6.8	Calculating the Correlation in the Speckle Pattern	
6.9	Graphical Methods	
6.10	Frequency Analysis	
6.11	Extra Functions	
6.12	Quality Tests	
	Bibliography	157
	Books	
	Sites	
	Index	161

6. Functions' Descriptions

6.1 Loading the Datapack

In the context of this manual we call datapack a 3D matrix created by grouping a collection of $NTIMES$ images $\mathbf{I}_k, \forall k \in \{1, 2, \dots, NTIMES\}$, with $NLIN$ lines and $NCOL$ columns (Figure 6.1). The matrix $\mathbf{I}_k$ represents an image in gray scale, where the image is taken in the $k - th$ picture.

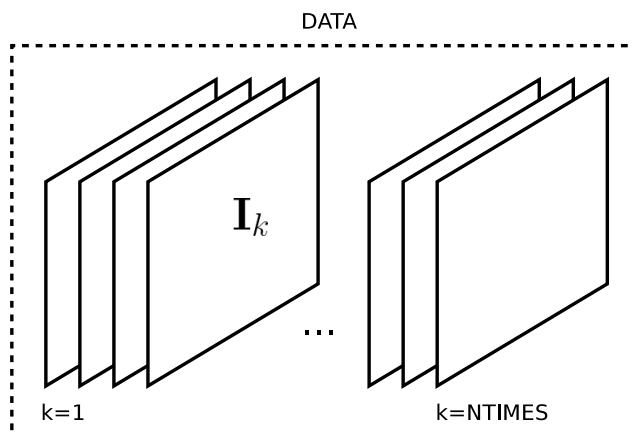


Figure 6.1: Datapack of a 3D matrix created by grouping $NTIMES$ matrices with $NLIN$ lines and $NCOL$ columns.

6.1.1 Function datapack() - Creating a 3D Matrix

This command creates a datapack (3D matrix) from a set of images in a directory. All images should have the same size and format.

For using this function, just type the following commands at the prompt:

```
DATA = datapack(IMGDIR, IMGNAME, IMGN1, IMGN2, IMGFMT) ;  
DATA = datapack(IMGDIR, IMGNAME, IMGN1, IMGN2, IMGFMT, LPOS, CPOS, NLIN, NCOL) ;
```

```
% Image named as image1.bmp, image2.bmp, ..., image133.bmp
DATA = datapack('D:\images\sample1','image',1,333,'bmp');
% To load only a window of NLINxNCOL pixels at position (LPOS,CPOS)
DATA = datapack('D:\images\sample1','image',1,333,'bmp',LPOS,CPOS,NLIN,NCOL);
```

Inputs:

IMGDIR	-	is the directory path where the images are. These images should have their names in the following format NAME=[IMGNAME, 'num', '.', IMGFMT]. For example, the images could have the following names: fig1.bmp, fig2.bmp, ..., fig128.bmp, so that, for this case, IMGNAME='fig', IMGFMT='bmp' and num will be an integer with values from IMGN1=1 to IMGN2=128.
IMGNAME	-	is the pre-name of the file. For example: if the files in IMGDIR are tagged as fig1.bmp, then IMGNAME='fig'.
IMGN1	-	is the index of the first image. For example: if the files have names from fig1.bmp to fig128.bmp, then IMGN1=1.
IMGN2	-	is the index of last image. For example: if the files have names from fig1.bmp to fig128.bmp, then IMGN2=128.
IMGFMT	-	is the file format of the images. For example: if the files are tagged as fig1.bmp, then IMGFMT='bmp'.
LPOS	-	[Optional] . It is the first line position of a region of interest (ROI) in the images. When it is used this optional parameter, CPOS, NLIN and NCOL are mandatory and the datapack is formed by the ROI. If this parameter is not used, then the datapack is formed with the complete image.
CPOS	-	[Optional] . It is the first column position of a ROI.
NLIN	-	[Optional] . It is the number of lines of a ROI.
NCOL	-	[Optional] . It is the number of columns of a ROI.

Output:

DATA	-	is the speckle datapack, where DATA is a 3D matrix created by grouping NTIMES=IMGN2-IMGN1+1 matrices with NLIN lines and NCOL columns. When N=size(DATA), then N(1,1) represents NLIN, N(1,2) represents NCOL and N(1,3) represents NTIMES.
------	---	--

6.1.2 Function datacut() - Selecting and Cutting the Image Datapack

The *datacut()* is a function in graphic mode to select and to cut a portion of the speckle datapack (DATA).

For using this function, just type the following commands at the prompt:

```

DATACUT = datacut(DATA);

% Load the indexes of LINES and COLUMNS cropped in the datapack DATA.
[DATACUT LINES COLUMNS] = datacut(DATA);

```

Input:

DATA - is the speckle data, where DATA is a 3D matrix created by grouping NTIMES matrices with NLIN lines and NCOL columns. When `N=size(DATA)`, then `N(1,1)` represents NLIN and `N(1,2)` represents NCOL and `N(1,3)` represents NTIMES.

Outputs:

DATACUT - is the cut part of the speckle datapack (DATA), where DATACUT is a 3D matrix created by grouping NTIMES cut parts of DATA.

LINES - **[Optional]** is a vector with two elements, which are the first line and the last line in the cut part of DATA.

COLUMNS - **[Optional]** is a vector with two elements, which are the first column and the last column in the cut part of DATA.

6.1.3 Function `datapack_to_gif()` - Datapack to GIF file

This function creates a Graphic Interchange Format (*GIF*) file from the datapack (DATA), in order to create an animated graphic file.

For using this function, just type the following commands at the prompt:

```

h=datapack_to_gif(DATA,Filename,Frames);

datapack_to_gif(DATA,Filename,Frames);
datapack_to_gif(DATA,Filename,Frames,Map);
datapack_to_gif(DATA,Filename,Frames,Map,Time);

% Creates a gif file with a gray colormap
datapack_to_gif(DATA,Filename,Frames,gray);

% Creates a gif file of datapack DATA with
% a filename 'image.gif', 10 frames,
% a jet colormap and 0.8 sec. between frames.
datapack_to_gif(DATA,'image.gif',10,jet,0.8);

```


Inputs:

DATA	-	is the speckle data, where DATA is a 3D matrix created by grouping NTIMES matrices with NLIN lines and NCOL columns. When $N = \text{size}(\text{DATA})$, then $N(1,1)$ represents NLIN, $N(1,2)$ represents NCOL and $N(1,3)$ represents NTIMES.
Filename	-	is the name of the gif file.
Frames	-	is the number of frames in the gif file.
Map	-	[Optional] is the color map, where the value can be: jet, gray, hsv, ..., etc. By default it is jet. For example, see the command: <code>colormap('list')</code>
Time	-	[Optional] is the time between frames in the gif file. By default it is 0.5 sec.

Output:

h	-	is the name of gif file.
---	---	--------------------------

6.2 Creating the Time History Speckle Pattern

Time History Speckle Pattern (*THSP*) is a 2D matrix that represents the evolution on time of a set of M points in the speckle image. Thus, this set of points will be represented in the lines of a *THSP* matrix, and the columns represent the evolution on time, as shown in Figure 6.2.

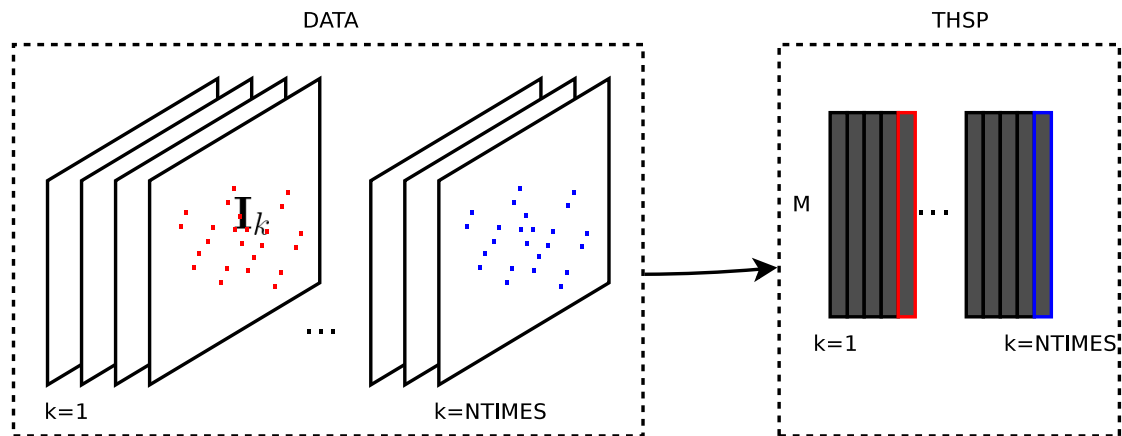


Figure 6.2: Time history speckle pattern construction. Evolution on time of a set of M points in the speckle image. Thus, this set of points will be represented in the lines of a *THSP* matrix, and the column represents the evolution on time

6.2.1 Function `thsp()` - Time History Speckle Pattern

This function returns the Time History Speckle Pattern (*THSP*) (OULAMARA; TRIBILLON; DUVERNOY, 1989b; XU; JOENATHAN; KHORANA, 1995), so that it creates a 2D matrix with M lines and NTIMES columns, where M represents the number of points analyzed, which is equal to the number of columns or lines of image I_k in the datapack. Thus, the user can choose between

these two options. *NTIMES* represents the number of images in the datapack and $1 \leq k \leq \text{NTIMES}$. Therefore, the *THSP* pattern is a matrix where each column is a representation of only one line or column in the *S* position of all $\mathbf{I}_k$ matrices whereas the other lines or columns are discarded (Figure 6.3).

For using this function, just type the following command at the prompt:

```
Y = thsp(DATA,R,S);
```

Inputs:

-
- DATA** - is the speckle datapack, where *DATA* is a 3D matrix created by grouping *NTIMES* intensity matrices with *NLIN* lines and *NCOL* columns. When $\mathbf{N}=\text{size}(\text{DATA})$, then
- $\mathbf{N}(1,1)$ represents *NLIN*,
 - $\mathbf{N}(1,2)$ represents *NCOL* and
 - $\mathbf{N}(1,3)$ represents *NTIMES*.
- R** - is a parameter to define if one line or column is to be analyzed over time:
- if *R* is equal to 1, you choose the lines from the images to make the THSP and
 - if *R* is equal to 2, you choose the columns, otherwise an error occurs.
- S** - is the line or column position used to make the time history speckle pattern. The function does not verify if *S* is in the range, so the user has to check it previously.
-

Output:

-
- Y** - is the THSP matrix, where *Y* is a 2D matrix with *M* lines and *NTIMES* columns. *M* is equal to *NCOL* when *R* is 1, or *M* is equal to *NLIN* when *R* is 2. *Y* is a matrix where each column is a representation of only one line or column in the position *S* of *DATA*.
-

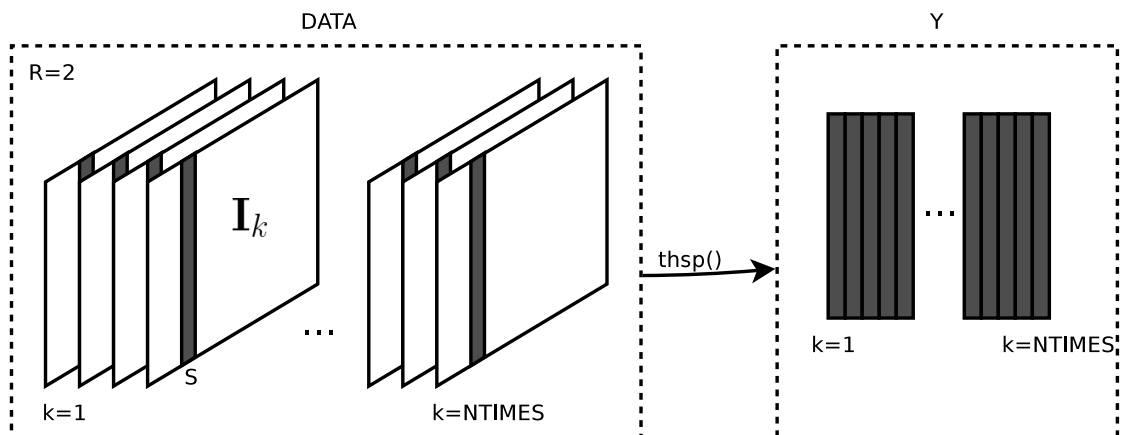


Figure 6.3: Time history speckle pattern of a line.

6.2.2 Function `thsp_line()` - Time History Speckle Pattern of a Line

This function is an alias of `thsp()` function. See section 6.2.1.

6.2.3 Function `thsp_random()` - Time History Speckle Pattern of a Random Points Set

This function creates the Time History Speckle Pattern (*THSP*) from a set of M points selected randomly, following a uniform distribution to choose the positions in the first image of the datapack. The pattern *THSP* is a 2D matrix with M lines and *NTIMES* columns, where M is the number of selected random points in each image $\mathbf{I}_k$ for $1 \leq k \leq \text{NTIMES}$, where *NTIMES* is the number of images. Thus, the main point here is that the pixels of the first column of the *THSP* represent a set of points chosen in the first image. These points, randomly selected in the first image, stand still and appear in the same position in the other *NTIMES*-1 images.

For using this function, just type the following commands at the prompt:

```
THSP          = thsp_random(DATA, M);
[THSP POINTS] = thsp_random(DATA, M);
```

Inputs:

-
- | | |
|------|--|
| DATA | - is the speckle datapack, where DATA is a 3D matrix created by grouping <i>NTIMES</i> matrices with <i>NLIN</i> lines and <i>NCOL</i> columns. When $N = \text{size}(\text{DATA})$, then $N(1,1)$ represents <i>NLIN</i> and $N(1,2)$ represents <i>NCOL</i> and $N(1,3)$ represents <i>NTIMES</i> . |
| M | - is the number of points, randomly selected. |
-

Outputs:

-
- | | |
|--------|--|
| THSP | - is the time history speckle pattern matrix, where THSP is a 2D matrix with M lines and <i>NTIMES</i> columns. |
| POINTS | - [Optional] . It is a matrix with two columns and M lines. Thus, each line represents each of the random points with their position in the image. The first element of points represents the line position, and the second, the column position. |
-

6.2.4 Function `thsp_gaussian()` - Time History Speckle Pattern of a Random Points Set with Gaussian Distribution

This function creates the Time History Speckle Pattern (*THSP*) from a set of randomly selected M points, following a 2D Gaussian distribution around the point P_0 in the first image $\mathbf{I}_1 \equiv \text{DATA}(:, :, 1)$, and through $\mathbf{I}_k \equiv \text{DATA}(:, :, k)$, $\forall k$. These points are concentrated mostly in a radius *Sigma* around P_0 . The pattern *THSP* is a 2D matrix with M lines and *NTIMES* columns. Thus, the main point here is that the pixels in the first column of the *THSP* are a set of points in the first image. The random points selected in this image stand still and appear in the same position in the other *NTIMES*-1 images, following a Gaussian distribution.

For using this function, just type any of the following commands at the prompt:

```

THSP      = thsp_gaussian(DATA,M,Sigma);

[THSP POINTS]= thsp_gaussian(DATA,M,Sigma,'on');
[THSP POINTS]= thsp_gaussian(DATA,M,Sigma,P0,'on');

```

Inputs:

-
- | | |
|-------|---|
| DATA | - is the speckle datapack, where DATA is a 3D matrix created by grouping NTIMES matrices with NLIN lines and NCOL columns. When $N=\text{size}(\text{DATA})$, then $N(1,1)$ represents NLIN and $N(1,2)$ represents NCOL and $N(1,3)$ represents NTIMES. |
| M | - is the number of points, randomly selected following a Gaussian distribution. |
| Sigma | - is the standard deviation in pixels. |
| P0 | - [Optional] is the initial point, around which M point are selected to create the time history speckle pattern. If this parameter is not used then a window is enabled in order to select a point P0. |
| SHOW | - [Optional] It can be used in the last position of input. It enables a graphic output of the selected points in the THSP. $\text{SHOW}='on'$ for enabling, $\text{SHOW}='off'$ by default. |
-

Outputs:

-
- | | |
|--------|--|
| THSP | - is the time history speckle pattern matrix, where THSP is a 2D matrix with M lines and NTIMES columns. |
| POINTS | - [Optional] . is a matrix with two columns and M lines. Thus, each line represents each of the random points with their position in the image. |
-

6.3 Creating and Working with the Co-occurrence Matrix

The Co-occurrence Matrix (*COM*), is also known as *GLCM* (gray-level co-occurrence matrix), *GLCH* (gray-level co-occurrence histogram) or spatial dependence matrix. Formally, the co-occurrence matrix (HARALICK; SHANMUGAM; DINSTEIN, 1973) of an image **I** of M lines and N columns is defined, for each pixel $I(x,y) \in \mathbf{I}$, as in the Equation (6.1).

$$C_{\Delta x, \Delta y}(a, b) = \sum_{x=1}^M \sum_{y=1}^N \begin{cases} 1, & \text{if } I(x,y) = a \\ & \text{and } I(x+\Delta x, y+\Delta y) = b \\ & \text{and } 1 \leq x+\Delta x \leq M \\ & \text{and } 1 \leq y+\Delta y \leq N \\ 0, & \text{otherwise} \end{cases} \quad (6.1)$$

6.3.1 Function `coom()` - Co-occurrence Matrix

This function calculates the Co-occurrence Matrix (*COM*) (ARIZAGA; TRIVI; RABAL, 1999). The `coom()` function represents the case when in the Equation (6.1) $(\Delta x, \Delta y) = (0, 1)$, and the matrix under analysis is a time history speckle pattern (THSP) matrix, where time is represented in the columns. Thus, the co-occurrence *COM* matrix returned by `coom()` function is calculated as in the Equation (6.2)

$$COM(a, b) = \sum_{i=1}^M \sum_{j=1}^{N-1} \begin{cases} 1, & \text{if } THSP(i, j) = a \\ & \text{and } THSP(i, j+1) = b \\ 0, & \text{otherwise} \end{cases} \quad (6.2)$$

Thus, the co-occurrence matrix represents the occurrences of intensity jumps, between any two consecutive points. Thus, the matrix can be understood as the quantity of times that: any pixel in any instant k with intensity $I_k(x, y) \equiv I_k = i$, has an intensity jump to intensity $I_{k+1} = j$ in the instant $k+1$, see Figure 6.4.

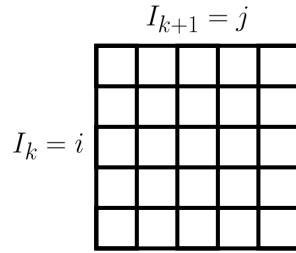


Figure 6.4: Co-Occurrence matrix of a *THSP* patterns.

To use this function just type the following command at the prompt:

```
COM = coom(THSP);
```

Input:

THSP - is an integer 2D matrix that represents the time history speckle pattern. This matrix can be obtained by using the function `thsp()` seen in the Section 6.2. For this purpose it is necessary that the THSP matrix only has values between 0 and 255, since the function does not make this verification and it truncates values outside this interval.

Output:

COM - is a 2D matrix, with 256 lines and 256 columns, which represents the co-occurrence matrix of the THSP matrix. The element $COM(a, b)$ in the *COM* matrix represents the quantity of times that, in two successive columns of the THSP matrix, the intensity values changed from $a - 1$ to $b - 1$.

6.3.2 Function `pmfad()` - Probability Mass Function of the Absolute Difference

The Probability Mass Function of Absolute Difference (*PMFAD*) represents the probabilities of a random variable Z , being $Z = |i - j|$, where i is the position line in COM matrix and j is the position column in COM matrix. Thus, the variable Pr returned by the function `pmfad()` has the value $Pr(z+1)$, representing the probability $Pr(|i - j| = z)$ of happening an absolute intensity jump of value $|i - j| = z$, $0 \leq z \leq 255$. This function calculates the difference probability proposed in (HARALICK; SHANMUGAM; DINSTEIN, 1973). The COM matrix must have 256×256 elements.

For using this function, just type any of the following commands at the prompt:

```
Pr      = pmfad(COM);

[Pr Z] = pmfad(COM);
```

Input:

COM - is the co-occurrence matrix, which is a 2D matrix with 256 lines and 256 columns.

Outputs:

Pr - is a vector with the probability mass function $Pr(|i - j| = z)$, where $Pr(z+1)$ is the probability of happening an absolute intensity jump of value z , $\forall 0 \leq z \leq 255$.
Pr is a vector with 256 elements.

Z - **[Optional]** is a vector with the absolute intensities jumps $|i - j| = z$, so that $Z=[0:255]$.

6.3.3 Function `pmfrd()` - Probability Mass Function of the Regular Difference

The Probability Mass Function of Regular Difference (*PMFRD*) represents the probabilities of a random variable W , being $W = (j - i)$, where i is the position line in COM matrix and j is the position column in COM matrix. Thus the variable Pr returned by the function `pmfrd()` has the value $Pr(w+256)$, representing the probability $Pr((j - i) = w)$ of happening an absolute intensity jump of value i to value $j = i + w$, $-255 \leq w \leq 255$. This function calculates the similar difference probability proposed by Haralick; Shanmugam; Dinstein (1973). The COM matrix must have 256×256 elements.

For using this function, just type any of the following commands at the prompt:

```
Pr      = pmfrd(COM);

[Pr W] = pmfrd(COM);
```

Input:

COM - is the co-occurrence matrix, which is a 2D matrix with 256 lines and 256 columns.

Outputs:

Pr - is a vector with the probability mass function $Pr((j-i) = w)$, where $Pr(w+256)$ is the probability of happening an absolute intensity jump of value w , $\forall -255 \leq w \leq 255$. Pr is a vector with 511 elements.

W - **[Optional]** is a vector with the intensities jumps $(j-i) = w$, so that $W = [-255:255]$.

6.4 Calculating the Inertia Moment

The Inertia Moment (*IM*) (ARIZAGA; TRIVI; RABAL, 1999) analyzes the square difference between the intensity values $I_k = i$ in time k , and the intensity $I_{k+1} = j$ in time $k+1$ for any value of k . Thus, the *IM* method (Equation 6.3) gets a measurement of relative square intensity jump value, in a same region, between the consecutive images I_k and I_{k+1} ,

$$IM = \sum_{ij} M_{ij}(i-j)^2. \quad (6.3)$$

The last equation shows that the *IM* value is a weighted value of $(i-j)^2$, where the weighting factor M_{ij} is calculated from the co-occurrence matrix, COM (Equation 6.4). Many authors have a different way to do the weighting. For example, Arizaga; Trivi; Rabal (1999) define

$$M_{ij} = \frac{COM(i,j)}{\sum_m COM(i,m)}. \quad (6.4)$$

On the other hand Cardoso; Braga (2014) define M_{ij} as in the Equation (6.5),

$$M_{ij} = \frac{COM(i,j)}{\sum_{lm} COM(l,m)}. \quad (6.5)$$

This latter approach is particularly interesting since M_{ij} can be seen as the probability mass function of happening a square intensity jump from i to j , thus, the weighting factor will be $M_{ij} \equiv Pr(i \rightarrow j)$. Thus, the *IM* value will be the expected value of $(i-j)^2$, also known as first moment of $(i-j)^2$, $IM = E[(i-j)^2]$ (Equation 6.6). The Table 6.1 shows the types of inertia moments known in the literature.

Where the *IM* moment (IM_1) is:

$$E[(i-j)^2] = \sum_{ij} \frac{COM(i,j)}{\sum_{lm} COM(l,m)} (i-j)^2. \quad (6.6)$$

Table 6.1: Inertia moment types.

TYPE	EQUATION	DESCRIPTION
1	$IM_1 = E[(i - j)^2]$, see Equation (6.6)	The Inertia Moment proposed by Cardoso; Braga (2014). This can also be called as second moment of <i>AVD</i> . (Absolute Value of the Differences)
2	$IM_2 = E_{Ariz.}[(i - j)^2]$, see Equation (6.7)	The Inertia Moment with the normalized co-occurrence matrix (COM) proposed by Arizaga; Trivi; Rabal (1999).

And, the *IM* moment following Arizaga; Trivi; Rabal (1999) (IM_2) is presented as in the Equation (6.7),

$$E_{Ariz.}[(i - j)^2] = \sum_{ij} \frac{COM(i, j)}{\sum_m COM(i, m)} (i - j)^2. \quad (6.7)$$

6.4.1 Function `inertiamoment()` - Inertia Moment Method

This function returns the Inertia Moment (*IM*) (ARIZAGA; TRIVI; RABAL, 1999). The `inertiamoment()` function needs a co-occurrence matrix COM as input and calculates the two types of *IM* values shown in Table 6.1. By default the `inertiamoment()` function returns the IM_1 , see (6.6), but this function accepts one second parameter. See Table 6.1.

For using this function, just type any of the following commands at the prompt:

```
IM1 = inertiamoment(COM);
% If it is necessary to get IM2
[IM1 IM2] = inertiamoment(COM, 2);
```

Inputs:

- | | |
|------|--|
| COM | - is a 2D matrix, with 256 lines and 256 columns, which represents a co-occurrence matrix of a THSP matrix, see section 6.2. The element $COM(a, b)$ in the co-occurrence matrix represents the quantity of times that, in two successive columns of a THSP matrix, the intensity values jumps from $a - 1$ to $b - 1$. |
| TYPE | - [Optional] When this parameter is used, the function returns an additional result in the same position of output.
If TYPE is equal to 2, the function also returns the IM_2 (see Table 6.1). |

Outputs:

- | | |
|-----|--|
| IM1 | - is the value of IM_1 (see Table 6.1). |
| X | - [Optional] If TYPE is equal to 2, the function also returns the IM_2 (see Table 6.1). |

6.5 Calculating the Absolute Value of the Differences

The Absolute Value of the Differences (*AVD*) method (BRAGA et al., 2011) is a modification of the Inertia Moment (*IM*) (ARIZAGA; TRIVI; RABAL, 1999). where we analyze the absolute difference between the intensity value $I_k = i$ in time k , and the intensity $I_{k+1} = j$ in time $k + 1$ for any value of k . Thus, the *AVD* method (Equation 6.8) presents a measurement relative to the absolute intensity jump value between the consecutive images, $\mathbf{I}_k$ and $\mathbf{I}_{k+1}$,

$$AVD = \sum_{ij} M_{ij} |i - j|. \quad (6.8)$$

The last equation shows that the *AVD* value is a weighted value of $|i - j|$, where the weighting factor M_{ij} is calculated from the co-occurrence matrix *COM*, as seen in section 6.3 and 6.4.1. The *AVD* value will be the expected value of $|i - j|$, also known as first moment of $|i - j|$, $AVD = E[|i - j|]$. The Table 6.2 shows the types of *AVD* known in the literature.

Table 6.2: *AVD* types.

TYPE	EQUATION	DESCRIPTION
1	$AVD_1 = E[i - j]$, see Equation (6.9)	The <i>AVD</i> method with the normalized co-occurrence matrix (<i>COM</i>) proposed by Cardoso; Braga (2014). This can also be called as first moment of <i>AVD</i> .
2	$AVD_2 = E[(i - j)^2]$, see Equation (6.10)	The Inertia Moment proposed by Cardoso; Braga (2014). This can also be called as second moment of <i>AVD</i> .
3	$AVD_3 = Var[i - j]$, see Equation (6.11)	Knowing AVD_1 and AVD_2 it is easy to calculate the <i>AVD</i> second central moment.
4	$AVD_4 = E_{Ariz.}[i - j]$, see Equation (6.12)	The <i>AVD</i> method with the normalized co-occurrence matrix (<i>COM</i>) proposed by Arizaga; Trivi; Rabal (1999).

Where the *AVD* first moment (AVD_1) is:

$$E[|i - j|] = \sum_{ij} \frac{COM(i, j)}{\sum_{lm} COM(l, m)} |i - j| \quad (6.9)$$

The *AVD* second moment (AVD_2) is:

$$E[(i - j)^2] = \sum_{ij} \frac{COM(i, j)}{\sum_{lm} COM(l, m)} (i - j)^2 \quad (6.10)$$

The *AVD* second central moment (AVD_3) is:

$$Var[|i - j|] = E[(i - j)^2] - E[|i - j|]^2 \quad (6.11)$$

And the *AVD* first moment following Arizaga; Trivi; Rabal (1999) (*AVD*₄) is:

$$E_{Ariz.}[|i-j|] = \sum_{ij} \frac{COM(i,j)}{\sum_m COM(i,m)} |i-j| \quad (6.12)$$

6.5.1 Function `avd()` - Absolute Value of the Differences Method

This function returns the Absolute Value of the Differences (*AVD*) (BRAGA et al., 2011). The `avd()` function needs a co-occurrence matrix *COM* as input and calculates the types of *AVD* values shown in Table 6.2. By default, the `avd()` function returns the *AVD*₁, see (6.9), but this function accepts more parameters.

For using this function, just type any of the following commands at the prompt:

```
AVD1 = avd(COM);
% If it is necessary to get AVD2
[AVD1 AVD2] = avd(COM,2);
% If it is necessary to get all AVD types.
[AVD1 AVD2 AVD3 AVD4] = avd(COM,2,3,4);
% The order of input parameters is relative to
% the order of output values.
[AVD1 AVD4 AVD3 AVD2] = avd(COM,4,3,2);
```

Inputs:

-
- | | |
|------|--|
| COM | - is a 2D matrix, with 256 lines and 256 columns, which represents a co-occurrence matrix of a THSP matrix (section 6.2). The element $COM(a,b)$ in the co-occurrence matrix represents the quantity of times that, in two consecutive columns of a THSP matrix, the intensity values jump from $a - 1$ to $b - 1$. |
| TYPE | - [Optional] This parameter can be used many times. When it is used this parameter the function returns an additional result in the same position.
If TYPE is equal to 2, the function also returns the <i>AVD</i> ₂ , see Table 6.2.
If TYPE is equal to 3, the function also returns the <i>AVD</i> ₃ , see Table 6.2.
If TYPE is equal to 4, the function also returns the <i>AVD</i> ₄ , see Table 6.2. |
-

Outputs:

-
- | | |
|------|---|
| AVD1 | - is the value of <i>AVD</i> ₁ (Table 6.2). |
| X | - [Optional] If TYPE is equal to 2, the function also returns the <i>AVD</i> ₂ , see Table 6.2.
If TYPE is equal to 3, the function also returns the <i>AVD</i> ₃ (Table 6.2).
If TYPE is equal to 4, the function also returns the <i>AVD</i> ₄ (Table 6.2). |
-

6.6 Calculating the Regular Value of the Differences

The Regular Value of the Differences (*RVD*) method is a modification of the Inertia Moment (*IM*). The *RVD* analyzes the difference between the intensity value $I_k = i$ in time k , and the intensity $I_{k+1} = j$ in time $k + 1$ to any value of k . Thus, the *RVD* method (Equation 6.13) measures a relative jump between the consecutive images, $\mathbf{I}_k$ and $\mathbf{I}_{k+1}$.

$$RVD = \sum_{ij} M_{ij}(j - i) \quad (6.13)$$

The last equation shows that the *RVD* value is a weighted value of $(j - i)$, where the weighting factor M_{ij} is calculated from the co-occurrence matrix, COM, seen in Section 6.3. Many authors have a different way to calculate this factor. For example, Arizaga; Trivi; Rabal (1999) define M_{ij} as in Equation (6.4). On the other hand, Cardoso; Braga (2014) define M_{ij} as in Equation (6.5). This latter approach is particularly interesting since M_{ij} can be seen as the probability mass function of an intensity jump from i to j , so the weighting factor $M_{ij} \equiv Pr(i \rightarrow j)$. Known this, the *RVD* value will be the expected value of $(j - i)$, also known as first moment, $RVD = E[j - i]$. The Table 6.3 shows all types of *RVD* known in the literature.

Table 6.3: *RVD* types.

TYPE	EQUATION	DESCRIPTION
1	$RVD_1 = E[j - i]$, see Equation (6.14)	The <i>RVD</i> method with the normalized co-occurrence matrix (COM) proposed by Cardoso; Braga (2014). This can also be called as first moment of <i>RVD</i> .
2	$RVD_2 = E[(j - i)^2]$, see Equation (6.15)	The Inertia Moment proposed by Cardoso; Braga (2014). This can also be called as second moment of <i>RVD</i> .
3	$RVD_3 = Var[j - i]$, see Equation (6.16)	Knowing RVD_1 and RVD_2 it is easy to calculate the <i>RVD</i> second central moment.
4	$RVD_4 = E_{Ariz.}[j - i]$, see Equation (6.17)	The <i>RVD</i> method with the normalized co-occurrence matrix (COM) proposed by Arizaga; Trivi; Rabal (1999).

Where the *RVD* first moment (RVD_1) is:

$$E[j - i] = \sum_{ij} \frac{COM(i, j)}{\sum_{lm} COM(l, m)} (j - i) \quad (6.14)$$

The *RVD* second moment (RVD_2) is:

$$E[(j - i)^2] = \sum_{ij} \frac{COM(i, j)}{\sum_{lm} COM(l, m)} (j - i)^2 \quad (6.15)$$

The RVD second central moment (RVD_3) is:

$$Var[j - i] = E[(j - i)^2] - E[j - i]^2 \quad (6.16)$$

And the RVD first moment following Arizaga; Trivi; Rabal (1999) (RVD_4) is:

$$E_{Ariz.}[j - i] = \sum_{ij} \frac{COM(i, j)}{\sum_m COM(i, m)} (j - i) \quad (6.17)$$

6.6.1 Function `rvd()` - Regular Value of the Differences Method

This function returns the Regular Value of the Differences (RVD). The `rvd()` function needs a co-occurrence matrix `COM` as input and calculates all RVD types shown in Table 6.3. By default, the `rvd()` function returns the RVD_1 , see Equation (6.14).

For using this function, just type any of the following commands at the prompt:

```
RVD1 = rvd(COM);
% If it is necessary to get RVD2
[RVD1 RVD2] = rvd(COM,2);
% If it is necessary to get all RVD types.
[RVD1 RVD2 RVD3 RVD4] = rvd(COM,2,3,4);
% The order of input parameters is relative to
% the order of output values.
[RVD1 RVD4 RVD3 RVD2] = rvd(COM,4,3,2);
```

Inputs:

-
- | | |
|------|--|
| COM | - is a 2D matrix, with 256 lines and 256 columns, which represents a co-occurrence matrix of a THSP (Section 6.2). The element $COM(a, b)$ in the co-occurrence matrix represents the quantity of times that, in two successive columns of a THSP matrix, the intensity values jump from $a - 1$ to $b - 1$. |
| TYPE | - [Optional] This parameter can be used many times. When it is used, the function returns an additional result in the same position.
If TYPE is equal to 2, the function also returns the RVD_2 (Table 6.3).
If TYPE is equal to 3, the function also returns the RVD_3 (Table 6.3).
If TYPE is equal to 4, the function also returns the RVD_4 (Table 6.3). |
-

Outputs:

-
- | | |
|------|---|
| RVD1 | - is the value of RVD_1 , see Table 6.3. |
| X | - [Optional] If TYPE is equal to 2, the function also returns the RVD_2 (see Table 6.3).
If TYPE is equal to 3, the function also returns the RVD_3 (Table 6.3).
If TYPE is equal to 4, the function also returns the RVD_4 (Table 6.3). |
-

6.7 Numerical Analysis of the Average Difference Method

The Numerical analysis of the Average Difference (*NUMAD*) method (REIS; RABAL; BRAGA, 2016) is a modification of the average difference method (Fujii method) (FUJII; ASAKURA, 1975), where we analyze the average difference between the intensity value $I_k = i$ in time k and the intensity $I_{k+1} = j$ in time $k + 1$, for any value of k (Equation 6.18). Thus, this method measures the relative contrast of the intensity jumps between the consecutive images $\mathbf{I}_k$ and $\mathbf{I}_{k+1}$.

$$NUMAD = \sum_{ij} M_{ij} \frac{|i-j|}{i+j} \quad (6.18)$$

The last equation shows that the *NUMAD* value is a weighted value of $|i-j|/(i+j)$, where the weighting factor M_{ij} is calculated from the co-occurrence matrix *COM*, as seen in section 6.3. The *NUMAD* value is the expected value of $|i-j|/(i+j)$, also known as first moment of $|i-j|/(i+j)$, so that $NUMAD = E[|i-j|/(i+j)]$. The Table 6.4 shows the types of *NUMAD* values known in the literature (REIS; RABAL; BRAGA, 2016).

Table 6.4: Types of numerical analysis of the average difference.

TYPE	EQUATION	DESCRIPTION
1	$NUMAD_1 = E \left[\frac{ i-j }{i+j} \right]$ (Equation 6.19)	The <i>NUMAD</i> outcome with the normalized co-occurrence matrix (<i>COM</i>) proposed by Cardoso; Braga (2014). This can also be called as first moment of <i>NUMAD</i> .
2	$NUMAD_2 = E \left[\left(\frac{ i-j }{i+j} \right)^2 \right]$ (Equation 6.20)	This outcome is the second moment of <i>NUMAD</i> .

Where the first moment ($NUMAD_1$) is:

$$E \left[\frac{|i-j|}{i+j} \right] = \sum_{ij} \frac{COM(i,j)}{\sum_{lm} COM(l,m)} \frac{|i-j|}{i+j} \quad (6.19)$$

And the second moment ($NUMAD_2$) is:

$$E \left[\left(\frac{|i-j|}{i+j} \right)^2 \right] = \sum_{ij} \frac{COM(i,j)}{\sum_{lm} COM(l,m)} \left(\frac{|i-j|}{i+j} \right)^2 \quad (6.20)$$

6.7.1 Function numad() - Numerical Analysis of the Average Difference Method

This function returns a numerical analysis of the average difference method (REIS; RABAL; BRAGA, 2016). The `numad()` function needs as input a co-occurrence matrix *COM* and presents the values shown in Table 6.4. By default, the `numad()` function returns the $NUMAD_1$ value (Equation 6.19), but this function also accepts another parameter for requesting the $NUMAD_2$ value.

To use this function just type any of the following commands at the prompt:

```
NUMAD1 = numad(COM);
```

```
% If it is necessary to get NUMAD2
[NUMAD1 NUMAD2] = numad(COM,2);
```

Inputs:

-
- COM - is a 2D matrix, with 256 lines and 256 columns, which represents a co-occurrence matrix of a THSP matrix (Section 6.2). The element $COM(a,b)$ in the co-occurrence matrix represents the quantity of times that, in two consecutive columns of a THSP matrix, the intensity values jump from $a - 1$ to $b - 1$.
- TYPE - **[Optional]** When this parameter is used the function returns an additional result in the second position.
If TYPE is equal to 2, the function also returns the $NUMAD_2$ (Table 6.4). Otherwise, an error occurs.
-

Outputs:

-
- NUMAD1 - is the value of $NUMAD_1$, see Table 6.4.
- NUMAD2 - **[Optional]** If TYPE is equal to 2, the function also returns the $NUMAD_2$ (Table 6.4).
-

6.8 Calculating the Correlation in the Speckle Pattern

Two types of correlations used in the analysis of the biospeckle images can be found in the literature. We rename them as correlation *Type 1* and *Type 2* as seen in the Table 6.5.

Table 6.5: Types of correlation coefficients

TYPE	DESCRIPTION
1	The correlation coefficient, used by Xu; Jonathan; Khorana (1995) (Equation 6.21).
2	The Pearson correlation coefficient, used by Zdunek et al. (2007) (Equation 6.22).

The Equation (6.21) shows the correlation coefficient (*Type 1*) of **a** and **b**, including the continuous part of signal,

$$corr(\mathbf{a}, \mathbf{b}) \equiv \frac{\langle \mathbf{ab} \rangle}{\sqrt{\langle \mathbf{a}^2 \rangle \langle \mathbf{b}^2 \rangle}}. \quad (6.21)$$

The Equation (6.22) shows the correlation coefficient (*Type 2*) of **a** and **b**, excluding the continuous part of signal,

$$corr(\mathbf{a}, \mathbf{b}) \equiv \frac{\langle (\mathbf{a} - \mu_a)(\mathbf{b} - \mu_b) \rangle}{\sigma_a \sigma_b}. \quad (6.22)$$

Where the operator $\langle . \rangle$ is the spatial expected operator; and the values $\mu_a = \langle \mathbf{a} \rangle$ and $\mu_b = \langle \mathbf{b} \rangle$ are the spatial mean values of matrices or vectors $\mathbf{a}$ and $\mathbf{b}$. In addition, $\sigma_a = \sqrt{\langle (\mathbf{a} - \mu_a)^2 \rangle}$ and $\sigma_b = \sqrt{\langle (\mathbf{b} - \mu_b)^2 \rangle}$ are the spatial standard deviation values of $\mathbf{a}$ and $\mathbf{b}$.

6.8.1 Function `thsp2corr()` - Time History Speckle Pattern to Correlation Coefficient

This function calculates the space-time speckle correlation technique (XU; JOENATHAN; KHORANA, 1995). This calculates the correlation between the pixels analyzed in the time history speckle pattern. Thus, we use, as input data, the matrix `THSP` of M lines and $NTIMES$ columns, which represents the intensity time evolution of M pixels in $NTIMES$ samples. So that, the correlation C_{il} (Equation 6.23) is calculated between all pixels $THSP(:, i)$ in the instant i and the pixels $THSP(:, i + j)$ in the instant $i + j$,

$$C_{il} = \text{corr}(THSP(:, i), THSP(:, i + l)). \quad (6.23)$$

The function finally returns, C_l (Equation 6.24), the mean value of this correlations for a lag of l ,

$$C_l = \frac{1}{NTIMES/2} \sum_{i=1}^{NTIMES/2} C_{il}. \quad (6.24)$$

We can use two types of correlation functions `corr()`, as shown in Table 6.5.

For using this function, just type any of the following commands at the prompt:

```
C      = thsp2corr(THSP);
C      = thsp2corr(THSP, TYPE);
[C L] = thsp2corr(THSP);
[C L] = thsp2corr(THSP, TYPE);
% Getting the vector C, with the Pearson correlation coefficients,
% and the sample lag vector (L).
[C L] = thsp2corr(THSP, 2);
```

Inputs:

THSP - is an integer 2D matrix that represents the time history speckle pattern. This matrix can be obtained using the function `thsp()`.

TYPE - **[Optional]**
 If **TYPE** is equal to 1, the function uses the *Type 1* correlation, see Table 6.5.
 If **TYPE** is equal to 2, the function uses the *Type 2* correlation, see Table 6.5.
 By default **TYPE** is equal to 1.

Outputs:

C - is the correlation vector, which corresponds to the C_l values.

L - **[Optional]** is the vector with the values of index l in the vector **C**.

6.8.2 Function `stscorr()` - Spatial-Temporal Speckle Correlation

This function calculates the spatial-temporal speckle correlation technique (ZDUNEK et al., 2007), i.e. it calculates the correlation between one image and its consecutive. The function `stscorr()` also calculates the correlation with the previous images and uses, as input data, a 3D matrix (DATA) of intensity matrices $\mathbf{I}_k \equiv \text{DATA}(:, :, k)$, $1 \leq k \leq \text{NTIMES}$. Thus, the correlation $C_{k_0}^{l\tau}$ (Equation 6.25) is calculated between the image $\mathbf{I}_{k_0}$ and the image $\mathbf{I}_{k_0+l}$, $\forall 1 \leq k_0 + l \leq \text{NTIMES}$. Here it is assumed that the images were taken with sampling rate equal to τ .

$$C_{k_0}^{l\tau} = \text{corr}(\mathbf{I}_{k_0}, \mathbf{I}_{k_0+l}), \quad (6.25)$$

where `corr()` function is the correlation *Type 2* seen in the Table 6.5.

To use this function just type any of the following commands at the prompt:

```
C          = stscorr(DATA, Tau, K0)
[C L Tau]  = stscorr(DATA, Tau, K0)
[C L Tau L] = stscorr(DATA, Tau, K0)
```

Inputs:

DATA	-	is the speckle data, where DATA is a 3D matrix created by grouping NTIMES matrices with NLIN lines and NCOL columns. When <code>N=size(DATA)</code> , then N(1,1) represents NLIN, N(1,2) represents NCOL and N(1,3) represents NTIMES.
Tau	-	is the sampling rate in seconds.
K0	-	is the number of the reference frame used in correlation analysis.

Outputs:

C	-	is the correlation vector. It corresponds to the $C_{k_0}^{l\tau}$ values (ZDUNEK et al., 2007), with the negative values of $l\tau$ being calculated.
L Tau	-	[Optional] is the vector with the values of time $l \times \tau$ in the vector C. It can have negative values.
L	-	[Optional] is the vector with the values of index l in the vector C. It can have negative values.

6.9 Graphical Methods

6.9.1 Function `fujii()` - Fujii Method

This function implements the Fujii technique also called Average Difference (FUJII et al., 1987; FUJII; ASAKURA, 1975). It uses, as input data, a 3D matrix (DATA) created by grouping intensity

matrices $\mathbf{I}_k \equiv \text{DATA}(:, :, k)$, $1 \leq k \leq \text{NTIMES}$, as can be seen in the Equation (6.26). Thus, the matrix operations are made element by element,

$$FUJII = \sum_{k=1}^{\text{NTIMES}-1} \frac{|\mathbf{I}_k - \mathbf{I}_{k+1}|}{\mathbf{I}_k + \mathbf{I}_{k+1} + \text{eps}}. \quad (6.26)$$

The original Fujii technique does not have the *eps* term. In Octave/Matlab *eps* is the distance from 1.0 to the next largest double-precision number ($\text{eps} = 2^{-52}$).

The function in Equation (6.26) is normalized to Equation (6.27):

$$Y = FUJII \frac{2 \times 100}{\text{NTIMES} - 1} \quad (6.27)$$

Where $(\text{NTIMES} - 1)$ is the number of elements in the sum, 2 is a factor to turn $\frac{\mathbf{I}_k + \mathbf{I}_{k+1}}{2}$ a mean value and 100 is a percentage factor. Thus, the Y matrix (Equation 6.28) represents the expected percentage value of the absolute difference $|\mathbf{I}_k - \mathbf{I}_{k+1}|$ relative to the mean value $\frac{\mathbf{I}_k + \mathbf{I}_{k+1}}{2}$ for any two consecutive values. In other words Y is the average percentage change of $|\mathbf{I}_k - \mathbf{I}_{k+1}|$.

$$Y = 100 E \left[\frac{|\mathbf{I}_k - \mathbf{I}_{k+1}|}{\frac{\mathbf{I}_k + \mathbf{I}_{k+1}}{2}} \right] \quad (6.28)$$

For using this function, just type any of the following commands at the prompt:

```
Y = fujii(DATA);
% Disabling the graphical output.
Y = fujii(DATA, 'off');
```

This function also shows a figure with Fujii outcome.

Inputs:

DATA - is the speckle data, where **DATA** is a 3D matrix created by grouping **NTIMES** matrices with **NLIN** lines and **NCOL** columns. When **N=size(DATA)**, then **N(1,1)** represents **NLIN**, **N(1,2)** represents **NCOL** and **N(1,3)** represents **NTIMES**.

SHOW **[Optional]** If **SHOW** is equal to string **'off'**, then the result is not plotted. By default, **SHOW='on'**.

Output:

Y - Returns the Fujii matrix.

6.9.2 Function `gendiff()` - Generalized Differences Method

This function implements the Generalized Difference (*GD*) technique (Equation 6.29) (ARIZAGA et al., 2002). It uses, as input data, a 3D matrix (*DATA*) created by grouping intensity matrices $\mathbf{I}_k \equiv \text{DATA}(:, :, k)$, $1 \leq k \leq \text{NTIMES} \equiv M$.

$$GD = \sum_{k=1}^{M-1} \sum_{l=1}^{M-k} |\mathbf{I}_k - \mathbf{I}_{k+l}| \quad (6.29)$$

The function is normalized to *Y* (Equation 6.30) with the number of elements in the sum.

$$Y = \frac{GD}{\binom{M}{2}} \quad (6.30)$$

Where $\binom{M}{2}$ is the binomial coefficient of *M* and 2. This is the number of combinations of *M* items taken 2 at a time. Thus *Y* matrix represents the expected value of absolute difference $|\mathbf{I}_{k1} - \mathbf{I}_{k2}|$ (Equation 6.31) for any two different *k1* and *k2* values.

$$Y = E[|\mathbf{I}_{k1} - \mathbf{I}_{k2}|] \quad (6.31)$$

For using this function, just type any of the following commands at the prompt:

```
Y = gendiff(DATA);  
% Disabling the graphical output.  
Y = gendiff(DATA, 'off');
```

This function also shows a figure with *GD* outcome.

Inputs:

DATA - is the speckle data, where *DATA* is a 3D matrix created by grouping *NTIMES* matrices with *NLIN* lines and *NCOL* columns. When *N=size(DATA)*, then
N(1,1) represents *NLIN*,
N(1,2) represents *NCOL* and
N(1,3) represents *NTIMES*.

SHOW **[Optional]** If *SHOW* is equal to string 'off' then the result is not plotted. By default, *SHOW*='on'.

Output:

Y - Returns the Generalized Difference matrix.

6.9.3 Function `stdcont()` - Contrast - Deviation - Mean Method

This function implements the temporal speckle contrast image (NOTHDURFT; YAO, 2005). It uses, as input data, a 3D matrix (*DATA*) created by grouping intensity matrices $\mathbf{I}_k \equiv \text{DATA}(:, :, k)$, $1 \leq k \leq \text{NTIMES}$.

To get the temporal speckle contrast image, first it is necessary to get the temporal speckle mean matrix (Equation 6.32), μ , as

$$\mu = E[\mathbf{I}_k] = \frac{1}{\text{NTIMES}} \sum_{k=1}^{\text{NTIMES}} \mathbf{I}_k, \quad (6.32)$$

Then the temporal speckle standard deviation (Equation 6.33), σ , is calculates as

$$\sigma = \sqrt{E[(\mathbf{I}_k - \mu)^2]} = \sqrt{\frac{1}{\text{NTIMES}} \sum_{k=1}^{\text{NTIMES}} (\mathbf{I}_k - E[\mathbf{I}_k])^2}. \quad (6.33)$$

We use here the population standard deviation. Finally, the temporal speckle contrast image can be calculated using the Equation (6.34),

$$\text{Contrast} = \frac{\sigma}{\mu} \quad (6.34)$$

For using this function, just type any of the following commands at the prompt:

```
% Returns the contrast matrix in C.
C      = stdcont(DATA);

% Returns the contrast matrix in C and the standard deviation matrix in D.
[C D]  = stdcont(DATA);

% Returns the same as above and the mean matrix in E.
[C D E] = stdcont(DATA);

% Disabling the graphical output.
C = stdcont(DATA, 'off');
[C D] = stdcont(DATA, 'off');
[C D E] = stdcont(DATA, 'off');
```

The function returns the temporal speckle contrast matrix, and additionally returns the temporal speckle standard deviation and the temporal speckle mean matrix following the Equations (6.32), (6.33) and (6.34).

Inputs:

DATA	- is the speckle data, where DATA is a 3D matrix created by grouping NTIMES matrices with NLIN lines and NCOL columns. When $N = \text{size}(\text{DATA})$, then $N(1,1)$ represents NLIN, $N(1,2)$ represents NCOL and $N(1,3)$ represents NTIMES.
SHOW	[Optional] If SHOW is equal to string 'off', then the result is not plotted. By default, SHOW='on'.

Outputs:

-
- C - Returns the temporal speckle contrast matrix of image datapack.
 - D - **[Optional]** Returns the temporal speckle standard deviation matrix of image datapack.
 - E - **[Optional]** Returns the temporal speckle mean matrix of image datapack.
-

6.9.4 Function `graphim()` - Inertia Moment Method in Graphic Mode

This function implements the Inertia Moment (*IM*) (ARIZAGA; TRIVI; RABAL, 1999) method to each pixel of the 3D matrix, with the co-occurrence normalization proposed by Cardoso; Braga (2014). We named it as Graphic Inertia Moment (*GIM*) method (Equation 6.35). The function `graphim()` uses, as input data, a 3D matrix (*DATA*) created by grouping intensity matrices $\mathbf{I}_k \equiv \text{DATA}(:, :, k)$, $1 \leq k \leq \text{NTIMES}$.

$$GIM = E[(\mathbf{I}_k - \mathbf{I}_{k+1})^2] = \frac{1}{\text{NTIMES} - 1} \sum_{k=1}^{\text{NTIMES}-1} (\mathbf{I}_k - \mathbf{I}_{k+1})^2 \quad (6.35)$$

For using this function, just type any of the following commands at the prompt:

```
GIM = graphim(DATA);

% Disabling the graphical output.
GIM = graphim(DATA, 'off');
```

This function also shows a figure with the *GIM* method outcome.

Inputs:

-
- DATA - is the speckle data, where *DATA* is a 3D matrix created by grouping *NTIMES* matrices with *NLIN* lines and *NCOL* columns. When $N = \text{size}(\text{DATA})$, then $N(1,1)$ represents *NLIN*, $N(1,2)$ represents *NCOL* and $N(1,3)$ represents *NTIMES*.
 - SHOW **[Optional]** If *SHOW* is equal to string 'off' then the result is not plotted. By default, *SHOW*='on'.
-

Output:

-
- GIM - Returns a matrix with the inertia moment in graphic mode.
-

6.9.5 Function `graphavd()` - AVD Method in Graphic Mode

This function implements the Absolute Value of the Differences (*AVD*) (BRAGA et al., 2011) method to each pixel of the 3D matrix, with the co-occurrence normalization proposed by Cardoso; Braga (2014). We named it as Graphic *AVD* (*GAVD*) method (Equation 6.36). The function

`graphavd()` uses, as input data, a 3D matrix (DATA) created by grouping intensity matrices $\mathbf{I}_k \equiv \text{DATA}(:, :, k)$, $1 \leq k \leq \text{NTIMES}$.

$$GAVD = E[|\mathbf{I}_k - \mathbf{I}_{k+1}|] = \frac{1}{\text{NTIMES} - 1} \sum_{k=1}^{\text{NTIMES}-1} |\mathbf{I}_k - \mathbf{I}_{k+1}| \quad (6.36)$$

For using this function, just type the following command at the prompt:

```
GAVD = graphavd(DATA);

% Disabling the graphical output.
GAVD = graphavd(DATA, 'off');
```

This function also shows a figure with the *GAVD* method outcome.

Inputs:

DATA - is the speckle data, where DATA is a 3D matrix created by grouping NTIMES matrices with NLIN lines and NCOL columns. When $N = \text{size}(\text{DATA})$, then $N(1, 1)$ represents NLIN, $N(1, 2)$ represents NCOL and $N(1, 3)$ represents NTIMES.

SHOW **[Optional]** If SHOW is equal to string 'off' then the result is not plotted. By default, SHOW='on'.

Output:

GAVD - Returns a matrix with the *AVD* in graphic mode.

6.9.6 Function graphrvd() - RVD Method in Graphic Mode

This function implements the Regular Value of the Differences (*RVD*) method to each pixel of the 3D matrix, with the co-occurrence normalization proposed by Cardoso; Braga (2014). We named it as Graphic *RVD* (*GRVD*) method (Equation 6.37). The function `graphrvd()` uses, as input data, a 3D matrix (DATA) created by grouping intensity matrices $\mathbf{I}_k \equiv \text{DATA}(:, :, k)$, $1 \leq k \leq \text{NTIMES}$.

$$GRVD = E[\mathbf{I}_{k+1} - \mathbf{I}_k] = \frac{1}{\text{NTIMES} - 1} \sum_{k=1}^{\text{NTIMES}-1} (\mathbf{I}_{k+1} - \mathbf{I}_k) \quad (6.37)$$

For using this function, just type any of the following commands at the prompt:

```
GRVD = graphrvd(DATA);

% Disabling the graphical output.
GRVD = graphrvd(DATA, 'off');
```

This function also shows a figure with the *GRVD* method outcome.

Inputs:

DATA	-	is the speckle data. Where DATA is a 3D matrix created by grouping NTIMES matrices with NLIN lines and NCOL columns. When $N = \text{size}(\text{DATA})$, then $N(1,1)$ represents NLIN, $N(1,2)$ represents NCOL and $N(1,3)$ represents NTIMES.
SHOW		[Optional] If SHOW is equal to string 'off' then the result is not plotted. By default, SHOW='on'.

Output:

GRVD	-	Returns a matrix with the RVD in graphic mode.
------	---	--

6.9.7 Function graphptd() - Parametrized form of Temporal Difference Method

This function implements the Parameterized form of Temporal Difference (*PTD*) (MINZ; NIRALA, 2014) method. The function *graphptd()* returns a normalized matrix with the values of *PTD* method (Equation 6.38). It uses, as input data, a 3D matrix (DATA) created by grouping intensity matrices $\mathbf{I}_k \equiv \text{DATA}(:, :, k)$, $1 \leq k \leq \text{NTIMES}$.

Thus, the *PTD* method is shown in the next equation

$$PTD = \sum_{k=1}^{\text{NTIMES}-1} |\mathbf{I}_k - \mathbf{I}_{k+1}|^p. \quad (6.38)$$

The last expression is normalized to Equation (6.39) with the number of elements in the sum

$$\text{GPTD} = E[|\mathbf{I}_k - \mathbf{I}_{k+1}|^p] = \frac{PTD}{\text{NTIMES} - 1}. \quad (6.39)$$

So that, GPTD matrix represents the expected value of the absolute difference $|I_k - I_{k+1}|^p$ for any k value.

For using this function, just type any of the following commands at the prompt:

```
% Enabling the graphical output by default.
GPTD = graphptd(DATA);

% Enabling the graphical output explicitly.
GPTD = graphptd(DATA, 'on');

% Disabling the graphical output.
GPTD = graphptd(DATA, 'off');
```

This function also shows a figure with the *GPTD* outcome.

Inputs:

DATA	-	is the speckle data, where DATA is a 3D matrix created by grouping NTIMES matrices with NLIN lines and NCOL columns. When $N = \text{size}(\text{DATA})$, then $N(1,1)$ represents NLIN, $N(1,2)$ represents NCOL and $N(1,3)$ represents NTIMES.
SHOW		[Optional] If SHOW is equal to string 'off' then the result is not plotted. By default, SHOW='on'.

Output:

GPTD	-	Returns a matrix with the result of <i>PTD</i> method.
------	---	--

6.9.8 Function graphkurt() - Temporal Speckle Kurtosis Matrix

This function implements the temporal speckle kurtosis (SHESKIN, 2003) matrix. It uses, as input data, a 3D matrix (DATA) created by grouping intensity matrices $\mathbf{I}_k \equiv \text{DATA}(:, :, k)$, $1 \leq k \leq \text{NTIMES}$.

In order to get the temporal speckle kurtosis matrix, first it is necessary to get the temporal speckle mean matrix, μ , and the temporal speckle standard deviation matrix, σ , as presented in Equations (6.32) and (6.33). Thus, the temporal speckle kurtosis matrix is calculated as the Equation (6.40),

$$K = E \left[\left(\frac{\mathbf{I}_k - \mu}{\sigma} \right)^4 \right]. \quad (6.40)$$

For using this function, just type any of the following commands at the prompt:

```
% Returns the kurtosis matrix in K.
K      = graphkurt(DATA);

% Returns the kurtosis matrix in K and the standard deviation matrix in D.
[K D]  = graphkurt(DATA);

% Returns the same as above and the mean matrix in E.
[K D E] = graphkurt(DATA);

% Disabling the graphical output.
[K D E] = graphkurt(DATA, 'off');
```

The function returns the temporal speckle kurtosis matrix and additionally returns the temporal speckle standard deviation matrix and temporal speckle mean matrix following the Equations (6.32), (6.33) and (6.40).

Inputs:

DATA	-	is the speckle data, where DATA is a 3D matrix created by grouping matrices with NLIN lines and NCOL columns. When $N = \text{size}(\text{DATA})$, then $N(1,1)$ represents NLIN, $N(1,2)$ represents NCOL and $N(1,3)$ represents NTIMES.
SHOW		[Optional] If SHOW is equal to string 'off' then the result is not plotted. By default, SHOW='on'.

Outputs:

K	-	Returns the temporal speckle kurtosis matrix of image datapack.
D	-	[Optional] Returns the temporal speckle standard deviation matrix of image datapack.
E	-	[Optional] Returns the temporal speckle mean matrix of image datapack.

6.9.9 Function `graphskew()` - Temporal Speckle Skewness Matrix

This function implements the temporal speckle skewness (SHESKIN, 2003) matrix. It uses, as input data, a 3D matrix (DATA) created by grouping intensity matrices $\mathbf{I}_k \equiv \text{DATA}(:, :, k)$, $1 \leq k \leq \text{NTIMES}$.

In order to get the temporal speckle skewness matrix, first it is necessary to get the temporal speckle mean matrix, μ , and the temporal speckle standard deviation, σ , as presented in Equations (6.32) and (6.33). Thus, the temporal speckle skewness matrix is calculated as the Equation (6.41),

$$S = E \left[\left(\frac{\mathbf{I}_k - \mu}{\sigma} \right)^3 \right]. \quad (6.41)$$

For using this function, just type any of the following commands at the prompt:

```
% Returns the skewness matrix in S.
S      = graphskew(DATA);

% Returns the skewness matrix in S and the standard deviation matrix in D.
[S D]  = graphskew(DATA);

% Returns the same as above and the mean matrix in E.
[S D E] = graphskew(DATA);

% Disabling the graphical output.
[S D E] = graphskew(DATA, 'off');
```

The function returns the temporal speckle skewness matrix and additionally returns the temporal speckle standard deviation matrix and the temporal speckle mean matrix following the Equations (6.32), (6.33) and (6.40).

Inputs:

-
- DATA - is the speckle data, where DATA is a 3D matrix created by grouping NTIMES matrices with NLIN lines and NCOL columns. When $N = \text{size}(\text{DATA})$, then $N(1,1)$ represents NLIN, $N(1,2)$ represents NCOL and $N(1,3)$ represents NTIMES.
- SHOW **[Optional]** If SHOW is equal to string 'off' then the result is not plotted. By default, SHOW='on'.
-

Outputs:

-
- S - Returns a matrix that represents the temporal speckle skewness of image datapack.
- D - **[Optional]** Returns a matrix that represents the temporal speckle standard deviation of image datapack.
- E - **[Optional]** Returns a matrix that represents the temporal speckle mean of image datapack.
-

6.9.10 Function graphmhi() - Motion History Image

This function implements the Motion History Image (*MHI*) technique (GODINHO et al., 2012; DAVIS, 2001). The function considers the activity of a pixel, which should have an absolute intensity jump superior to U . It uses, as input data, a 3D matrix (DATA) created by grouping intensity matrices $\mathbf{I}_k \equiv \text{DATA}(:, :, k)$, $1 \leq k \leq \text{NTIMES}$.

The calculus of MHI matrix, that is the result of applying the *MHI* technique, can be seen in the Equations (6.42), (6.43), (6.44) and (6.45).

$$S_k = \mathbf{I}_k - \mathbf{I}_{k-1}. \quad (6.42)$$

$$T_k(i, j) = \begin{cases} 1 & \text{if } |S_k(i, j)| > U \\ 0 & \text{if } |S_k(i, j)| \leq U \end{cases} \quad (6.43)$$

$$\text{MHI} = 255 \sum_{i=0}^{\text{NTIMES}-2} T_{\text{NTIMES}-i} h_i, \quad (6.44)$$

being that

$$h_i = \frac{\text{NTIMES} - 1 - i}{M} \quad (6.45)$$

and $M = \text{NTIMES}(\text{NTIMES} - 1)/2$.

For using this function, just type any of the following commands at the prompt:

```
MHI=graphmhi(DATA,U);
% Disabling the graphical output.
MHI=graphmhi(DATA,U,'off');
```

Inputs:

-
- DATA** - is the speckle data, where DATA is a 3D matrix created by grouping NTIMES matrices with NLIN lines and NCOL columns. When $N = \text{size}(\text{DATA})$, then $N(1,1)$ represents NLIN, $N(1,2)$ represents NCOL and $N(1,3)$ represents NTIMES.
- U** - is the activity threshold. Only the changes of intensity larger than U. $0 \leq U \leq 255$ are considered an activity
- SHOW** **[Optional]** If SHOW is equal to string 'off' then the result is not plotted. By default, SHOW='on'.
-

Output:

-
- MHI** - is the motion history image matrix of datapack. The elements with highest gray values in the matrix address to most recent activities.
-

6.10 Frequency Analysis

6.10.1 Function datapack_conv() - Convolution of Datapack

This function evaluates the convolution for each pixel of an input datapack. The function uses as input a datapack DATA, with NTIMES images, and a vector $H = [h_0, h_1, h_2, \dots, h_{M-1}]$ of M elements; the vector represents the impulse response function $h[n] \equiv h_{n+M_2}$ seen in the Figure 6.5, so that the Z transform, $H(Z)$, of $h[n]$ is equal to Equation (6.46),

$$H(Z) = h_0 Z^{+M_2} + h_1 Z^{+M_2-1} + \dots + h_{M_2} + \dots + h_{M-1} Z^{+M_2-(M-1)}, \quad (6.46)$$

where $M_2 = \lfloor M/2 \rfloor$.

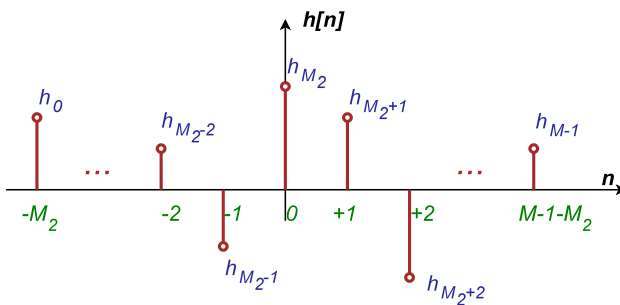


Figure 6.5: Impulse response function $h[n]$.

Thus, the elements of H vector represent an impulse response function, $h[n]$, with the value h_{M_2}

in the instant $n = 0$, and value 0 for $n < -M_2$ or $M - 1 - M_2 < n$. The convolution of datapack DATA and the vector H return in the datapack DATAOUT, and the operation is applied in the scope of DATA, so that the size of DATAOUT is equal to the size of DATA.

Thus, the datapacks DATA and DATAOUT are 3D matrices created by grouping intensity matrices $\mathbf{I}_k \equiv \text{DATA}(:, :, k)$ and $\mathbf{O}_k \equiv \text{DATAOUT}(:, :, k)$ respectively, for $1 \leq k \leq \text{NTIMES}$ (Equation 6.47). The convolution aforementioned can be expressed as

$$\mathbf{O}_k = \sum_{i=1}^{\text{NTIMES}} \mathbf{I}_i h[k-i]. \quad (6.47)$$

For using this function, just type the following command at the prompt:

```
DATAOUT = datapack_conv(DATA,H)
```

Inputs:

-
- DATA - is the speckle data, where DATA is a 3D matrix created by grouping NTIMES matrices with NLIN lines and NCOL columns. When $N = \text{size}(\text{DATA})$, then $N(1,1)$ represents NLIN, $N(1,2)$ represents NCOL and $N(1,3)$ represents NTIMES.
- H - is a vector with M elements, which represents the impulse response function $h[n]$, while the $M_2 = \lfloor M/2 \rfloor$ element represents the $h[0]$ value.
-

Output:

-
- DATAOUT - is a datapack, where DATAOUT is a 3D matrix with the same size of DATA and is the result of the convolution of DATA and H.
-

6.10.2 Function firfilterbank() - Step of a FIR Filter Bank

This function evaluates, for each pixel of a datapack DATA, a step of a filter bank. The **firfilterbank()** function uses two non-causal *FIR* filters, H0 and H1, with order $M - 1$. Thus, if the vector $\mathbf{H0} = [h_0, h_1, h_2, \dots, h_{M-1}]$ represents the impulse response $h_0[n] \equiv h_{n+M_2}$, then its Z transform is represented in the Equation (6.48),

$$H_0(Z) = h_0 Z^{+M_2} + h_1 Z^{+M_2-1} + \dots + h_{M_2} + \dots + h_{M-1} Z^{+M_2-(M-1)}, \quad (6.48)$$

where the value h_{M_2} represents the value in the instant $n = 0$ and $M_2 = \lfloor M/2 \rfloor$. Similarly, the vector H1 follows the same formation rule of vector H0.

The **firfilterbank()** function implements two modes (Figure 6.6), as seen in the Table 6.6.

For using this function, just type any of the following commands at the prompt:

```
% Generalized way of to use the function
[DATAO DATA1] = firfilterbank(DATA,FILTER,MODE);
```

Table 6.6: Use modes of the *FIR* filter bank.

TYPE	DESCRIPTION
'MODE0'	This mode is composed of two non-causal <i>FIR</i> filters, represented by H_0 and H_1 as seen in the Figure 6.6a. Thus, it can accept as input parameter a H_0 filter or, the H_0 and H_1 filters. If only the H_0 filter is delivered as input parameter, then the H_1 filter is calculated as the complement filter of H_0, so that $H_1(Z) = 1 - H_0(Z)$.
'MODE2'	This mode uses two non-causal <i>FIR</i> filters, H_0 and H_1, as seen in the Figure 6.6b. The difference with the mode 'MODE0' is that it only can accept H_0 as input parameter and should have an even number of elements, M. The H_1 filter will be calculated as the quadrature mirror filter (AGRAWAL; SAHU, 2013) of H_0; so that $H_1(Z) = H_0(-Z)$. Finally after the filters, H_0 and H_1, there are down-samplers by 2. Thus, this mode is commonly used with a low pass H_0 non-causal <i>FIR</i> filter with cut-off in $F_s/4$, being F_s the sampling frequency. For perfect reconstruction, it is necessary that $H_0^2(Z) - H_0^2(-Z) \equiv AZ^B$ for any A and B real.

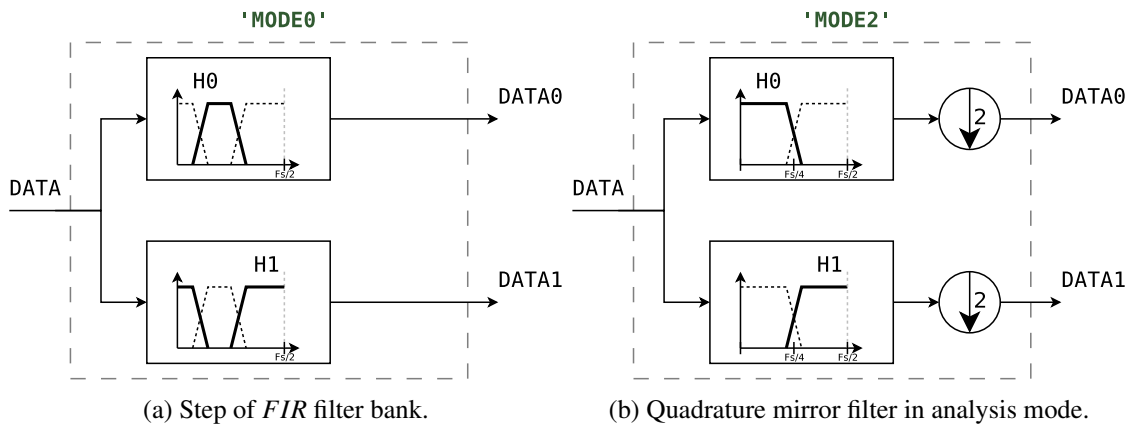


Figure 6.6: Decomposition scheme according to the used mode.

```

% Using the MODE2 with the filter H0.
% H0 can be calculated using the qmfmaker function.
[DATA0 DATA1] = firfilterbank(DATA,H0,'MODE2');

% Using the MODE0 with the filter H0
% and the filter H1 calculated automatically.
% H0 can be calculated using the get_fir_filter function.
[DATA0 DATA1] = firfilterbank(DATA,H0,'MODE0');

% Using the MODE0 with the filter H0
% and the filter H1 delivered explicitly.
% H0 and H1 can be calculated using the get_fir_filter function.
[DATA0 DATA1] = firfilterbank(DATA,[H0;H1],'MODE0');
```

Inputs:

DATA	-	is the speckle data, where DATA is a 3D matrix created by grouping NTIMES matrices with NLIN lines and NCOL columns. When $N = \text{size}(\text{DATA})$, then $N(1,1)$ represents NLIN, $N(1,2)$ represents NCOL and $N(1,3)$ represents NTIMES.
FILTER	-	is a matrix (with 2 lines) or a vector, where $\text{FILTER} = \text{H0}$ or $\text{FILTER} = [\text{H0}; \text{H1}]$. H0 and H1 represent two FIR filters. $\text{FILTER} = \text{H0}$ is used in the mode 'MODE2', while $\text{FILTER} = \text{H0}$ or $\text{FILTER} = [\text{H0}; \text{H1}]$ can be used in the mode 'MODE0'.
MODE	-	is the mode type used in the step filter bank. It can be 'MODE0' or 'MODE2'.

Outputs:

DATA0	-	is obtained after the output of H0 filter, when each pixel of datapack DATA is sent as input. If the used mode has a down-sampler after the filter, then DATA0 is the output of the down-sampler.
DATA1	-	is obtained after the output of H1 filter, when each pixel of datapack DATA is sent as input. If the used mode has a down-sampler after the filter, then DATA1 is the output of the down-sampler.

6.10.3 Function `firsynthesisbank()` - Step of a FIR Filter Bank in Synthesis Mode

This function makes a step of a synthesis filter bank (AGRAWAL; SAHU, 2013). The `firsynthesisbank()` function has two inputs, DATA0 and DATA1, and one output, DATA. If the input datapacks have NTIMES samples, then the output has 2NTIMES samples.

Internally, after each one of the function inputs there exists one up-sampler with gains of 2 and non-causal FIR filters ($G0$ and $G1$) of order $M - 1$, as can be seen in the Figure 6.7.

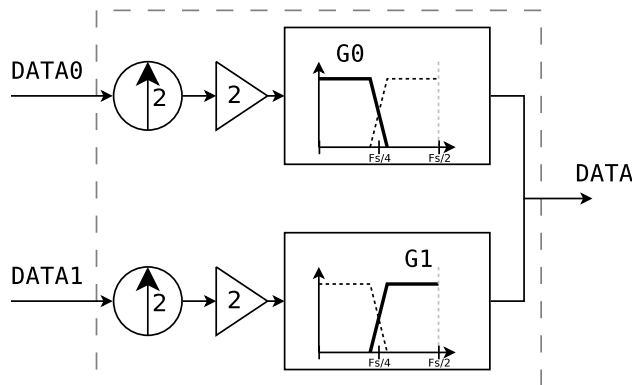


Figure 6.7: Quadrature mirror filter in synthesis mode.

The function accepts a H0 filter as input parameter, so that the filters $G0$ and $G1$ are calculated as its function of this. The H0 filter must be a low-pass FIR filter with cut-off at $F_s/4$, being F_s the sampling frequency.

If $H_0 = [h_0, h_1, h_2, \dots, h_{M-1}]$, then its Z transform is expressed by the Equation (6.49),

$$H_0(Z) = h_0 Z^{+M_2} + h_1 Z^{+M_2-1} + \dots + h_{M_2} + \dots + h_{M-1} Z^{+M_2-(M-1)}, \quad (6.49)$$

where the value h_{M_2} represents the value in the instant $n = 0$ and $M_2 = \lfloor M/2 \rfloor$. Thus, the filters G0 and G1 are calculated as $G_0(Z) = H_0(Z)$ and $G_1(Z) = -H_0(-Z)$.

The input datapacks, DATA0 and DATA1 are the outputs of a quadrature mirror filter in analysis mode, as in the function `firfilterbank()` in `MODE='MODE2'`. Thus, in order to have a perfect reconstruction of inputs it is necessary that the H0 filter fulfills the following $H_0^2(Z) - H_0^2(-Z) \equiv AZ^B$ for any real A and B.

For using this function, just type the following command at the prompt:

```
DATA=firsynthesisbank(DATA0,DATA1,H0);
```

Inputs:

-
- | | |
|-------|--|
| DATA0 | - is a speckle datapack, where DATA0 is a 3D matrix created by grouping NTIMES matrices with NLIN lines and NCOL columns. When <code>N=size(DATA0)</code> , then <code>N(1,1)</code> represents NLIN, <code>N(1,2)</code> represents NCOL and <code>N(1,3)</code> represents NTIMES. DATA0 must be obtained after the down-sampler in the branch of a low-pass H0 non-causal <i>FIR</i> filter in a quadrature mirror filter. DATA0 and DATA1 must have the same size. |
| DATA1 | - is a speckle datapack, where DATA1 is a 3D matrix created by grouping NTIMES matrices with NLIN lines and NCOL columns. When <code>N=size(DATA1)</code> , then <code>N(1,1)</code> represents NLIN, <code>N(1,2)</code> represents NCOL and <code>N(1,3)</code> represents NTIMES. DATA1 must be obtained after the down-sampler in the branch of a low-pass H1 non-causal <i>FIR</i> filter in a quadrature mirror filter. DATA0 and DATA1 must have the same size. |
| H0 | - is a vector with the parameters of a FIR filter. H0 must be a low-pass filter with cut-off at $F_s/4$ and an even number of elements, M . For a perfect reconstruction it is necessary that $H_0^2(Z) - H_0^2(-Z) \equiv AZ^B$ for any real A and B. |
-

Output:

-
- | | |
|------|---|
| DATA | - is the synthesis of the speckle datapacks DATA0 and DATA1. The number of images inside DATA is twice the number of images inside DATA0 and DATA1. |
|------|---|
-

6.10.4 Function `firsynthesispath()` - Synthesis of a Path in a Filter Bank

This function makes a synthesis of a path in a filter bank for each pixel of an input datapack DATA. The `firsynthesispath()` function accepts as inputs parameters: the datapack DATA, a H0 filter and a vector SEQ. On the other hand, it returns a datapack DATAOUT, with a reconstructed version of DATA.

The function uses a set of blocks B_i to reconstruct DATA, with $1 \leq i \leq L$, with L being the number of elements of the binary vector $SEQ = [j_1, j_2, \dots, j_L]$, so that $\forall i, j_i \in \{0, 1\}$. Each reconstruction block B_i is composed of:

- An up-sampler by 2,
- A gain of 2 and
- A non-causal Gj_i FIR filter.

As shown in Figure 6.8, the blocks $B_1, B_2, B_3, \dots$ and B_L are called in inverse order to the vector SEQ .

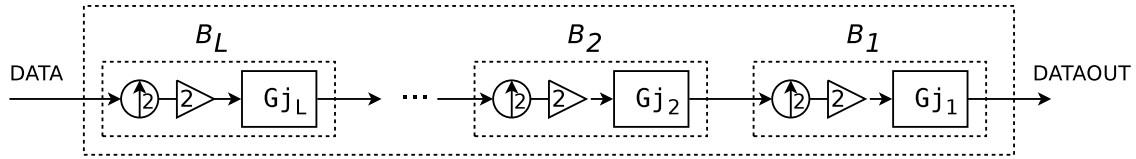


Figure 6.8: Synthesis of a path in a filter bank.

If the $H_0 = [h_0, h_1, h_2, \dots, h_{M-3}, h_{M-2}, h_{M-1}]$ then its Z transform is expressed by the Equation (6.50),

$$H_0(Z) = h_0 Z^{+M_2} + h_1 Z^{+M_2-1} + \dots + h_{M_2} + \dots + h_{M-1} Z^{+M_2-(M-1)}, \quad (6.50)$$

$M_2 = \lfloor M/2 \rfloor$ and the filters G_0 and G_1 are calculated as $G_0(Z) = H_0(Z)$ and $G_1(Z) = -H_0(-Z)$ respectively.

If a datapack DATA is reconstructed as in the Figure 6.9, then in order to have a perfect reconstruction of signal in the DATA obtained at the end of a path SEQ , it is necessary that the H_0 filter, delivered to the `firsynthesispath()` function, fulfill the following $H_0^2(Z) - H_0^2(-Z) \equiv AZ^B$ for any real A and B.

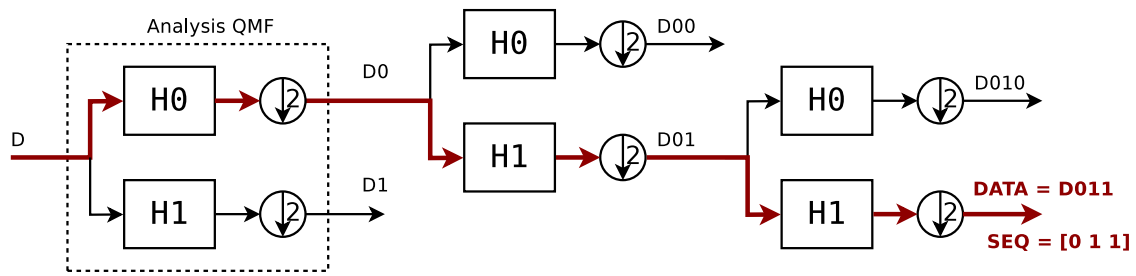


Figure 6.9: Datapack DATA obtained at the end of a path $SEQ = [0 \ 1 \ 1]$.

For using this function, just type the following command at the prompt:

```
DATAOUT=firsynthesispath(DATA,H0,SEQ);

% Getting a reconstructed version of signal
% obtained at the end of a path SEQ=[0 1 1].
% Corresponding to traverse the filters {H0,H1,H1}.
DATAOUT=firsynthesispath(DATA,H0,[0 1 1]);
```

Inputs:

-
- DATA - is a speckle datapack, where DATA is a 3D matrix created by grouping NTIMES matrices with NLIN lines and NCOL columns. When $N = \text{size}(\text{DATA})$, then $N(1,1)$ represents NLIN, $N(1,2)$ represents NCOL and $N(1,3)$ represents NTIMES. DATA must be obtained at the end of a path of any branch of an analysis filter bank scheme.
- H0 - is a vector with the parameters of a FIR filter. H0 must be a low-pass filter with cut-off at $F_s/4$ and an even number of elements, M . For perfect reconstruction it is necessary that $H_0^2(Z) - H_0^2(-Z) \equiv AZ^B$ for any real A and B.
- SEQ - is a vector with binary values, which indicate the follow path in the decomposition scheme used to get the datapack DATA.
-

Output:

-
- DATAOUT - is the synthesis of the speckle datapack DATA. The number of images inside DATAOUT is 2^L times the number of images of DATA, L being $\text{length}(\text{SEQ})$.
-

6.10.5 Function freqmod() - Modulus of Frequency Response

The function calculates the modulus of the frequency response of a system with an impulse response, $h[n]$. The `freqmod()` function uses as input parameters: a vector $H = [h_0, h_1, h_2, \dots, h_{M-1}]$ of M elements that represent $h[n] \equiv h_{n+M_2}$; and a value N that represents the number of points analyzed in the frequency response.

The Z transform of $h[n]$ is expressed by the Equation (6.51),

$$H(Z) = h_0 Z^{+M_2} + h_1 Z^{+M_2-1} + \dots + h_{M_2} + \dots + h_{M-1} Z^{+M_2-(M-1)}, \quad (6.51)$$

with $M_2 = \lfloor M/2 \rfloor$ and the element h_{M_2} representing the value in the instant $n = 0$. Known this, we can describe the expression returned by the `freqmod()` function as: a vector AH of N elements with, $|H(e^{jw})|$, and the modulus of the frequency response of the system with $0 \leq w \leq \pi$. The function also returns a vector FREQN with the normalized frequency for each analyzed point in the vector AH, so that a value equal to a , in the FREQN vector, represents $w = a\pi$ or in other words a frequency $aF_s/2$, F_s being the frequency sampling.

For using this function, just type any of the following commands at the prompt:

```
% Returns the module of the Fourier transform
AH = freqmod(H,N);

% Returns the module of the Fourier transform and
% the normalized frequency to each element of AH
[AH FREQN] = freqmod(H,N);
```



```
% Returns the module of the Fourier transform of h[n] represented
% by the vector H. The vector AH has 256 elementos.
AH = freqmod(H,256);
```

Inputs:

H - is a vector $[h_0, h_1, h_2, \dots, h_{M-1}]$ with the parameters of the impulse response $h[n]$.
The element h_{M_2} represents the value in the instant $n = 0$.

N - is the number of analysis points in the frequency response.

Outputs:

AH - is a vector of N elements with $|H(e^{jw})|$ (the modulus of the frequency response of a system with the impulse response $h[n]$).

FREQN - **[Optional]** is the normalized frequency of the elements in AH. Thus, to the element AH(id) we have a value $w = \text{FREQN(id)} \pi$ or, in other words, a frequency of $\text{FREQN(id)} F_s/2$.

6.10.6 Function qmfmaker() - Quadrature Mirror Filter Maker

This function returns a vector H with the parameters of a low-pass *FIR* filter with a cut-off at $F_s/4$, F_s being the sampling frequency. The `qmfmaker()` function try fulfills the Equation (6.52),

$$D(Z) \equiv H^2(Z) - H^2(-Z) = AZ^B, \quad (6.52)$$

for any real A and B, so that, the standard deviation of the values in $|D(e^{jw})|$ is the minimum to $0 \leq w \leq \pi$.

For using this function, just type any of the following commands at the prompt:

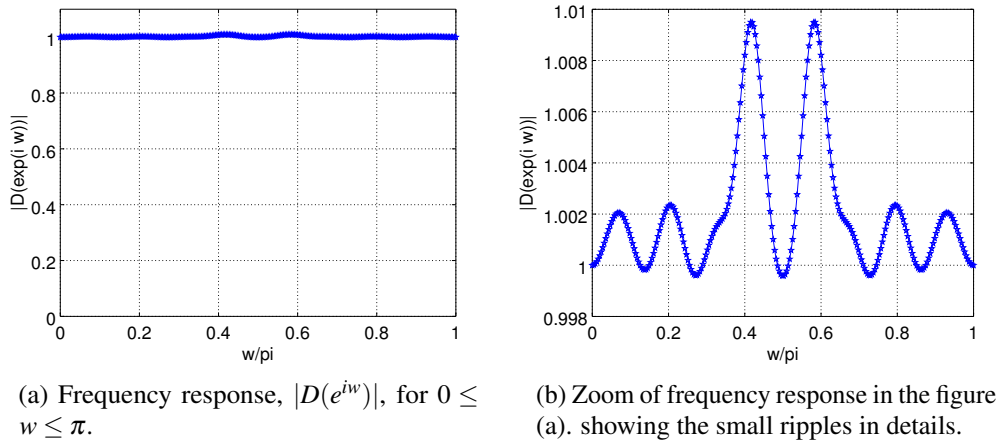
```
H=qmfmaker(ORDER);

% Generating a low-pass FIR filter of ORDER=31 and with cut-off at Fs/4.
H=qmfmaker(31);
```

The Figure 6.10 shows an example of $|D(e^{jw})|$ of a low-pass *FIR* filter, H, of ORDER=31 and with cut-off at $F_s/4$.

Input:

ORDER - is the order of $H = [h_0, h_1, \dots, h_{\text{ORDER}}]$ *FIR* filter. The function tries to fulfill $D(Z) \equiv H^2(Z) - H^2(-Z) = AZ^B$ for any real A and B. For this reason, the number of elements of *FIR* filter must be even and consequently the ORDER should be odd.

Figure 6.10: Frequency response of $D(Z)$.

Output:

H - is a vector with the parameters of a low-pass *FIR* filter with a cut-off at $F_s/4$, F_s being the sampling frequency.

6.10.7 Function `qmfmirror()` - Mirror of a Filter in a Quadrature Mirror Filter Scheme

This function returns a vector H1, representing the quadrature mirror filter of an input vector H0. If $H_0 = [h_0, h_1, h_2, \dots, h_{M-1}]$ is a filter with an impulse response, $h[n] \equiv h_{n+M_2}$, then its Z transform is expressed by Equation (6.53),

$$H_0(Z) = h_0 Z^{+M_2} + h_1 Z^{+M_2-1} + \dots + h_{M_2} + \dots + h_{M-1} Z^{+M_2-(M-1)}, \quad (6.53)$$

with $M_2 = \lfloor M/2 \rfloor$. Thus, the vector H1 is calculated as $H_1(Z) = H_0(-Z)$.

Optionally the `qmfmirror()` function returns a vector AD with the values of $|D(e^{iw})|$ to $0 \leq w \leq \pi$, where $D(Z) \equiv H_0^2(Z) - H_0^2(-Z)$. Another optional output parameter is the vector FREQN that contains the normalized frequency of points in AD.

For using this function, just type any of the following commands at the prompt:

```
H1=qmfmirror(H0);

% H1 is the QMF of H0 and AD is the modulus of D(exp(i*w)).
% AD by default has 65 elements.
[H1 AD]=qmfmirror(H0);

% H1 is the QMF of H0 and AD is the modulus of D(exp(i*w)).
% So that the vector AD has N elements.
[H1 AD]=qmfmirror(H0,N);

% H1 is the QMF of H0, AD is the modulus of D(exp(i*w)) and
% FREQN is the normalized frequency of the elements in AD.
```

```
[H1 AD FREQN]=qmfmirror(H0);
[H1 AD FREQN]=qmfmirror(H0,N);
```

Inputs:

H0 - is a vector with the parameters of a *FIR* filter.

N - **[Optional]** is the number of points (elements) analyzed in the AD vector. By default, N=65.

Outputs:

H1 - is the quadrature mirror filter of H0, so that $H_1(Z) = H_0(-Z)$.

AD - **[Optional]** is the modulus of $D(e^{iw})$, where $D(Z) \equiv H_0^2(Z) - H_0^2(-Z)$. N is the number of analyzed points in the vector AD.

FREQN - **[Optional]** is the normalized frequency of points in the vector AD. Thus, for the point AD(id) we have a $w = \text{FREQN}(id)\pi$ and a frequency $\text{FREQN}(id)F_s/2$, F_s being the sampling frequency.

6.11 Extra Functions

6.11.1 Function hbpmf() - Binary Entropy of a Probability Mass Function

This function returns the binary entropy H of a probability mass function $Pr(a)$ or $Pr(a,b)$ following the Equations (6.54) and (6.55) respectively,

$$H = - \sum_a Pr(a) \log_2(Pr(a)), \quad (6.54)$$

$$H = - \sum_a \sum_b Pr(a,b) \log_2(Pr(a,b)). \quad (6.55)$$

For using this function, just type the following command at the prompt:

```
H=hbpmf(Pr);
```

Input:

Pr - is a probability mass function. The sum of all values $Pr(a,b)$ could be 1.0.

Output:

H - The binary entropy of a probability mass function.

6.11.2 Function threshold2d() - Threshold of 2D Matrix

The `threshold2d()` function uses a matrix `X` as input parameter and returns another matrix `X1` similar to `X`, but with elements smaller than the thresholded value `U`. Thus, all elements in `X` superior to `U` are truncated to `U`, creating `X1`.

For using this function, just type any of the following commands at the prompt:

```
X1=threshold2d(X,U);
[X1 X2]=threshold2d(X,U);
```

Inputs:

-
- `X` - is a 2D matrix that needs to be truncated.
 - `U` - is the threshold value or matrix. It makes all values superior to `U` equal to `U`. `U` can be the same size of `X` or a scalar value.
-

Outputs:

-
- `X1` - is the truncated matrix with its highest values limited to `U`.
 - `X2` - [Optional] is the complement of the truncated matrix, i.e. `X2 = X-X1`.
-

6.11.3 Function mwindowing() - Mean Values in the Windows of a Matrix

This function divides the `MAT` matrix in windows of `WLines` lines and `WColumns` columns, then the mean value of all elements is calculated. With this information, a new matrix `MATW` is created with the same size and as the `MAT`, and the mean values in the `MAT` matrix are replaced in all elements in the `MATW` matrix for each window.

For using this function, just type any of the following commands at the prompt:

```
MATW = mwindowing(MAT,WLines,WColumns);
% Mean values, in analysis windows of 8x10 pixels, in the MAT matrix.
MATW = mwindowing(MAT,8,10);
```

Inputs:

-
- `MAT` - is a matrix with `NLIN` lines and `NCOL` columns.
 - `WLines` - is the number of lines in each analysis window.
 - `WColumns` - is the number of columns in each analysis window.
-

Output:

-
- `MATW` - is a matrix with the mean values in analysis windows of `WLines×WColumns` pixels in the `MAT` matrix.
-

6.12 Quality Tests

6.12.1 Function `satdark()` - Analyzing Saturation and Sub-Exposition of Light

This function analyzes the saturation and sub-exposition of light (CARDOSO; BRAGA; RABAL, 2012) in windows of one image $\mathbf{I}_k \equiv \text{DATAFRAME}$. The image is divided into windows of `WLines` lines by `WColumns` columns.

The function considers intensity values below `MaxDark` dark, and considers intensity values above `MinSat` saturated. Each window under analysis is considered dark or saturated, when it reaches the `P` percentage previously defined.

For using this function, just type any of the following commands at the prompt:

```
[F S D] = satdark(DATAFRAME, WLines, WColumns, MaxDark, MinSat, P);
% Windows analysed (ROI) of 5x6 pixels and
% 50 percent of pixels in windows to tag dark or saturate.
[F S D] = satdark(DATAFRAME,6,5, MaxDark, MinSat, 50);
```

Inputs:

<code>DATAFRAME</code>	-	is the image in analysis, which is a 2D matrix $\mathbf{I}_k$.
<code>WLines</code>	-	is the number of lines in the windows analysis.
<code>WColumns</code>	-	is the number of columns in the windows analysis.
<code>MaxDark</code>	-	is the maximum gray-scale level that is considered as dark.
<code>MinSat</code>	-	is the minimum gray-scale level that is considered as saturated.
<code>P</code>	-	is the percentage of pixels in a windows tagged as dark or saturate.

Outputs:

<code>F</code>	-	is an image with dark or saturated areas in a region of interest (ROI). The dark windows are filled with 0, and the saturated windows with 255. A window is considered as dark or saturated, when a collection of values within the matrix overcomes a percentage established by the user in the ROI.
<code>S</code>	-	is a matrix with the same size of <code>F</code> , which has ones in regions with saturated windows and zeros in other ROI's.
<code>D</code>	-	is a matrix with the same size of <code>F</code> , which has ones in regions with dark windows and zeros in other ROI's.

6.12.2 Function `sscont()` - Spatial Speckle Contrast Window Method

Spatial speckle contrast window (CARDOSO; BRAGA; RABAL, 2012) addresses the contrast in a ROI within an image. The image ($\mathbf{I}_k \equiv \text{DATAFRAME}$) is divided into L windows W^l , $1 \leq l \leq L$ of `WLines` lines and `WColumns` columns.

The contrast value C^l (Equation 6.56) in a windows W^l is calculated as the quotient between the values of the spatial (populational) standard deviation σ^l seen in the Equation (6.58) and the

spatial mean μ^l seen in the Equation (6.57). All the pixels in analysis windows are filled with the contrast value C^l ,

$$C^l = \frac{\sigma^l}{\mu^l}, \quad (6.56)$$

where

$$\mu^l = \langle W^l \rangle \quad (6.57)$$

and

$$\sigma^l = \sqrt{\langle (W^l - \mu^l)^2 \rangle}. \quad (6.58)$$

For using this function, just type any of the following commands at the prompt:

```
[C,mC] = sscont(DATAFRAME,WLines,WColumns);
% Windows analysed (ROI) of 6x5 pixels.
[C,mC] = sscont(DATAFRAME,6,5);
```

Inputs:

DATAFRAME	-	is the image in analysis, which is a 2D matrix $\mathbf{I}_k$.
WLines	-	is the number of lines in the ROI.
WColumns	-	is the number of columns in the ROI.

Outputs:

C	-	is the spatial speckle contrast ROI.
mC	-	is the mean value of the contrast in all windows.

6.12.3 Function homogeneity() - Homogeneity of Spatial Variability

Homogeneity of spatial variability (CARDOSO; BRAGA; RABAL, 2012) divides a datapack (DATA) in 3D analysis windows (or sub datapacks) of NTIMES matrices with WLines lines and WColumns columns, where DATA is a 3D matrix grouping intensity matrices $\mathbf{I}_k \equiv \text{DATA}(:, :, k)$, $1 \leq k \leq \text{NTIMES}$. In these windows we calculate the activity indicator selected with the input variable Type. Once the activity indicator value $A(i, j)$ of window (i, j) is known for all windows, the homogeneity values $H(i, j)$ are calculated as ¹ the Equation (6.59),

$$H(i, j) = 100 \left(1 - \frac{C(i, j)}{C_{\max}} \right), \quad (6.59)$$

¹We use the populational case of standard deviation

where the value $C(i, j)$ is calculated as in the Equations (6.60), (6.61), (6.62), and (6.63),

$$C(i, j) = \frac{\sigma_{Z^{(i,j)}}}{\mu_{Z^{(i,j)}}}, \quad (6.60)$$

$$\mu_{Z^{(i,j)}} = \frac{1}{5} \sum_l Z^{(i,j)}(l), \quad (6.61)$$

$$\sigma_{Z^{(i,j)}}^2 = \frac{1}{5} \sum_l (Z^{(i,j)}(l) - \mu_{Z^{(i,j)}})^2, \quad (6.62)$$

$$\begin{aligned} Z^{(i,j)} &= \{A(i, j-1) \quad A(i-1, j) \quad A(i, j) \quad A(i+1, j) \quad A(i, j+1)\} \\ &\equiv \{Z^{(i,j)}(0) \quad Z^{(i,j)}(1) \quad Z^{(i,j)}(2) \quad Z^{(i,j)}(3) \quad Z^{(i,j)}(4)\} \end{aligned} \quad (6.63)$$

and C_{max} represent the maximum value of spatial contrast of all $C(i, j)$ values.

For using this function, just type any of the following commands at the prompt:

```
[Y X] = homogeneity(DATA,WLines,WColumns,Type);

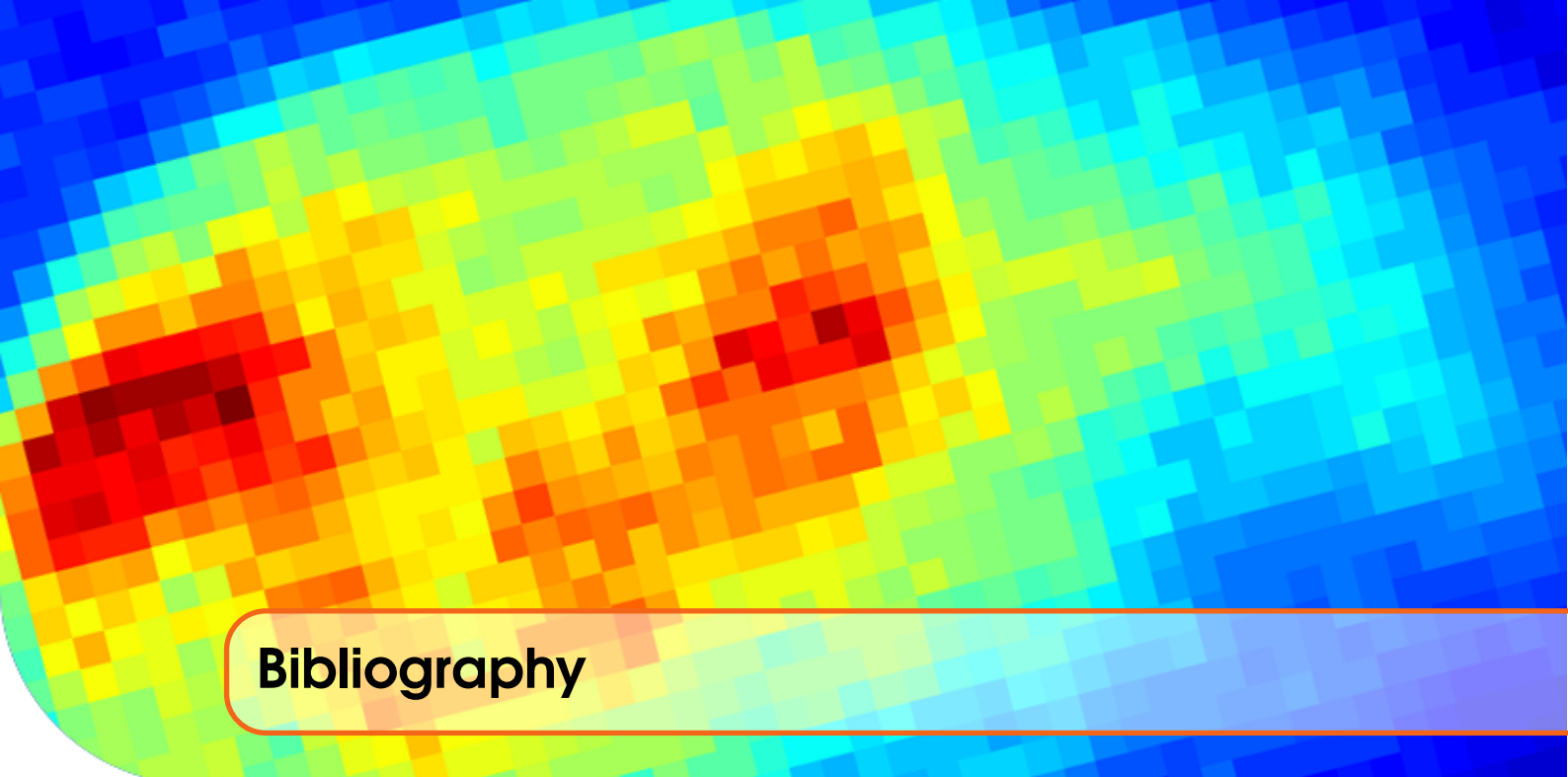
% Analysis of the datapack DATA using
% a windows of 6x5 pixels with
% inertia moment as activiyy indicator.
[Y X] = homogeneity(DATA,6,5,0);
```

Inputs:

DATA	-	is the speckle data. Where DATA is a 3D matrix grouping NTIMES matrices with NLIN lines and NCOL columns. When $N = \text{size}(\text{DATA})$, then N(1,1) represents NLIN, N(1,2) represents NCOL and N(1,3) represents NTIMES.
WLines	-	is the number of lines in the ROI.
WColumns	-	is the number of columns in the ROI.
Type	-	If Type is 1, AVD (BRAGA et al., 2011) is used as activity indicator. If Type is 2, it is used the temporal speckle standard deviation as activity indicator (NOTHDURFT; YAO, 2005). Otherwixe, the inertia moment (ARIZAGA; TRIVI; RABAL, 1999) technique is used as activity indicator. In the cases of AVD and inertia moment, the Cardoso's normalization (CARDOSO; BRAGA, 2014) is used over the co-occurrence matrix. In all cases, the activity indicators were calculated over all pixels in the window and not only over a line.

Outputs:

-
- Y - is the homogeneity percentages in the ROI (CARDOSO; BRAGA; RABAL, 2012). The homogeneity value $H(i, j)$ is represented as a window (matrix) with WLines lines with WColumns columns inside Y.
 - X - is the activity indicator value in all ROI's. The activity indicator value $A(i, j)$ is represented as a window (matrix) with WLines lines with WColumns columns inside X.
-



Bibliography

Articles and Books

- [ABV13] ALVES, J. A.; BRAGA, R. A.; VILAS-BOAS, E. V. B. Identification of respiration rate and water activity change in fresh-cut carrots using biospeckle laser and frequency approach. In: **Postharvest Biology and Technology** 86: 381–386. 2013.
- [Ari+02] ARIZAGA, R. et al. Display of local activity using dynamical speckle patterns. In: **Optical Engineering** 41(2): 287–294. 2002.
- [AS13] AGRAWAL, S. K.; SAHU, O. P. Two-channel quadrature mirror filter bank: An overview. In: **ISRN Signal Processing** 2013. 2013.
- [Asa88] ASAKURA, T. Dynamic properties of bio-speckles and their application to blood flow measurements. In: **Anritsu News** 8: 4–9. 1988.
- [ATR99] ARIZAGA, R.; TRIVI, M.; RABAL, H. J. Speckle time evolution characterization by the co-occurrence matrix analysis. In: **Optics & Laser Technology** 31(2): 163–169. 1999.
- [Blo+11] BLOTTA, E. et al. Evaluation of speckle-interferometry descriptors to measuring drying-of-coatings. In: **Signal Processing** 91(10): 2395–2403. 2011.
- [Bra+11] BRAGA, R. A. et al. Evaluation of activity through dynamic laser speckle using the absolute value of the differences. In: **Optics Communications** 284(2): 646–650. 2011.
- [Bra+12] BRAGA, R. A. et al. Biospeckle numerical values over spectral image maps of activity. In: **Optics Communications** 285: 553–561. 2012.

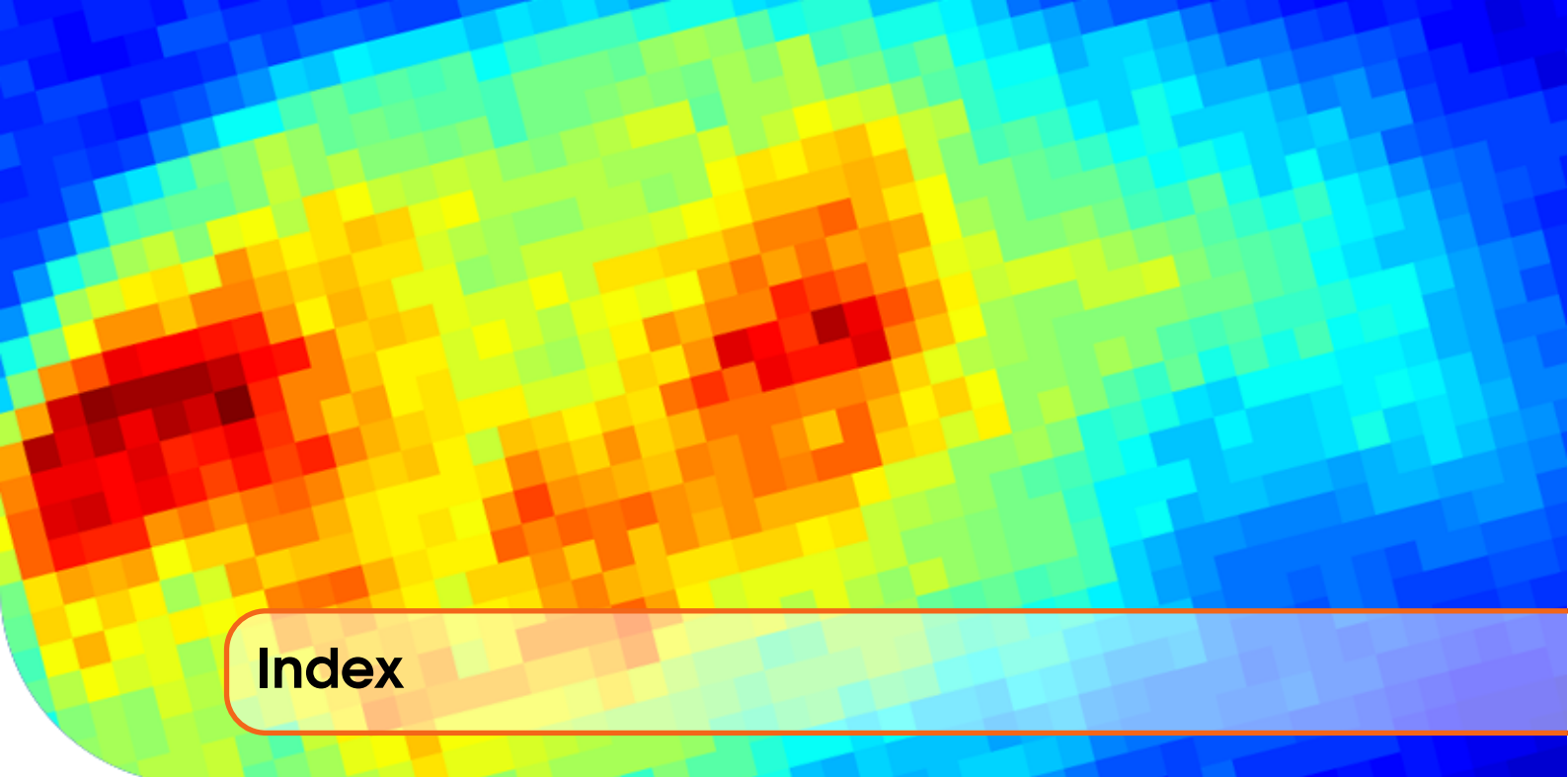
- [Bri75] BRIERS, J. Wavelength dependence of intensity fluctuations in laser speckle patterns from biological specimens. In: **Optics Communications** 13: 324–326. 1975.
- [BW96] BRIERS, J. D.; WEBSTER, S. Laser speckle contrast analysis (LASCA): a non-scanning, full-field technique for monitoring capillary blood flow. In: **Journal of Biomedical Optics** 1(2): 174–179. 1996.
- [CB14] CARDOSO, R. R.; BRAGA, R. A. Enhancement of the robustness on dynamic speckle laser numerical analysis. In: **Optics and Lasers in Engineering** 63: 19–24. 2014.
- [CBR12] CARDOSO, R. R.; BRAGA, R. A.; RABAL, H. J. Alternative protocols on dynamic speckle laser analysis. In: **Proc. SPIE** 8413: pages. 2012.
- [Dav01] DAVIS, J. W. Hierarchical motion history images for recognizing human motion. In: **Detection and Recognition of Events in Video, 2001. Proceedings. IEEE Workshop on**: 39–46. 2001.
- [DSW11] DISTEFANO, J. J.; STUBBERUD, A. J.; WEHBRING, I. J. W. **Digital Signal Processing 2nd ed.** McGraw-Hill Education. 2011.
- [Eat+15] EATON, J. W. et al. **GNU Octave version 4.0.0 manual: a high-level interactive language for numerical computations**. Available at: <http://www.gnu.org/software/octave/doc/interpreter> . 2015.
- [FA75] FUJII, H.; ASAKURA, T. Statistical properties of image speckle patterns in partially coherent light. In: **Nouvelle Revue d'Optique** 6(1): 5. 1975.
- [Fuj+85] FUJII, H. et al. Blood flow observed by time-varying laser speckle. In: **Optics Letters** 10(3): 104–106. 1985.
- [Fuj+87] FUJII, H. et al. Evaluation of blood flow by laser speckle image sensing. Part 1. In: **Applied Optics** 26(24): 5321–5325. 1987.
- [God+12] GODINHO, R. et al. Online biospeckle assessment without loss of definition and resolution by motion history image. In: **Optics and Lasers in Engineering** 50: 366–372. 2012.
- [HSD73] HARALICK, R. M.; SHANMUGAM, K.; DINSTEN, I. Textural Features for Image Classification. In: **Systems, Man and Cybernetics, IEEE Transactions on SMC**-3(6): 610–621. 1973.
- [KAZ12] KURENDA, A.; ADAMIAK, A.; ZDUNEK, A. Temperature effect on apple bio-speckle activity evaluated with different indices. In: **Postharvest Biology and Technology** 67: 118–123. 2012.
- [MCB14] MOREIRA, J.; CARDOSO, R. R.; BRAGA, R. A. Quality test protocol to dynamic laser speckle analysis. In: **Optics and Lasers in Engineering** 61: 8–13. 2014.
- [MN14] MINZ, P. D.; NIRALA, A. Intensity based algorithms for biospeckle analysis. In: **Optik - International Journal for Light and Electron Optics** 125(14): 3633–3636. 2014.

- [Mor+82a] MORLET, J. et al. Wave propagation and sampling theory-Part I: Complex signal and scattering in multilayered media. In: **Geophysics** 47(2): 203–221. 1982.
- [Mor+82b] MORLET, J. et al. Wave propagation and sampling theory-Part II: Sampling theory and complex waves. In: **Geophysics** 47(2): 222–236. 1982.
- [NY05] NOTHDURFT, R.; YAO, G. Imaging obscured subsurface inhomogeneity using laser speckle. In: **Optics Express** 13(25): 10034–10039. 2005.
- [OTD89a] OULAMARA, A.; TRIBILLON, G.; DUVERNOY, J. Biological Activity Measurement on Botanical Specimen Surfaces Using a Temporal Decorrelation Effect of Laser Speckle. In: **Journal of Modern Optics** 36: 165–179. 1989.
- [OTD89b] OULAMARA, A.; TRIBILLON, G.; DUVERNOY, J. Biological Activity Measurement on Botanical Specimen Surfaces Using a Temporal Decorrelation Effect of Laser Speckle. In: **Journal of Modern Optics** 36(2): 165–179. 1989.
- [Pap84] PAPOULIS, A. **Probability, Random Variables, and Stochastic Processes**. McGraw-Hill Series in Electrical Engineering. McGraw-Hill. ISBN: 9780070484689. 1984.
- [PZ09] POPOV, A.; ZHUKOV, M. Computation of continuous wavelet transform of discrete signals with adapted mother functions. In: **Proc. SPIE** 7502: pages. 2009.
- [RB08] RABAL, H. J.; BRAGA, R. A. **Dynamic Laser Speckle and Applications**. Optical Science and Engineering. CRC Press. Available at: https://books.google.com.br/books?id=T4MA_XtWAwMC . 2008.
- [RRB16] REIS, R. O.; RABAL, H. J.; BRAGA, R. A. Light intensity independence during dynamic laser speckle analysis. In: **Optics Communication** 366: 185–193. 2016.
- [She03] SHESKIN, D. J. **Handbook of Parametric and Nonparametric Statistical Procedures: Third Edition**. CRC Press. 2003.
- [VH92] VETTERLI, M.; HERLEY, C. Wavelets and filter banks: theory and design. In: **Signal Processing, IEEE Transactions on** 40(9): 2207–2232. 1992.
- [XJK95] XU, Z.; JOENATHAN, C.; KHORANA, B. M. Temporal and spatial properties of the time-varying speckles of botanical specimens. In: **Optical Engineering** 34(5): 1487–1502. 1995.
- [Zdu+07] ZDUNEK, A. et al. New nondestructive method based on spatial-temporal speckle correlation technique for evaluation of apples quality during shelf-life. In: **International Agrophysics** 21(3): 305–310. 2007.

Websites

- [BPM] BRAGA, R. A.; PUJAICO RIVERA, F.; MOREIRA, J. **Biospeckle Laser Tool Library v1.0.0**. Available at: <http://repositorio.ufla.br/handle/1/11116> , visited on 04/29/2016.

- [Bra] BRAGA, R. A. **Biospeckle data of a maize seed**. Available at:<http://repositorio.ufla.br/jspui/handle/1/10560> , visited on 11/04/2015.
- [Bra+] BRAGA, R. A. et al. **Speckle tool v 1.2**. Available at:<http://repositorio.ufla.br/jspui/handle/1/1760> , visited on 03/12/2014.
- [BS] BRAGA, R. A.; SILVA, M. M. **Quality test for dynamic laser speckle - executable version**. Available at:<http://repositorio.ufla.br/handle/1/4593> , visited on 11/03/2014.
- [bsl] BSLTL-PROJECT-GROUP. **Project homepage**. Available at:<http://www.nongnu.org/bsl1/> , visited on 09/14/2016.
- [Oct] OCTAVE-FORGE-DEVELOPERS. **Octave-Forge**. Available at:<http://octave.sourceforge.net/packages.php> , visited on 09/14/2016.
- [VB] VIVAS, P. G.; BRAGA, R. A. **Biospeckle data of a coffee seed**. Available at:<http://repositorio.ufla.br/jspui/handle/1/10619> , visited on 11/25/2015.



Index

A

AD
 Fujii 41, 83, 131
AVD 46, 76, 85, 124, 135

C

COM 44, 72, 119
Contrast 84, 133
Correlation 45, 81, 82, 129

D

Datapack 69, 113

E

Entropy 106, 150
Extra Functions 103, 150

F

FAQ 109
Frequency Analysis 47, 93, 141
 Convolution 141
 CWT 50, 100
 DWT 48, 96, 142
 Filter Bank . 47, 48, 50, 96, 142, 144, 145,
 148, 149
 Filtering 47, 93, 142
 Frequency Response 147
 IDWT 48, 97, 99, 144, 145
 QMF 48, 96, 148, 149
Fujii
 AD 41, 83, 131

G

GD 41, 83, 133
Graphic Methods 40, 83, 131

I

IM.....46, 75, 85, 122, 135
Installing.....66

K

Kurtosis 87, 138

M

MHI.....37, 90, 140

N

NUMAD 80, 128
Numerical Methods.....43, 75

P

PMF.....73
 PMFAD 73, 121
 PMFRD 74, 121
PTD.....87, 137

Q

Quality Tests 31, 91, 152
 Contrast 33, 91, 152
 Homogeneity 35, 92, 153
 Saturation 31, 91, 152

R

Recommendations
 Gnu/Linux 61
 Windows.....62
Repository
 Picture Packages.....110

Requirements.....59
 System Requirements 60
 User Requirements.....59
RVD 78, 86, 126, 136

S

SD 42, 84, 133
Skewness.....88, 139

T

Threshold.....105, 151
THSP43, 71, 116

W

Website
 Bug Reports.....110
 Download Page.....110
 Home Page.....110
Windowing 103, 151

