University of Texas at El Paso
DigitalCommons@UTEP

Open Access Theses & Dissertations

2016-01-01

Blind Image Deconvolution Based On Sparsity: Theoretical Justification And Improvement Of State-Of-The-Art Techniques

Fernando Cervantes
University of Texas at El Paso, fcervantes@utep.edu

Follow this and additional works at: https://digitalcommons.utep.edu/open_etd

Part of the Electrical and Electronics Commons

Recommended Citation

Cervantes, Fernando, "Blind Image Deconvolution Based On Sparsity: Theoretical Justification And Improvement Of State-Of-The-Art Techniques" (2016). *Open Access Theses & Dissertations*. 823.
https://digitalcommons.utep.edu/open_etd/823

This is brought to you for free and open access by DigitalCommons@UTEP. It has been accepted for inclusion in Open Access Theses & Dissertations by an authorized administrator of DigitalCommons@UTEP. For more information, please contact lweber@utep.edu.

BLIND IMAGE DECONVOLUTION BASED ON SPARSITY: THEORETICAL JUSTIFICATION AND IMPROVEMENT OF STATE-OF-THE-ART TECHNIQUES

FERNANDO CERVANTES

Doctoral Program in Electrical and Computer Engineering

APPROVED:

Bryan E. Usevitch, Ph.D., Chair

Scott A. Starks, Ph.D.

Vladik Kreinovich, Ph.D.

Charles H. Ambler, Ph.D.
Dean of the Graduate School

To my mother Elizabeth, my late father Jose Luis, my twin sister Diana, and my brother Raul. I also dedicate this to my friends, specifically Leobardo Valera and Octavio Lerma, who were always there when I needed them the most.

BLIND IMAGE DECONVOLUTION BASED ON SPARSITY: THEORETICAL JUSTIFICATION AND IMPROVEMENT OF STATE-OF-THE-ART TECHNIQUES

by

FERNANDO CERVANTES

DISSERTATION

Presented to the Faculty of the Graduate School of
The University of Texas at El Paso
in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

Department of Electrical and Computer Engineering
THE UNIVERSITY OF TEXAS AT EL PASO
August 2016

Acknowledgements

I would like to thank God for giving me the wisdom and the patience to overcome the challenges that I faced throughout my school career. I would also like to thank Him for surrounding me with loving people that supported and motivated me to overcome these challenges.

I would like to thank my family and all my friends who supported me throughout this difficult endeavor.

I would also like to thank Dr. Bryan Usevitch for accepting me as his student, his support, whether it was academic or non-academic related, his guidance, time and patience. He was always there for me when I needed him. I could never thank him enough. I would like to thank Dr. Scott Starks for being part of my dissertation committee at the final stages of my research. His opinions and suggestions were highly valued and appreciated.

I would also like to thank Dr. Vladik Kreinovich for his support and his guidance. His loving and caring nature for his students are unmatchable. He was always willing to help his students, whether they were in his department or not.

Overall, I would like to thank the Electrical Engineering Department for their support and helping me graduate.

Abstract

Observed signals and images are distorted by noise and blurring. In precise terms, blurring is a convolution of the original image with some blurring function. From this viewpoint, to reconstruct the original image, we need to reverse the effects of this convolution, i.e., to "deconvolve" the image.

Many efficient methods for image deconvolution are known. However, most of these methods assume that we know the blurring function – or at least that we have some partial information about the blurring function. In some practical situations, however, we do not have this information. In such situations, we need to perform *blind image deconvolution*, i.e., deconvolution without any knowledge about the original blurring function.

Recently, several algorithms have been proposed that successfully perform blind image deconvolution. However, these algorithms have some limitations. First, some of the techniques used by these algorithms lack a convincing theoretical justification. Without a theoretical justification, there is no guarantee that these methods will be as successful on other images as they are on currently tested examples. Second, these methods are still not perfect, so it is desirable to try tot improve them.

In this dissertation, we start with the state-of-the-art blind image deconvolution method based on the ideas of sparsity and ℓ^p-regularization – ideas which do not have a convincing theoretical justification. We provide a theoretical justification for these ideas, and we also describe a modification of the state-of-the-art techniques that lead to a better quality blind image deconvolution.

Table of Contents

Page

Acknowledgements iv

Abstract v

Table of Contents vi

Chapter

1 Introduction 1

2 Blind Image Deconvolution: Formulation of the General Problem and Description of State-of-the-Art Techniques 3

- 2.1 Signal and Image Deconvolution: Formulation of the Problem 3
- 2.2 Signal and Image Deconvolution in the Ideal No-Noise Case 6
- 2.3 Signal and Image Deconvolution in the Presence of Noise 12
- 2.4 Blind Image Deconvolution in the Presence of Prior Knowledge 15
- 2.5 Blind Image Deconvolution in the Absence of Prior Knowledge: Need to Use Sparsity-Based Techniques 17
- 2.6 State-of-the-Art Techniques for Sparsity-Based Blind Image Deconvolution 18

3 Open Problems Related to Blind Image Deconvolution: Need for Theoretical Justification and Need for Improvement of the Existing Techniques 22

- 3.1 Need for Theoretical Justification 22
- 3.2 Need for Improvement 23

4 Theoretical Justification of Sparsity-Based Techniques in Blind Image Deconvolution 24

- 4.1 Formulation of the Problem 24
  - 4.1.1 Sparsity-based techniques are useful 24
  - 4.1.2 But why are sparsity-related techniques useful? 25
  - 4.1.3 What we do in this chapter 25
- 4.2 General Analysis of the Problem 25

4.2.1 Why do we need data processing in the first place? 25

4.2.2 Before we perform data processing, we first need to know which inputs are relevant 27

4.3 Analysis of the Problem: Let Us Use Fuzzy-Related Techniques 27

4.3.1 Description of our problem in natural-language terms 27

4.3.2 Fuzzy techniques can translate this natural-language description into a precisely formulated problem 28

4.3.3 Resulting precise formulation of the first stage of data processing in precise terms 30

4.3.4 An additional fact that we can use 30

4.3.5 A comment about t-norms 31

4.4 Definitions and the Main Result: Fuzzy-Related Techniques Explain Sparsity 31

4.5 A Similar Derivation Can Be Obtained in the Probabilistic Case 34

5 Theoretical Justification of ℓ^p-Techniques in Blind Image Deconvolution 37

5.1 Formulation of the Problem 37

5.1.1 In general, signal and image reconstruction are ill-posed problems . 37

5.1.2 Tikhonov's regularization: a brief reminder 39

5.1.3 From continuous to discrete signals and images 40

5.1.4 Limitations of Tikhonov regularization 41

5.1.5 ℓ^p-methods as a heuristic idea to take non-smoothness into account 41

5.1.6 Remaining problem 42

5.1.7 What we do in this chapter 42

5.2 Let us Apply Fuzzy Techniques to Our Problem 42

5.2.1 Need to describe imprecise ("fuzzy") expert knowledge 42

5.2.2 "And"- and "or"-operations: a brief reminder 43

5.2.3 What we are trying to formalize 44

5.2.4 What we need to do to formalize this statement 44

5.2.5 Selecting an "and"-operation 45

5.2.6 The selection of an "and"-operation simplifies the corresponding optimization problem 45
5.2.7 Which function $g(x)$ should we select: idea 46
5.2.8 Let us find the corresponding function $g(x)$ 46
5.2.9 Conclusion: we have indeed justified the ℓ^p-method 48
6 The Idea of Rotation-Invariance Enables Us To Further Improve the Results of the State-of-the-Art Blind Deconvolution Techniques 50
6.1 Need for Improvement: Reminder 50
6.2 How to Improve: Main Idea 51
6.3 From the Idea to the Algorithmic Details 53
6.4 Resulting Modification of the State-of-the-Art Blind Deconvolution Algorithm 54
6.5 Testing the New Algorithm 56
7 Towards Using Embedded Zerotree Wavelet (EZW) Algorithm 59
7.1 Theoretical Background of Embedded Zerotree Wavelet (EZW) Coding . . 59
7.1.1 How to Incorporate EZW Into Blind Deconvolution: First Idea . . . 65
7.1.2 How to Incorporate EZW Into Blind Deconolution: Second Idea . . 70
7.2 Numerical Results of Applying the Two Ideas 72
8 Conclusions and Future Work 75
8.1 Conclusions 75
8.2 Future Work 75
A Matlab Code for the Original Algorithm and for Its Proposed Modifications . . 82
A.1 Matlab Code for the Original State-of-the-Art Image Deconvolution Algorithm 82
A.2 Matlab Code for the Newly Proposed Rotation-Invariant Modification of the State-of-the-Art Image Deconvolution Algorithm 87
A.3 Matlab Code for the Method Using the First EZW Idea 92
A.4 Matlab Code for the Method Using the First EZW Idea Combined With Rotation-Invariance 99
A.5 Matlab Code for the Method Using the Second EZW Idea 105

A.6 Matlab Code for the Method Using the Second EZW Idea Combined With Rotation-Invariance 109

Curriculum Vita 115

Chapter 1

Introduction

Observed signals and images are distorted by noise and blurring. In precise terms, blurring is a convolution of the original image with some blurring function. From this viewpoint, to reconstruct the original image, we need to reverse the effects of this convolution, i.e., to "deconvolve" the image.

Many efficient methods for image deconvolution are known. However, most of these methods assume that we know the blurring function – or at least that we have some partial information about the blurring function. In some practical situations, however, we do not have this information. In such situations, we need to perform *blind image deconvolution*, i.e., deconvolution without any knowledge about the original blurring function.

Recently, several algorithms have been proposed that successfully perform blind image deconvolution; see, e.g., [3]. However, these algorithms have some limitations.

- First, some of the techniques used by these algorithms lack a convincing theoretical justification. Without a theoretical justification, there is no guarantee that these methods will be as successful on other images as they are on currently tested examples.
- Second, these methods are still not perfect, so it is desirable to try to improve them.

In this dissertation, we start with the state-of-the-art blind image deconvolution method [3] based on the ideas of sparsity and ℓ^p-regularization – ideas which do not have a convincing theoretical justification.

We provide a theoretical justification for these ideas, and we also describe a modification of the state-of-the-art techniques that lead to a better quality blind image deconvolution. The structure of the dissertation is as follows:

- Chapter 2 contains the formulation of the general problem of blind deconvolution and the description of the state-of-the-art blind image deconvolution algorithm.

- Chapter 3 discussed the limitations of the state-of-the-art blind image deconvolution techniques – i.e., open problems that we solve in this dissertation: the need for a theoretical justification and the need for improvement.

- Chapter 4 contains a theoretical justification of the use of sparsity-based techniques in blind image deconvlution.

- Chapter 5 contains a theoretical justification of the use of ℓ^p-regularization techniques in blind image deconvolution.

- Chapter 6 describes rotation invariance ideas that leads to an improved algorithm for blind image deconvolution. This chapter also contains the results of the experimental comparison that shows that the newly proposed techniques lead to a statistically significant improvement in the accuracy of image reconstruction.

- In Chapter 7, we discuss possible combination of our ideas with the Embedded Zerotree Wavelet (EZW) algorithm.

- Chapter 8 contains conclusions and suggestions for future work.

- Finally, the Appendix contains the corresponding Matlab codes.

Chapter 2

Blind Image Deconvolution: Formulation of the General Problem and Description of State-of-the-Art Techniques

The main problem with which we deal in this dissertation is the problem of blind image decomposition, when we need to reconstruct a blurred image in situations when no information about the blurring is available. In this chapter, we describe this problem in precise terms and explain how this problem is currently solved.

For that, we start with a brief reminder of the general problem of image deconvolution and techniques for solving this problem in situations when the blurring function is known, then we describe techniques for solving the image deconvolution problem when we have partial information about the blurring function, and finally, we get to the problem of blind image deconvolution.

2.1 Signal and Image Deconvolution: Formulation of the Problem

Our information about the physical world comes from measurements. Measurement results are, in general, different from the actual values of the corresponding quantities. For exampleple, when we measure the values of the quantity of interest at different moments of time, then the measurement results y_1, y_2, etc., are, in general, different from the actual values x_1, x_2, $\ldots$, of the physical quantity at the corresponding moments of time.

There are two main reasons for this difference between the actual values x_k and the observed values y_k:

- first, there is usually an additive noise n_k;
- second, there is inertia: even if the actual value changes abruptly, it takes some time for the sensor to capture this change.

As a result of the noise, the value y_k is different from the value x_k. As a result of inertia, we have what is called a blur: the measurement result y_k depends not only on the current value x_k of the physical quantity, but also on the previous values x_{k-1}, x_{k-2}, $\ldots$. Usually, the dependence of y_k on x_i is linear, so we conclude that

$$y_k = h_{k,0} \cdot x_k + h_{k,1} \cdot x_{k-1} + \ldots + n_k,$$

for some coefficients $h_{k,i}$.

The coefficients $h_{k,0}$, $h_{k,1}$, $\ldots$ describe how the measuring instrument distorts the signal in the k-th moment of time. The properties of measuring instruments usually do not change with time; thus, these coefficients usually do not depend on the moment k:

$$h_{1,i} = h_{2,i} = \ldots = h_{k,i} = \ldots$$

for Let us denote this common value of $h_{k,i}$ by h_i.

The values h_i describe the blur. They are known as a *point spread function* (PSF, for short) or, alternatively, a *blurring function*.

Substituting $h_{k,i} = h_i$ into the above formula, we get

$$y_k = h_0 \cdot x_k + h_1 \cdot x_{k-1} + \ldots + n_k = \sum_{i=0} h_i \cdot x_{k-i} + n_k.$$

The sum

$$\sum_{i=0} h_i \cdot x_{k-i} \tag{2.1.1}$$

is known as as the *convolution* of the sequences x_k and h_k.

Once we get the measurement results y_k, we would like to reconstruct the actual values x_k from y_k as accurately as possible. This reconstruction – i.e., inverting the effects of convolution – is known as *deconvolution.*

A similar problem can be formulated for image deconvolution. When we observe an image, we usually observe the image intensity values $y(i, j)$ at different locations (i, j) on a rectangular grid, i.e., at a spatial location $(u_0 + i \cdot \Delta u, v_0 + j \cdot \Delta v)$, where (u_0, v_0) is the starting point and Δu and Δv are distances between the neighboring pixels in the u- and v-directions.

Similarly to signals, each observed value $y(i, j)$ is, in general, different from the actual (desired) value $x(i, j)$ of the corresponding intensity. First, there is noise (measurement error), and second, the image is blurred, in the sense that the observed signal $y(i, j)$ reflects not only the actual intensity $x(i, j)$ at the same spatial location (i, j), but also the intensities $x(i', j')$ at nearby locations. Under the assumption that the dependence of $y(i, j)$ on x is linear, we conclude that

$$y(i, j) = \sum_{i', j'} h(i, j, i', j') \cdot x(i', j') + n(i, j)$$

for some coefficients $h(i, j, i', j')$, where $n(i, j)$ denotes the (additive) noise.

Usually, the blurring is the same for all the pixels, so the coefficients $h(i, j, i', j')$ depend only on the differences $(i - i', j - j')$ between the spatial locations (i, j) and (i', j'):

$$y(i, j) = \sum_{i'} \sum_{j'} h(i - i', j - j') \cdot x(i', j') + n(i, j). \tag{2.1.2}$$

Thus, the observed image $y(i, j)$ is obtained from the actual image $x(i, j)$ by a convolution. Based on the observed image $y(i, j)$, we need to reconstruct the original image $x(i, j)$, i.e., to perform a *deconvolution.*

2.2 Signal and Image Deconvolution in the Ideal No-Noise Case

Our ultimate goal is to describe methods for blind signal and image deconvolution, when we reconstruct the signal x_i or the image $x(i,j)$ without knowing the corresponding blurring function h_j or $h(i,j)$. Most methods for solving this problem, however, are modifications of the methods for solving the signal/image deconvolution problem in situations when we do know the blurring function, so we will start by describing these non-blind deconvolution techniques.

These non-blind deconvolution techniques, in their turn, are usually modifications of the techniques that could be applied in the ideal no-noise case, when the additive noise (n_i or $n(i,j)$) can be safely ignored. Let us therefore start by describing these methods.

In the absence of noise, the formula (2.1.1) becomes a system of linear equations for the unknowns x_i:

$$y_k = \sum_i h_{k-i} \cdot x_i. \tag{2.2.1}$$

In practice, we can only make finitely many observations. Let us denote the number of these observations by n. Without losing generality, let us denote these observations by $y_0, y_1, \ldots, y_{n-1}$. Based on these observations, we need to find the corresponding values $x_0, x_1, \ldots, x_{n-1}$. To find these values, we need to solve the corresponding system of linear equations

$$y_k = \sum_{i=0}^{k} h_{k-i} \cdot x_i. \tag{2.2.2}$$

Standard methods of using systems of linear equations – e.g., standard Matlab procedures – are based on the matrix representation of the corresponding system. In our case, the system can be represented as

$$y = \mathbf{H} \cdot x,$$

where

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{bmatrix},$$

$$y = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix},$$

and

$$\mathbf{H} = \begin{bmatrix} h_0 & 0 & 0 & \dots & 0 \\ h_1 & h_0 & 0 & \dots & 0 \\ h_2 & h_1 & h_0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{n-1} & h_{n-2} & h_{n-3} & \dots & h_0 \end{bmatrix}.$$

The matrix **H** is known as the *convolution matrix*, or, to be more precise, the convolution matrix corresponding to the vector

$$h = \begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \\ h_{n-1} \end{bmatrix}.$$

In the convolution matrix **H**, each descending diagonal from left to right is constant. Such matrices are known as *diagonal-constant*, or *Toeplitz* matrices. Thus, convolution matrices are a particular case of Toeplitz matrices.

For images, to represent the corresponding linear systems of equations

$$y(i,j)=\sum_{i'}\sum_{j'}h(i-i',j-j')\cdot x(i',j') \tag{2.2.3}$$

in the matrix form, we need to represent all the values $x(i,j)$ as a single vector. This is usually done by lexicographically ordering the values corresponding to different locations (i,j) by rows. As a result, an $n\times m$ images is represented as a vector with $N=n\cdot m$ elements.

For example, a 3×3 image

$$\begin{bmatrix} x_{00} & x_{01} & x_{02} \\ x_{10} & x_{11} & x_{12} \\ x_{02} & x_{12} & x_{22} \end{bmatrix}$$

is transformed into the following vector with 9 elements:

$$x=\begin{bmatrix} x_{00} \\ x_{01} \\ x_{02} \\ x_{10} \\ x_{11} \\ x_{12} \\ \vdots \\ x_{22} \end{bmatrix}.$$

Similarly, the corresponding matrix describing the blurring

$$\begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{02} & h_{12} & h_{22} \end{bmatrix}$$

is transformed into the following vector with 9 elements:

$$h = \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ \vdots \\ h_{22} \end{bmatrix}.$$

The relation between the vectors x and y that represent the actual and the observed images takes the form

$$y = \mathbf{H}x$$

for an appropriate matrix $\mathbf{H}$.

In the 3×3 case, the matrix $\mathbf{H}$ has the form

$$\mathbf{H} = \begin{bmatrix} h_{00} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ h_{01} & h_{00} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ h_{02} & h_{01} & h_{00} & 0 & 0 & 0 & 0 & 0 & 0 \\ h_{10} & 0 & 0 & h_{00} & 0 & 0 & 0 & 0 & 0 \\ h_{11} & h_{10} & 0 & h_{01} & h_{00} & 0 & 0 & 0 & 0 \\ h_{12} & h_{11} & h_{10} & h_{02} & h_{01} & h_{00} & 0 & 0 & 0 \\ h_{20} & 0 & 0 & h_{10} & 0 & 0 & h_{00} & 0 & 0 \\ h_{21} & h_{20} & 0 & h_{11} & h_{10} & 0 & h_{01} & h_{00} & 0 \\ h_{22} & h_{21} & h_{20} & h_{12} & h_{11} & h_{10} & h_{02} & h_{01} & h_{00} \end{bmatrix}.$$

This matrix can be represented as

$$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{B} & \mathbf{A} & \mathbf{0} \\ \mathbf{C} & \mathbf{B} & \mathbf{A} \end{bmatrix},$$

where

$$\mathbf{A} = \begin{bmatrix} h_{00} & 0 & 0 \\ h_{01} & h_{00} & 0 \\ h_{02} & h_{01} & h_{00} \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} h_{10} & 0 & 0 \\ h_{11} & h_{10} & 0 \\ h_{12} & h_{11} & h_{10} \end{bmatrix},$$

and

$$\mathbf{C} = \begin{bmatrix} h_{20} & 0 & 0 \\ h_{21} & h_{20} & 0 \\ h_{22} & h_{21} & h_{20} \end{bmatrix}.$$

One can see that this is Toeplitz matrix of blocks, and each of the blocks **A**, **B**, and **C** is itself a Toeplitz matrix. Such matrices are known as *block Toeplitz matrices.*

The above reduction to matrices works well for small-length signals and small-size images. However, already for images of modest size 256×256, the corresponding convolution matrix will be of size 65,536×65,536, with billion entries. This size is too huge for usual computers, way above their limits on memory and speed.

To handle such images, it is beneficial to use Fourier transforms. The possibility to use Fourier transform comes from the known fact that the Fourier transform $Y(\omega)$ of the convolution $y = h \star x$ is equal to the product of the Fourier transforms:

$$Y(\omega) = H(\omega) \cdot X(\omega).$$

Thus, once we know y and h, we can:

- first compute their Fourier transforms $Y(\omega) = \mathcal{F}(x)$ and $H(\omega) = \mathcal{F}(h)$;
- then compute

$$X(\omega) = \frac{Y(\omega)}{H(\omega)};$$

- and, finally, we can reconstruct x by applying the inverse Fourier transform

$$x = \mathcal{F}^{-1}(X(\omega)).$$

Similarly, for 2-D images, we have

$$Y(\omega_1, \omega_2) = H(\omega_1, \omega_2) \cdot X(\omega_1, \omega_2).$$

Thus, once we know y and h, we can:

- first compute their Fourier transforms $Y(\omega_1, \omega_2) = \mathcal{F}(x)$ and $H(\omega_1, \omega_2) = \mathcal{F}(h)$;
- then compute
$$X(\omega_1, \omega_2) = \frac{Y(\omega_1, \omega_2)}{H(\omega_1, \omega_2)};$$
- and, finally, we can reconstruct x by applying the inverse Fourier transform

$$x = \mathcal{F}^{-1}(X(\omega_1, \omega_2)).$$

It is important to mention that while these formulas are exact for an infinite signal or an infinite image, for finite signals, the use of the above Fourier transform method does not exactly solve the original system $y = \mathbf{H}x$, but a similar system in which $\mathbf{H}$ is a *circular* matrix:

$$\mathbf{H} = \begin{bmatrix} h_0 & h_{n-1} & h_{n-2} & \dots & h_1 \\ h_1 & h_0 & h_{n-1} & \dots & h_2 \\ h_2 & h_1 & h_{n-1} & \dots & h_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h_{n-1} & h_{n-2} & h_{n-3} & \dots & h_0 \end{bmatrix}.$$

Similarly, for the 2-D 3×3 case, we have

$$\mathbf{H} = \begin{bmatrix} \mathbf{A} & \mathbf{C} & \mathbf{B} \\ \mathbf{B} & \mathbf{A} & \mathbf{C} \\ \mathbf{C} & \mathbf{B} & \mathbf{A} \end{bmatrix},$$

where

$$\mathbf{A} = \begin{bmatrix} h_{00} & h_{02} & h_{01} \\ h_{01} & h_{00} & h_{02} \\ h_{02} & h_{01} & h_{00} \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} h_{10} & h_{12} & h_{11} \\ h_{11} & h_{10} & h_{12} \\ h_{12} & h_{11} & h_{10} \end{bmatrix},$$

and

$$\mathbf{C} = \begin{bmatrix} h_{20} & h_{22} & h_{21} \\ h_{21} & h_{20} & h_{22} \\ h_{22} & h_{21} & h_{20} \end{bmatrix}.$$

2.3 Signal and Image Deconvolution in the Presence of Noise

Let us now consider the case when the noise is present. We assume that we know the statistical characteristics of the signal and of the noise. Specifically, we know that the signal and noise are independent and that their statistical properties do not change with time, i.e., that they are *stationary* random processes. We also assume that the mean values of the signal and the noise are 0. We assume that we know the power spectral densities

$$S_I(\omega) \stackrel{\text{def}}{=} \lim_{T\to\infty} E\left[\frac{1}{T}\cdot |X_T(\omega)|^2\right] = \lim_{T\to\infty} E\left[\frac{1}{T}\cdot X_T^*(\omega)\cdot X_T(\omega)\right]$$

and

$$S_N(\omega) \stackrel{\text{def}}{=} \lim_{T\to\infty} E\left[\frac{1}{T}\cdot |N_T(\omega)|^2\right] = \lim_{T\to\infty} E\left[\frac{1}{T}\cdot N_T^*(\omega)\cdot N_T(\omega)\right],$$

where $X_T(\omega)$ and $N_T(\omega)$ are the Fourier transforms of the signal and noise restricted to the interval $\left[-\frac{T}{2}, \frac{T}{2}\right]$.

It is reasonable to use the Least Squares approach to find an estimate

$$\widehat{X}_T(\omega) = K(\omega)\cdot Y_T(\omega)$$

for $X_T(\omega)$. According to this approach, we should find the coefficients $K(\omega)$ for which the expected value of the mean square difference

$$D \stackrel{\text{def}}{=} \lim_{T\to\infty} \frac{1}{T} \cdot \int_{-T/2}^{T/2} (\widehat{x}(t) - x(t))^2\,dt$$

is the smallest possible.

Due to the Parseval's Theorem, the means square difference is equal to the same difference between the Fourier transforms, i.e., to the value

$$D = \lim_{T\to\infty} \frac{1}{T} \cdot \int |\widehat{X}_T(\omega) - X_T(\omega)|^2\,d\omega.$$

Substituting

$$\widehat{X}_T(\omega) = K(\omega) \cdot Y_T(\omega) = K(\omega) \cdot (H(\omega) \cdot X_T(\omega) + N_T(\omega))$$

into this expression, we get

$$D = \lim_{T\to\infty} \frac{1}{T} \cdot \int |K(\omega) \cdot (H(\omega) \cdot X_T(\omega) + N_T(\omega)) - X_T(\omega)|^2\,d\omega =$$

$$\lim_{T\to\infty} \frac{1}{T} \cdot \int |(K(\omega) \cdot H(\omega) - 1) \cdot X_T(\omega) + K(\omega) \cdot N_T(\omega)|^2\,d\omega.$$

Since $|z|^2 = z^* \cdot z$, we get

$$D = \lim_{T\to\infty} \frac{1}{T} \cdot \int ((K^*(\omega) \cdot H^*(\omega) - 1) \cdot X_T^*(\omega) +$$

$$K^*(\omega) \cdot N_T^*(\omega)) \cdot ((K(\omega) \cdot H(\omega) - 1) \cdot X_T(\omega) + K(\omega) \cdot N_T(\omega))\,d\omega.$$

The expected value of the sum or an integral is equal to the sum or the integral of expected values. So, the expected value of the above limit is equal to the following expression:

$$\lim_{T\to\infty} \int \left((K^*(\omega) \cdot H^*(\omega) - 1) \cdot (K(\omega) \cdot H(\omega) - 1) \cdot E\left[\frac{1}{T} \cdot X^*(\omega) \cdot X(\omega)\right] + \right.$$

$$\left. K^*(\omega) \cdot K(\omega) \cdot E\left[\frac{1}{T} \cdot N^*(\omega) \cdot N(\omega)\right] \right) d\omega;$$

we took into account that the signal and the noise are independent and have 0 means, hence

$$E[X_T(\omega) \cdot N_T(\omega)] = E[X_T(\omega)] \cdot E[N_T(\omega)] = 0 \cdot 0 = 0.$$

We know the values

$$S_I(\omega) = \lim_{T\to\infty} E\left[\frac{1}{T}\cdot X_T^*(\omega)\cdot X_T(\omega)\right]$$

and

$$S_N(\omega) = \lim_{T\to\infty} E\left[\frac{1}{T}\cdot N_T^*(\omega)\cdot N_T(\omega)\right],$$

so the above limit takes the form

$$\int [(K^*(\omega)\cdot H^*(\omega)-1)\cdot (K(\omega)\cdot H(\omega)-1)\cdot S_I(\omega) + K^*(\omega)\cdot K(\omega)\cdot S_N(\omega))]\,d\omega.$$

Differentiating this expression with respect to $K^*(\omega)$ and equating the derivative to 0, we conclude that

$$(K(\omega)\cdot H(\omega)-1)\cdot H^*(\omega)\cdot S_I(\omega) + K(\omega)\cdot S_N(\omega) = 0,$$

so

$$K(\omega)\cdot (|H(\omega)|^2\cdot S_I(\omega) + S_N(\omega)) = H^*(\omega)\cdot S_I(\omega)$$

and

$$K(\omega) = \frac{H^*(\omega)\cdot S_I(\omega)}{|H(\omega)|^2\cdot S_I(\omega) + S_N(\omega)}.$$

Dividing both the numerator and the denominator of this expression by $S_I(\omega)$, we get the known formula for the Wiener filter:

$$K(\omega) = \frac{H^*(\omega)}{|H(\omega)|^2 + \dfrac{S_N(\omega)}{S_I(\omega)}}.$$

Similarly, for 2-D images, we get

$$\widehat{X}(\omega_1,\omega_2) = K(\omega_1,\omega_2)\cdot Y(\omega_1,\omega_2),$$

where

$$K(\omega_1,\omega_2) = \frac{H^*(\omega_1,\omega_2)}{|H(\omega_1,\omega_2)|^2 + \dfrac{S_N(\omega_1,\omega_2)}{S_I(\omega_1,\omega_2)}}.$$

For both signal and image processing, one can see that in the no-noise case, when $S_N(\omega) = 0$, we get exactly the above formulas

$$X(\omega) = \frac{Y(\omega)}{H(\omega)}$$

and

$$X(\omega_1, \omega_2) = \frac{Y(\omega_1, \omega_2)}{H(\omega_1, \omega_2)}.$$

As the noise increases, the coefficient $K(\omega)$ decreases.

Wiener filter is based on using the Least Squares method, which is optimal for Gaussian distributions. In practice, the distribution is often non-Gaussian. For example, an image is often well described by a Poisson process. In such non-Gaussian situations, a more accurate estimate for the original signal comes from using the Maximum Likelihood approach (which for Gaussian distributions leads to Least Squares). Such methods are described, e.g., in [39].

2.4 Blind Image Deconvolution in the Presence of Prior Knowledge

The above methods assumed that we know the blurring function h. In practice, we often do not have a complete knowledge about the blurring function. In such situations, we need to perform a *blind deconvolution*, i.e., reconstruct both the original signal x and the blurring function h.

In many cases, while we do not have a *complete* information about the blurring function, we have some *partial* knowledge about this function (and about the image). Let us describe blurring deconvolution techniques in the presence of such prior knowledge. This knowledge is usually of two types.

In some situations, we know that the actual blurring function (and the image) belongs to a certain class of functions. For example, we know that the blurring function is smooth, and we know the bounds on its smoothness, e.g., we know the bound on the mean squared derivative: $\int(\dot{h}(t))^2\,dt \le C$. We usually also know similar restrictions on the image. In this case, we can use regularization techniques, i.e., find the image x and the blurring function h for which the mean square error is the smallest under the known constraints; see, e.g., [9]. In addition, we may know that the blurring function belongs to a known finite-parametric family of functions.

In other situations, we know the class of blurring functions, but we know a prior probability distribution on the set of all possible blurring functions (and images). Specifically, we have a finite-parametric family of distributions $p(h\,|\,\Omega)$ and $p(x\,|\,\Omega)$ characterized by some parameters Ω. We know prior probabilities $p(\Omega)$ of different values of these parameters, and we know the probabilities of different noise values, i.e., we know the probabilities $p(y\,|\,x,h,\Omega)$. Based on these probabilities, we can then determine the joint distribution of Ω, x, h, and y as

$$p(\Omega,x,h,y) = p(\Omega)\cdot p(x\,|\,\Omega)\cdot p(h\,|\,\Omega)\cdot p(y\,|\,x,h,\Omega);$$

in this formula, we took into account that the signal x and the blur have different origins and can, thus be considered statistically independent:

$$p(x,h\,|\,\Omega) = p(x\,|\,\Omega)\cdot p(h\,|\,\Omega).$$

Once we know this joint distribution, then, based on the observations y, we can compute the most probable values of the parameters

$$\widehat{\Omega} = \arg\max_{\Omega} p(\Omega\,|\,y) = \int\int_{x,h} p(\Omega,x,h,y)\,dx\,dh.$$

We can now use this estimate $\widehat{\Omega}$ to find estimates $\widehat{x}$ and $\widehat{h}$ for which

$$(\widehat{x},\widehat{h}) = \arg\max_{x,h} p(x,h\,|\,\widehat{\Omega},y);$$

see, e.g., [30] for details.

2.5 Blind Image Deconvolution in the Absence of Prior Knowledge: Need to Use Sparsity-Based Techniques

In the previous section, we described image deconvolution techniques in situations when we have a partial knowledge of the blurring function. In many practical situations, we do not have such prior knowledge. In such situation, what often helps is the observation that the observed image has a *sparsity property*: namely, if we represent the observed image as a linear combination of functions from an appropriate basis (e.g., an appropriate wavelet basis), then most of the coefficients $a = (a_1, a_2, \ldots)$ in this representation will be 0s (or almost 0s).

In such situations, we have a bound on the number of non-zero coefficients: $\|a\|_0 \leq c$, where $\|a\|_0 \stackrel{\text{def}}{=} \#\{i : a_i \neq 0\}$ and c is a known constant.

In general, once we have a constraint $f(a) \leq c$, then the original problem of minimizing an objective function J gets transformed into a constrained optimization problem of minimizing the objective function J under this constraint $f(a) \leq c$. To solve this constraint optimization problem, we can use the Lagrange multiplier approach, according to which the above constraint optimization problem is equivalent to minimizing a function

$$J + \lambda \cdot f(a)$$

for some constant λ (known as *Lagrange multiplier*) which needs to be determined from the condition that $f(a) \leq c$.

For $f(a) = \|a\|_0$, no efficient algorithms are known for minimizing such an objective function: most efficient optimization algorithms require that the objective function be differentiable (or at least continuous), and the expression $\|a\|_0$ is not even continuous. The good news is that under some reasonable conditions, minimizing this function is equivalent to minimizing the similar expression

$$J + \lambda \cdot \|a\|_1$$

with a continuous ℓ^1-norm $\|a\|_1 \stackrel{\text{def}}{=} \sum\limits_i |a_i|$ instead of the discontinuous expression $\|a\|_0$; see, e.g., [4]. The ℓ_1-norm is convex, so if J is also convex (and it often is), then we get an additional advantage of being able to use known algorithms for minimizing convex functions.

2.6 State-of-the-Art Techniques for Sparsity-Based Blind Image Deconvolution

In [3], the following algorithm was proposed to solve the blind deconvolution problem. We know that $y \approx \mathbf{H}x$, we also know that y has the sparsity property, i.e., that $y \approx \mathbf{W}a$, where $\mathbf{W}$ is the matrix describing the corresponding decomposition (e.g., into wavelets), and the vector a is sparse. We also impose additional restrictions $R_1(x) \le \text{const}$ and $R_2(h) \le \text{const}$ that imply that x and h are sufficiently smooth.

Since $y \approx \mathbf{W}a$, the condition $y \approx \mathbf{H}x$ can be equivalently described as $\mathbf{W}a \approx \mathbf{H}x$. The least square approach thus leads us to minimizing the value $\|y-\mathbf{W}a\|_2^2$ under the constraints that $\mathbf{W}a \approx \mathbf{H}x$ (i.e., that $\|\mathbf{W}a - \mathbf{H}x\|_2^2 \le \text{const}$), that a is sparse (i.e., that $\|a\|_1 \le \text{const}$), that $R_1(x) \le \text{const}$, and that $R_2(h) \le \text{const}$. Applying the Lagrange multiplier technique to this constraint optimization problem, we can reduce it to the unconstrained optimization problem of minimizing

$$Q(a,x,h) \stackrel{\text{def}}{=} \frac{\beta}{2} \cdot \|y - \mathbf{W}a\|_2^2 + \frac{\eta}{2} \cdot \|\mathbf{W}a - \mathbf{H}x\|_2^2 + \tau \cdot \|a\|_1 + \alpha \cdot R_1(x) + \gamma \cdot R_2(h), \quad (2.6.1)$$

for appropriate parameters β, η, τ, α, and γ.

Specifically, the authors select

$$R_1(x) = \sum_{d \in D} 2^{1-o(d)} \sum_i |\Delta_i^p(x)|^p, \quad (2.6.2)$$

where $o(d) \in \{1,2\}$ is the order of the difference operator $\Delta_i^p(\mathbf{x})$, $0 < p < 1$, and $d \in D = \{h, v, hh, vv, hv\}$. Here, Δ_i^h and Δ_i^v correspond, respectively to horizontal and vertical first order differences at pixel i:

$$(\Delta^h x)(n_x, n_y) \overset{\text{def}}{=} x(n_x, n_y) - x(n_x - 1, n_y)$$

and

$$(\Delta^v x)(n_x, n_y) \overset{\text{def}}{=} x(n_x, n_y) - x(n_x, n_y - 1).$$

and the operators Δ_i^{hh}, Δ_i^{vv}, Δ_i^{hv} correspond to second order horizontal, vertical, and horizontal-vertical differences at pixel i:

$$\Delta_i^{hh}(x) \overset{\text{def}}{=} \Delta_i^h(\Delta_i^h(x)),$$

$$\Delta_i^{vv}(x) \overset{\text{def}}{=} \Delta_i^v(\Delta_i^v(x)),$$

and

$$\Delta_i^{hv}(x) \overset{\text{def}}{=} \Delta_i^h(\Delta_i^v(x)).$$

The authors also select

$$R_2(h) = \|\mathbf{C}h\|^2, \tag{2.6.3}$$

where $\mathbf{C}$ is the circulant matrix that represents the convolution with the discrete Laplacian operator:

$$(\mathbf{C}h)(n_x, n_y) = h(n_x - 1, n_y) + h(n_x + 1, n_y) + h(n_x, n_y - 1) + h(n_x, n_y + 1) - 4h(n_x, n_y).$$

Our goal is to find the values x, h, and a that minimize the objective function Q. The algorithm for optimizing this objective function is iterative. It starts with some first approximations to the blur. Then, the algorithm interchangingly uses two steps:

- first, we fix a and find h and x that minimize Q;
- then, they fix x and h and find a that minimizes Q.

The process stops when the images x^k and x^{k-1} on the two consequent iterations are sufficiently close to each other, i.e., when

$$\frac{\|x^k - x^{k-1}\|}{\|x^{k-1}\|} < \varepsilon$$

for some pre-determined small threshold $\varepsilon > 0$. In [3], the authors select $\varepsilon = 0.01$.

To minimize over a, the authors use an *l1-ls* method described in [23]. Minimization over h is easy, since the objective function Q is quadratic in h and thus, we can differentiate with respect to h, equate derivatives to 0, and get a system of linear equations for determining h.

For x, the situation is not so simple, since in addition to the quadratic term proportional to $\|\mathbf{W}a-\mathbf{H}x\|^2$, we also have a non-quadratic term $R_1(x)$ that includes terms proportional to $|L(x)|^p$ for some linear operators L. To perform the corresponding minimization, the authors take into account that this non-quadratic term can be represented as

$$|L(x)|^p = \frac{|L(x)|^2}{|L(x)|^{2-p}}.$$

Thus, to minimize this expression over x, we can perform the following iterative approach: we start with some initial value x, and then on each $(\ell+1)$-th iteration, we minimize the quadratic expression

$$\frac{|L(x)|^2}{|L(x^\ell)|^{2-p}},$$

where in the denominator, we use the value x^ℓ from the previous iteration. By explicitly differentiating the corresponding expressions and equating the derivatives to 0, we arrive at the following algorithm.

First, we select α, β, γ, τ, θ, η^1, and $0 < p < 1$. We then select some initial estimate $h^{1,1}$ of the blur, and the initial values of an auxiliary vector $v_d^{1,1}$, where $d \in D = \{h, v, hh, vv, hv\}$.

The simplest idea is to select each of the components of each initial vector $v_d^{1,1}$ by using a random number generator that generates numbers uniformly distributed on the interval $[0,1]$.

Then, for $k = 1, 2, \ldots$, we perform the following until the above stopping criterion is met:

1. For $\ell = 1, \ldots, L_0$ for some L_0:

(a) Compute

$$x^{k,l+1} = \left[\eta^k (\mathbf{H}^{k,l})^T (\mathbf{H}^{k,l}) + \alpha p \sum_{d \in D} 2^{1-o(d)} (\boldsymbol{\Delta}^d)^T \mathbf{B}_d^{k,l} \boldsymbol{\Delta}^d \right]^{-1} \cdot \eta^k (\mathbf{H}^{k,l})^T \mathbf{W} a^k,$$

where $B_d^{k,\ell}$ is a diagonal matrix with entries $B_d^{k,\ell}(i,i) = \left(v_{d,i}^{k,\ell}\right)^{p/2-1}$ and $\boldsymbol{\Delta}^d$ is the convolution matrix of the difference operator $\Delta_i^d(\cdot)$. For solving this system of linear equations, the authors use the Fourier transform approach.

(b) Compute

$$h^{k,l+1} = \left[\eta^k (\mathbf{X}^{k,l})^T (\mathbf{X}^{k,l}) + \gamma \mathbf{C}^T \mathbf{C}\right]^{-1} \cdot \eta^k (\mathbf{X}^{k,l})^T \mathbf{W} a^k,$$

where $\mathbf{X}^{k,\ell}$ is the convolution matrix of the image $x^{k,\ell}$. For solving this system of linear equations, the authors also use the Fourier transform approach.

(c) For each $d \in D = \{h, v, hh, vv, hv\}$, calculate

$$v_{d,i}^{k,\ell+1} = \left[\Delta_i^d(x^{k,\ell})\right]^2.$$

2. Set $x^k = x^{k,\ell+1}$, $h^k = h^{k,\ell+1}$, and $v_{d,i}^k = v_{d,i}^{k,\ell+1}$.

3. Now, we need to find a^{k+1} by minimizing the expression

$$\frac{\beta}{2} \cdot \|y - \mathbf{W}a\|_2^2 + \frac{\eta}{2} \cdot \|\mathbf{W}a - \mathbf{H}x\|_2^2 + \tau \cdot \|a\|_1.$$

This can be done by applying the *l1-ls* algorithm for minimizing the equivalent minimization problem

$$\|y' - \boldsymbol{\Phi}' \mathbf{W} a^k\|^2 + \tau \|a^k\|_1,$$

where

$$y' = \begin{bmatrix} \sqrt{\frac{\beta}{2}} y \\ \sqrt{\frac{\eta^k}{2}} \mathbf{H}^k x^k \end{bmatrix}$$

and

$$\boldsymbol{\Phi}' = \begin{bmatrix} \sqrt{\frac{\beta}{2}} \mathbf{I} \\ \sqrt{\frac{\eta^k}{2}} \mathbf{I} \end{bmatrix}.$$

4. Set $\eta^{k+1} = \theta \eta^k$.

Chapter 3

Open Problems Related to Blind Image Deconvolution: Need for Theoretical Justification and Need for Improvement of the Existing Techniques

While the state-of-the-art blind image deconvolution techniques, as described in [3], are very successful in blindly deconvolving images, these techniques have several limitations.

3.1 Need for Theoretical Justification

While the state-of-the-art techniques work well on several examples, there is no guarantee that these methods will work well on other examples – because the success of these techniques is purely empirical, it has no fundamental theoretical explanation. Because of this lack of theoretical explanation, practitioners are somewhat reluctant to use these advanced techniques.

To make these techniques more widely used, and to guarantee that these techniques indeed lead to reasonable results, it is thus desirable to come up with a theoretical explanation for these techniques. Specifically, two things need to be explained:

- why we can use sparsity-based techniques in the first place, and
- why l^p-methods are useful here.

A theoretical explanation of both facts is given in this dissertation, in Chapters 4 and 5.

3.2 Need for Improvement

The current state-of-the-art method for blind image deconvolution is based on minimizing the sum $|\Delta_x I_{i,j}|^p + |\Delta_y I_{i,j}|^p$ for some $p < 2$, where $\Delta_x I_{i,j} \stackrel{\text{def}}{=} I_{i,j} - I_{i-1,j}$, and $\Delta_y I_{ij} \stackrel{\text{def}}{=} I_{i,j} - I_{i,j-1}$. This is a discrete analog of the term

$$\left|\frac{\partial I}{\partial x}\right|^p + \left|\frac{\partial I}{\partial y}\right|^p.$$

In the traditional least squares approach, when $p = 2$, the corresponding expression

$$\left|\frac{\partial I}{\partial x}\right|^2 + \left|\frac{\partial I}{\partial y}\right|^2$$

is rotation-invariant: namely, it describes the square of the length of the gradient vector

$$\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right).$$

However, for $p \neq 2$, the corresponding expression is no longer rotation-invariant.

Since the current state-of-the-art technique is not rotation-invariant, the result of using this technique may change with rotation. Thus, if we rotate the image a little bit, the method, in general, leads to a different deconvolution result. So, even when the original reconstruction is optimal, the reconstruction of a rotated image will be different and, thus, not optimal.

To improve the quality of image decomposition, it is therefore desirable to modify the current state-of-the art techniques by making them rotation-invariant. This will be done in Chapter 6.

Chapter 4

Theoretical Justification of Sparsity-Based Techniques in Blind Image Deconvolution

In many practical applications, it turned out to be efficient to assume that the signal or an image is *sparse*, i.e., that when we decompose it into appropriate basis functions (e.g., sinusoids or wavelets), most of the coefficients in this decomposition will be zeros. At present, the empirical efficiency of sparsity-based techniques remains somewhat a mystery. In this chapter, we show that fuzzy-related techniques can explain this empirical efficiency. A similar explanation can be obtained by using probabilistic techniques; this fact increases our confidence that our explanation is correct.

4.1 Formulation of the Problem

4.1.1 Sparsity-based techniques are useful

In many practical applications, it turned out to be efficient to assume that the signal or an image is sparse; see, e.g., [3, 10, 11, 12, 13, 17, 18, 19, 20, 27, 29, 31, 38, 41].

In precise terms, sparsity means that when we decompose the original signal $x(t)$ (or original image) into appropriate basis functions $e_1(t)$, $e_2(t)$, $\ldots$ (e.g., sinusoids or wavelets), i.e., represent this signal (or image) as a linear combination $x(t) = \sum\limits_{i=1}^{\infty} a_i \cdot e_i(t)$, then most of the coefficients a_i in this decomposition will be zero.

Moreover, it is usually beneficial to select, among all the signals which are consistent with all the observations (and will all additional knowledge), the signal for which:

- either the number of non-zero coefficients is the smallest possible,
- or, more generally, the "weighted number" of such coefficient is the smallest possible, where the weighted number is defined as $\sum\limits_{i:a_i\neq 0} w_i$, for some weights $w_i > 0$.

Comment. Sparsity can be viewed as a particular case of the Occam's razor, according to which we should always select the simplest model that fits the observation.

4.1.2 But why are sparsity-related techniques useful?

At present, the empirical efficiency of sparsity-based techniques remains somewhat a mystery.

4.1.3 What we do in this chapter

In this chapter, we show that fuzzy-related techniques can explain this empirical efficiency.

We also show that a similar explanation can be obtained by using probabilistic techniques; this fact increases our confidence that our explanation is correct.

4.2 General Analysis of the Problem

4.2.1 Why do we need data processing in the first place?

To better understand why different techniques are more or less empirically successful in data processing, it is important to recall why we need data processing in the first place.

One of the main goals of science and engineering is to predict the future state of the world, and to design gadgets and strategies that would make the future state of the world more beneficial for us.

To predict the state of the world, we need to know the current state of the world, and we need to know how this state changes in time. In general, the state of the world can be described by the numerical values of different physical quantities. In these terms, to predict the future state of the world means to predict the future values of the corresponding quantities $y_1, \ldots, y_m$.

In each practical problem, we are usually interested only in a small number of quantities. However, to predict the future values of these quantities, we often need to know the initial values of some auxiliary quantities as well. For example, when we launch a spaceship, we are interested in its location and direction when it leaves the atmosphere, and we are not directly interested in the future values of the winds on different heights. However, these winds affect the spaceship's trajectory, and, as a result, we need to know their initial values to correctly predict the desired values $y_1, \ldots, y_m$. In general, we need to know $n \gg m$ initial values $x_1, \ldots, x_n$ to make the desired prediction.

The relation between x_i and y_j is often complicated. So, to predict the values $y_1, \ldots, y_m$ based on the inputs $x_1, \ldots, x_n$, we need to apply complex computer-based algorithms, i.e., perform *data processing.*

The above description captures the main reason for data processing, but it is somewhat oversimplified, since it assumes that we know the values $x_1, \ldots, x_n$ of the original quantities. For some quantities, this is indeed true, since we can directly measure their values. However, there are many other quantities which are difficult to measure directly. For example, when we are trying to predict the state of the engine, it is desirable to know the current temperature inside; however, this temperature is difficult to measure directly. If we cannot directly measure a certain value x_i, a natural idea is to find easier-to-measure auxiliary quantities $z_1, \ldots, z_p$ which are related to x_i by a known dependence, and then use this known dependence to reconstruct x_i. The corresponding computations may be complex, so we have another reason why data processing is needed.

4.2.2 Before we perform data processing, we first need to know which inputs are relevant

In general, in data processing, we estimate the value of the desired quantity y_j based on the values of the known quantities $x_1, \ldots, x_n$ that describe the current state of the world.

In principle, all possible quantities $x_1, \ldots, x_N$ that describe the current state of the world could be important for predicting some future quantities. However, for each specific quantity y_j, usually, only a few of the quantities x_i are actually useful.

So, before we decide how to transform the inputs x_i into the desired output, we first need to check which inputs are actually useful. This checking is a very important stage of data processing – if we do not filter out unnecessary quantities x_i, we will waste time and resources measuring and processing these unnecessary quantities.

4.3 Analysis of the Problem: Let Us Use Fuzzy-Related Techniques

4.3.1 Description of our problem in natural-language terms

We are interested in a reconstructing a signal or image $x(t) = \sum\limits_{i=1}^{\infty} a_i \cdot e_i(t)$ based on the measurement results and prior knowledge. In this formula, the basis functions $e_i(t)$ are known, and the coefficients a_i need to be determined. Based on measurement results and prior knowledge, we need to estimate the values a_i.

This reconstruction problem is, of course, a particular case of a general data processing problem. As we have mentioned in the previous section, a natural way to approach data processing problems in general is that:

- first, we find out which quantities are important for this particular problem, and
- then, we find the values of the important quantities.

In the above data processing problem, the quantities are the coefficients a_i. The quantity a_i is irrelevant if it does not affect the resulting signal, i.e., if $a_i = 0$. When $a_i \neq 0$, this means that this quantity affects the resulting signal $x(t) = \sum\limits_{i=1}^{\infty} a_i \cdot e_i(t)$ and is, therefore, relevant. Thus, for our problem, the above two-stage data processing process takes the following form:

- first, we decided which values a_i are zeros and which are non-zeros, and
- then, we use an appropriate data processing algorithm to estimate the numerical values of non-zero coefficients a_i.

On the first stage, we can make several different decisions, all of which are consistent with the measurements and with the prior knowledge. For example, if in one decision, we take $a_i = 0$, then taking a_i to be very small but still different from 0 will still make this slightly modified signal consistent with all the measurement results. Out of all such possible decisions, we need to select *the most reasonable one.*

"Reasonable" is not a precise term. So, to be able to solve the problem, we need to translate this imprecise natural-language description into precise terms.

4.3.2 Fuzzy techniques can translate this natural-language description into a precisely formulated problem

In order to translate the above natural-language problem into precise terms, it is reasonable to use techniques specifically designed for such translations – namely, the techniques of *fuzzy logic*; see, e.g., [24, 33, 43].

In fuzzy logic, the meaning of each imprecise ("fuzzy") natural-language statement $P(x)$ about a quantity x is described by assigning, to each possible value x, the degree $\mu_P(x) \in [0, 1]$ to which we are sure that x satisfies this property P. For simple properties, we can determine these values, e.g., by simply asking the experts to mark, on a scale from 0 to 10, how much they are sure that P holds for x; if an expert marks the number 7, we take $\mu_P(x) = 7/10$.

This can be done for properties that depend on a single quantity. However, for properties like "reasonable", that depend on many values $a_1, \ldots, a_n, \ldots$, it is not feasible to ask the expert for the degrees corresponding to all possible combinations of the values a_i. In such situations, we can use the fact that from the commonsense viewpoint, a sequence $(a_1, a_2, \ldots)$ is reasonable if and only if a_1 is reasonable *and* a_2 is reasonable, *and* … For each of the quantities a_i, we can elicit, from the expert, degree to which different values of a_i are reasonable.

Since this is all the information that we have, we need to estimate the degree to which a_1 is reasonable and a_2 is reasonable, and …, based on the degrees to which a_1 is reasonable, to which a_2 is reasonable, etc. In other words, we know the degrees of belief $a = d(A)$ and $b = d(B)$ in statements A and B, and we need to estimate the degree of belief in the composite statement $A \,\&\, B$.

It is worth mentioning that this *estimate* cannot be always *exact*, because our degree of belief in a composite statement $A \,\&\, B$ depends not only on our degrees of belief in A and B, it also depends on the (usually unknown) dependence between A and B. Let us give an example.

- If A is "coin falls heads", and B is "coin falls tails", then for a fair coin, degrees a and b are equal: $a = b$. Here, $A \,\&\, B$ is impossible, so our degree of belief in $A \,\&\, B$ is zero: $d(A \,\&\, B) = 0$.
- However, if we take $A' = B' = A$, then $A' \,\&\, B'$ is simply equivalent to A, so we still have $a' = b' = a$ but this time $d(A' \,\&\, B') = a > 0$.

In these two cases, $d(A') = d(A)$, $d(B') = d(B)$, but $d(A \,\&\, B) \neq d(A' \,\&\, B')$.

In general, let $f_\&(a, b)$ be the estimate for $d(A \,\&\, B)$ based on the known values $a = d(A)$ and $b = d(B)$. The corresponding function $f_\&(a, b)$ must satisfy some reasonable properties: e.g.,

- since $A \,\&\, B$ means the same as $B \,\&\, A$, this operation must be commutative;

- since $(A \,\&\, B) \,\&\, C$ is equivalent to $A \,\&\, (B \,\&\, C)$, this operation must be associative, etc.

Operations with these properties are known as *"and"-operations*, or, alternatively, *t-norms.*

Let us apply an appropriate t-norm to our problem. In our case, for each variable a_i, we only need to find the degrees of belief in two situations: that $a_i = 0$ and that $a_i \neq 0$. Let us denote the degree to which it is reasonable to believe that $a_i = 0$ by $d_i^{=}$, and the degree to which it is reasonable to believe that $a_i \neq 0$ by $d_i^{\neq}$. Thus, we arrive at the following formulation of the first stage of data processing.

4.3.3 Resulting precise formulation of the first stage of data processing in precise terms

Our goal is to select a sequence $(\varepsilon_1, \varepsilon_2, \ldots)$, where each ε_i is equal either to $=$ or to $\neq$. If ε_i is $=$, this means that we have decided that $a_i = 0$, and if ε_i is $\neq$, this means that we have decided that $a_i \neq 0$.

For each such sequence $\varepsilon = (\varepsilon_1, \varepsilon_2, \ldots)$, we can determine the degree $d(\varepsilon)$ to which this sequence is reasonable, by applying the selected t-norm $f_\&(a, b)$ to the degrees $d_i^{\varepsilon_i}$ to which we belief that each choice ε_i is reasonable:

$$d(\varepsilon) = f_\&(d_1^{\varepsilon_1}, d_2^{\varepsilon_2}, \ldots).$$

Out of all sequences ε which are consistent with the measurements and with the prior knowledge, we must select the one for which this degree of belief is the largest possible.

4.3.4 An additional fact that we can use

If we have no information about the signal, i.e., in other words, if there is no evidence that there is a non-zero signal, then the most reasonable choice is to select $x(t) = 0$, i.e., to select a signal for which $a_1 = a_2 = \ldots = 0$.

In other words, if we do not have any way to impose restrictions on the sequence ε, then the most reasonable should be the sequence $(=, =, \ldots)$.

Similarly, the worst reasonable is the sequence in which we take all the values into account, i.e., the sequence $(\neq, \ldots, \neq)$.

4.3.5 A comment about t-norms

In principle, there are many possible t-norms. However, it is known (see, e.g., [32]) that an arbitrary continuous t-norm can be approximated, with an arbitrary accuracy, by an *Archimedean* t-norm, i.e., by a t-norm of the type $f_\&(a, b) = f^{-1}(f(a) \cdot f(b))$, for some continuous strictly increasing function $f(x)$. Thus, without losing generality, we can assume that the actual t-norm is Archimedean.

Now, we are ready to formulate and solve the corresponding problem.

4.4 Definitions and the Main Result: Fuzzy-Related Techniques Explain Sparsity

Definition 1.

- *By a* t-norm*, we means a function* $f_\& : [0, 1] \times [0, 1] \to [0, 1]$ *of the form* $f_\&(a, b) = f^{-1}(f(a) \cdot f(b))$*, where*
  $f : [0, 1] \to [0, 1]$ *is a continuous strictly increasing function for which* $f(0) = 0$ *and* $f(1) = 1$*.*
- *By a* sequence*, we mean a sequence* $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_N)$*, where each symbol* ε_i *is equal either to* $=$ *or to* $\neq$*.*
- *Let* $d^= = (d^=_1, \ldots, d^=_N)$ *and* $d^{\neq} = (d^{\neq}_1, \ldots, d^{\neq}_N)$ *be sequences of real numbers from the interval* $[0, 1]$*. For each sequence* ε*, we define its* degree of reasonableness *as* $d(\varepsilon) \stackrel{\text{def}}{=} f_\&(d^{\varepsilon_1}_1, \ldots, d^{\varepsilon_N}_N)$*.*

- *We say that the sequences* $d^=$ *and* $d^{\neq}$ properly describe reasonableness *if the following two conditions are satisfied:*
  - *the sequence* $\varepsilon_= \stackrel{\text{def}}{=} (=, \ldots, =)$ *is more reasonable than all others, i.e.,* $d(\varepsilon_=) > d(\varepsilon)$ *for all* $\varepsilon \neq \varepsilon_=$*, and*
  - *the sequence* $\varepsilon_{\neq} \stackrel{\text{def}}{=} (\neq, \ldots, \neq)$ *is less reasonable than all others, i.e.,* $d(\varepsilon_{\neq}) < d(\varepsilon)$ *for all* $\varepsilon \neq \varepsilon_{\neq}$.
- *For each set* S *of sequences, we say that a sequence* $\varepsilon \in S$ is the most reasonable *if its degrees of reasonableness is the largest possible, i.e., if* $d(\varepsilon) = \max\limits_{\varepsilon' \in S} d(\varepsilon')$.

Proposition 1. *Let us assume that the sequences* $d^=$ *and* $d^{\neq}$ *properly describe reasonableness. Then, there exist weights* $w_i > 0$ *for which, within each set* S*, a sequence* $\varepsilon \in S$ *is the most reasonable if and only if for this sequence, the sum* $\sum\limits_{i:\varepsilon_i = \neq} w_i$ *is the smallest possible.*

Discussion. In other words, a sequence is the most reasonable if and only if the sum $\sum\limits_{i:a_i \neq 0} w_i$ attains the smallest possible value. Thus, fuzzy-based techniques indeed naturally lead to the sparsity condition.

Proof. By definition of the t-norm, we have

$$d(\varepsilon) = f_\&(d_1^{\varepsilon_1}, \ldots, d_N^{\varepsilon_N}) = f^{-1}(f(d_1^{\varepsilon_1}) \cdot \ldots \cdot f(d_N^{\varepsilon_N})),$$

i.e.,

$$d(\varepsilon) = f_\&(d_1^{\varepsilon_1}, \ldots, d_N^{\varepsilon_N}) = f^{-1}(e_1^{\varepsilon_1} \cdot \ldots \cdot e_N^{\varepsilon_N}),$$

where we denoted $e_i^{\varepsilon_i} \stackrel{\text{def}}{=} f(d_i^{\varepsilon_i})$.

Since the continuous function $f(x)$ is strictly increasing, its inverse $f^{-1}(x)$ is also strictly increasing. Thus, maximizing $d(\varepsilon)$ is equivalent to maximizing the function $e(\varepsilon) \stackrel{\text{def}}{=} f(d(\varepsilon))$. This function has the form

$$e(\varepsilon) = f(d(\varepsilon)) = f(f^{-1}(e_1^{\varepsilon_1} \cdot \ldots \cdot e_N^{\varepsilon_N})),$$

i.e., the form

$$e(\varepsilon) = e_1^{\varepsilon_1} \cdot \ldots \cdot e_N^{\varepsilon_N}.$$

From the condition that the sequences $d^=$ and $d^{\neq}$ properly describe reasonableness, it follows, in particular, that for each i, we have $d(\varepsilon_=) > d(\varepsilon_=^{(i)})$, where

$$\varepsilon_=^{(i)} \stackrel{\text{def}}{=} (=, \ldots, =, \neq \text{ (on } i\text{-th place)}, =, \ldots, =).$$

This inequality is equivalent to $e(\varepsilon_=) < e(\varepsilon_=^{(i)})$.

Since the values $e(\varepsilon)$ are simply the products, we thus conclude that

$$\prod_{j=1}^{N} e_j^= > \left(\prod_{j\neq i} e_j^=\right) \cdot e_i^{\neq}.$$

The values $e_j^=$ corresponding to $j \neq i$ cannot be equal to 0, since otherwise, both products would be equal to 0s. Thus, these values are non-zeros. Dividing both sides of the inequality by all these values, we conclude that $e_i^= > e_i^{\neq}$.

Similarly, from the condition that the sequences $d^=$ and $d^{\neq}$ properly describe reasonableness, it also follows, in particular, that for each i, we have $d(\varepsilon_{\neq}) < d(\varepsilon_{\neq}^{(i)})$, where

$$\varepsilon_{\neq}^{(i)} = \stackrel{\text{def}}{=} (\neq, \ldots, \neq, = \text{ (on } i\text{-th place)}, \neq, \ldots, \neq).$$

This inequality is equivalent to $e(\varepsilon_{\neq}) > e(\varepsilon_{\neq}^{(i)})$.

Since the values $e(\varepsilon)$ are simply the products, we thus conclude that

$$\prod_{j=1}^{N} e_j^{\neq} < \left(\prod_{j\neq i} e_j^{\neq}\right) \cdot e_i^=.$$

The values $e_j^{\neq}$ corresponding to $j \neq i$ cannot be equal to 0, since otherwise, both products would be equal to 0s.

Thus, for all i, we have $e_i^= > e_i^{\neq} > 0$.

Now, in general, maximizing the product $e(\varepsilon) = \prod\limits_{i=1}^{N} e_i^{\varepsilon_i}$ is equivalent to maximizing the same product divided by a constant $c \stackrel{\text{def}}{=} \prod\limits_{i=1}^{N} e_i^=$. The ratio $\dfrac{e(\varepsilon)}{c}$ can be equivalently reformulated as $\dfrac{e(\varepsilon)}{c} = \prod\limits_{i:\varepsilon_i=\neq} \dfrac{e_i^{\neq}}{e_i^=}$.

Since logarithm is a strictly increasing function, maximizing this product is, in its turn, equivalent to minimizing its negative logarithm, i.e., the value

$$L(\varepsilon) \overset{\text{def}}{=} -\ln\left(\frac{e(\varepsilon)}{c}\right) = \sum_{i:\varepsilon_i=\neq} w_i,$$

where we denoted $w_i \overset{\text{def}}{=} -\ln\left(\frac{e_i^{\neq}}{e_i^{=}}\right)$. Since $e_i^{=} > e_i^{\neq} > 0$, we have $\frac{e_i^{\neq}}{e_i^{=}} < 1$ and thus, $w_i > 0$. The proposition is proven.

4.5 A Similar Derivation Can Be Obtained in the Probabilistic Case

In the probabilistic approach, reasonableness can be described by assigning a prior probability $p(\varepsilon)$ to each possible sequences ε. In this case, out of each set of sequences, we should select the most probable one, i.e., the one with the largest value of the prior probability.

Let $p_i^{=}$ be the prior probability that $a_i = 0$, and let $p_i^{\neq} = 1 - p_i^{=}$ be the probability that $a_i \neq 0$. A priori we do not know the relation between the values ε_i and ε_j corresponding to different coefficients $i \neq j$, so it makes sense to assume that the corresponding random variables ε_i and ε_j are independent.

This assumption is in perfect agreement with the maximum entropy idea (also known as the Laplace's indeterminacy principle), according to which, out of all probability distributions which are consistent with our observations, we should select the one for which the entropy $-\sum p_i \cdot \ln(p_i)$ is the largest possible; see, e.g., [22]. Indeed, if we only know marginal distributions, then the maximum entropy idea implies that, according to the joint distribution, all the random variables are independent.

Under this assumption, $p(\varepsilon) = \prod\limits_{i=1}^{N} p_i^{\varepsilon_i}$. Thus, we arrive at the following definition.

Definition 2.

- *Let* $p^=(p_1^=,\ldots,p_N^=)$ *be a sequence of real numbers from the interval* $[0,1]$, *and let* $p_i^{\neq} \overset{\text{def}}{=} 1-p_i^=$. *For each sequence* ε, *we define its* prior probability *as*

$$p(\varepsilon) \overset{\text{def}}{=} \prod_{i=1}^{N} p_i^{\varepsilon_i}.$$

- *We say that the sequence* $p^=$ properly describes reasonableness *if the following two conditions are satisfied:*
  - *the sequence* $\varepsilon_= \overset{\text{def}}{=} (=,\ldots,=)$ *is more probable than all others, i.e.,* $p(\varepsilon_=) > p(\varepsilon)$ *for all* $\varepsilon \neq \varepsilon_=$, *and*
  - *the sequence* $\varepsilon_{\neq} \overset{\text{def}}{=} (\neq,\ldots,\neq)$ *is less probable than all others, i.e.,* $p(\varepsilon_{\neq}) < p(\varepsilon)$ *for all* $\varepsilon \neq \varepsilon_{\neq}$.
- *For each set* S *of sequences, we say that a sequence* $\varepsilon \in S$ is the most probable *if its prior probability is the largest possible, i.e., if* $p(\varepsilon) = \max\limits_{\varepsilon' \in S} p(\varepsilon')$.

Proposition 2. *Let us assume that the sequence* $p^=$ *properly describes reasonableness. Then, there exist weights* $w_i > 0$ *for which, within each set* S, *a sequence* $\varepsilon \in S$ *is the most probable if and only if for this sequence, the sum* $\sum\limits_{i:\varepsilon_i=\neq} w_i$ *is the smallest possible.*

Discussion. In other words, probabilistic techniques also lead to the sparsity condition.

Proof is similar to the Proof of Proposition 1.

Comments.

- The fact that the probabilistic approach leads to the same conclusion as the fuzzy approach makes us more confident that our justification of sparsity is valid.
- The comparison of the above two derivations shows an important advantage of fuzzy-based approach in situations like this, when we have a large amount of uncertainty:
  - the probability-based result is based on the assumption of independence, while

- the fuzzy-based result can allow different types of dependence – as described by different t-norms.

Chapter 5

Theoretical Justification of ℓ^p-Techniques in Blind Image Deconvolution

In signal and image processing, it is often beneficial to use semi-heuristic ℓ^p-methods, i.e., methods that minimize the sum of the p-th powers of the discrepancies. In this chapter, we show that a fuzzy-based analysis of the corresponding intuitive idea leads exactly to the ℓ^p-methods.

The main results of this chapter appear in [15].

5.1 Formulation of the Problem

5.1.1 In general, signal and image reconstruction are ill-posed problems

While cameras and other image-capturing devices are getting better and better every day, none of them is perfect, there is always some blur. This blur comes from the fact that while we would like to capture the intensity $I(x, y)$ at each spatial location (x, y), the signal captured by a real-life measuring device is influenced not only by the intensity $I(x, y)$ at the desired location (x, y), but also by the intensities $I(x', y')$ at nearby locations (x', y'). As a result, instead of reflecting the intensity $I(x, y)$ at the desired point (x, y), the signal $s(x, y)$ measured by the device is a combination of intensities at the point (x, y) and at the nearby points:

$$s(x, y) = \int w(x, y, x', y') \cdot I(x', y') \, dx' \, dy',$$

for appropriate weights $w(x, y, x', y')$.

When we take a photo of a friend with a modern sophisticated cell phone camera, this blur is barely visible – and does not constitute a serious problem. However, when a spaceship takes a photo of a distant plant, the blur is very visible – and needs to be eliminated. In such situations, we need to reconstruct the original image $I(x, y)$ from the blurred image $s(x, y)$.

From the purely mathematical viewpoint, this reconstruction problem is *ill-posed* in the sense that large changes in $I(x, y)$ can lead to very small changes in $s(x, y)$ – and, as a result, unless we impose additional constraints on the original image $I(x, y)$, we cannot reconstruct the image with any reasonable accuracy. This mathematical feature is easy to explain: blurring averages the image. Instead of the original intensity at the point (x, y), we have an average intensity over all the neighbors (x', y') of the original point (x, y). It is known that such averaging eliminates high-frequency components. Thus, if instead of the original signal $I(x, y)$, we consider a different signal

$$I^*(x, y) = I(x, y) + c \cdot \sin(\omega_x \cdot x + \omega_y \cdot y),$$

with a high-frequency component added, the resulting signal $s^*(x, y)$ will be practically the same $s^*(x, y) \approx s(x, y)$ – but the original image, for large c, may be very different.

To be able to reconstruct the image reasonably uniquely, we cannot allow all possible dependencies $I(x, y)$, we need to impose some additional conditions on the original image. This imposition of additional conditions that helps reconstruct the original image is known as *regularization*; see, e.g., [37].

Similarly, the problem of reconstructing a 1-D signal $x(t)$ from observations is ill-posed.

5.1.2 Tikhonov's regularization: a brief reminder

If a signal or an image is smooth (differentiable), then a natural idea is to require that the corresponding derivatives are, on average, small, i.e., e.g., that the mean square value of the derivative does not exceed a certain constant C.

Let us describe this requirement in precise terms. If we have n values $d_1, \ldots, d_n$, then the mean square value is

$$d \stackrel{\text{def}}{=} \sqrt{\frac{d_1^2 + \ldots + d_n^2}{n}}.$$

The requirement that this mean square value is bounded by C, i.e., that

$$\sqrt{\frac{d_1^2 + \ldots + d_n^2}{n}} \le C$$

is equivalent to

$$\frac{d_1^2 + \ldots + d_n^2}{n} \le C^2$$

and, thus, to

$$d_1^2 + \ldots + d_n^2 \le c,$$

where $c \stackrel{\text{def}}{=} n \cdot C^2$.

When we go from the discrete data to the continuous signal or image, then the sum turns into an integral. So, for 1-D signals, we have a constraint

$$\int (\dot{x}(t))^2 \, dt \le c,$$

and for 2-D images $I(x, y)$, we have a similar constraint

$$\int \left(\left(\frac{\partial I}{\partial x} \right)^2 + \left(\frac{\partial I}{\partial y} \right)^2 \right) dx \, dy \le c.$$

Out of all signals or images that satisfy this constraint, we want to find a one which is the best fit with the observation, i.e., for which, e.g., the mean square error is the smallest:

$$J \stackrel{\text{def}}{=} \sum_i e_i^2 \to \min,$$

where e_i is the difference between the value measured in the i-th measurement and the value predicted based on the corresponding signal or image. Thus, we need to minimize J under the constraint

$$\int (\dot{x}(t))^2\,dt \le c$$

or

$$\int \left(\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2\right)\,dx\,dy \le c.$$

In general, constraint optimization problems can be solved by using Lagrange multiplier method, which reduced the above constraint optimization problems to the following unconstrained ones:

$$J + \lambda \cdot \int (\dot{x}(t))^2\,dt \to \min_{x(t)}$$

or

$$J + \lambda \cdot \int \left(\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2\right)\,dx\,dy \to \min_{I(x,y)}.$$

This idea is known as *Tikhonov regularization*.

5.1.3 From continuous to discrete signals and images

In practice, we can only observe a signal with a certain temporal resolution. As a result, in effect, we can only reconstruct the values $x_i = x(t_i)$ of the signal $x(t)$ at points $t_i = t_0 + i \cdot \Delta t$ from an appropriate grid.

Similarly, we only observe an image with a certain spatial resolution, so we can only reconstruct the values

$$I_{ij} = I(x_i, y_j)$$

on a certain grid $x_i = x_0 + i \cdot \Delta x$ and $y_j = y_0 + j \cdot \Delta y$.

In this discrete case, instead of the derivatives, we have differences:

$$J + \lambda \cdot \sum_i (\Delta x_i)^2 \to \min_{x_i}$$

or

$$J + \lambda \cdot \sum_i \sum_j ((\Delta_x I_{ij})^2 + (\Delta_y I_{ij})^2) \to \min_{I_{ij}},$$

where:

- $\Delta x_i \overset{\text{def}}{=} x_i - x_{i-1}$,
- $\Delta_x I_{ij} \overset{\text{def}}{=} I_{ij} - I_{i-1,j}$, and
- $\Delta_y I_{ij} \overset{\text{def}}{=} I_{ij} - I_{i,j-1}$.

5.1.4 Limitations of Tikhonov regularization

Tikhonov regularization is based on the assumption that the signal or the image is smooth. In real life, signals and images are, in general, not smooth; for example, many of them exhibit a fractal behavior; see, e.g., [28]. In such non-smooth situations, Tikhonov regularization does not work so well.

5.1.5 ℓ^p-methods as a heuristic idea to take non-smoothness into account

To take into account non-smoothness, researchers have proposed to modify the Tikhonov regularization formulas by using, instead of the squares of the derivatives, the p-th powers corresponding to some $p \neq 2$. In the resulting ℓ^p-approach, we solve the following minimization problems (see, e.g., [3, 25, 34]):

$$J + \lambda \cdot \sum_i |\Delta x_i|^p \to \min_{x_i}$$

or

$$J + \lambda \cdot \sum_i \sum_j (|\Delta_x I_{ij}|^p + |\Delta_y I_{ij}|^p) \to \min_{I_{ij}}.$$

These methods work much better than the original Tikhonov regularization [3, 25, 34].

5.1.6 Remaining problem

The remaining problem is that the ℓ^p-methods are heuristic, there is no convincing explanation of why necessarily we replace the square with a p-th power and not, for example, with some other function.

5.1.7 What we do in this chapter

In this chapter, we show that a natural formalization of the corresponding intuitive ideas indeed leads to ℓ^p-methods.

To formalize the intuitive ideas behind signal and image reconstruct, we use *fuzzy techniques*, a known way to transform imprecise intuitive ideas into exact formulas.

5.2 Let us Apply Fuzzy Techniques to Our Problem

5.2.1 Need to describe imprecise ("fuzzy") expert knowledge

In many areas of science and engineering, a large portion of expert knowledge is formulated by using imprecise words from natural language, such as "small", "fast", etc. Fuzzy logic is a technique designed to translate such knowledge into precise computer-understandable form; see, e.g., [24, 33, 43].

The main idea behind fuzzy logic is that to describe an imprecise notion like "small", we assign, to each possible value x of the corresponding quantity, a degree $\mu(x)$ from the interval $[0, 1]$ to which this quantity satisfies the given property (e.g., to what extent x is small):

- the value $\mu(x) = 1$ means that the expert is absolutely sure that x is small;
- the value $\mu(x) = 0$ means that the expert is absolutely sure than x is not small, and
- values $\mu(x)$ strictly between 0 and 1 indicate that the expert is not fully confident that x is small.

Each value $\mu(x)$ can be obtained, e.g., by asking an expert to indicate, on a scale from 0 to 1, to what extend the value x is small.

5.2.2 "And"- and "or"-operations: a brief reminder

A large part of expert knowledge is formulated in terms of if-then rules. For example, we can have a rule like: "if the temperature is high and the humidity is low, then there is a high chance that the fertilizer may self-ignite".

To adequately translate these rules into precise terms, we need to know the degree to which, for given temperature and given humidity, the condition of this rule is satisfied, i.e., to which extent it is true that the temperature is high *and* the humidity is low.

Ideally, we show be able to elicit these degrees from the expert, by asking the expert to what extent this condition is true for all possible combinations of temperature and humidity. However, this is often not practically possible, since there are many possible such combinations, and it is even less practical if the condition of a rule combines three or more statements.

To deal with such situations, it is necessary to be able, given the expert's degrees of belief a and b in statements A and B, to estimate the expert's degree of belief in a composite statement $A \,\&\, B$. The corresponding estimation algorithm $f_\&(a,b)$ is known as an *"and"-operation*, or *t-norm.*

The fact that this operation corresponds to "and" implies some of its natural properties. For example, since $A \,\&\, B$ means that same as $B \,\&\, A$, this operation should be commutative:

$$f_\&(a,b) = f_\&(b,a).$$

The fact that $A \,\&\, (B \,\&\, C)$ means the same as $(A \,\&\, B) \,\&\, C$ implies that the "and"-operation should be associative, i.e.,

$$f_\&(a, f_\&(b,c)) = f_\&(f_\&(a,b),c),$$

etc.

Similarly, to describe the expert's degree of belief in statements of the type $A \vee B$ ("A or B") based on his/her degrees of confidence a and b in individual statements A and B, we need to use "or"-operations (t-conorms) $f_\vee(a, b)$ which are also commutative and associative.

5.2.3 What we are trying to formalize

We are trying to formalize the statement that the signal or image is continuous, i.e., for signal, that all the differences $\Delta x_i = x_i - x_{i-1}$ are small. In other words, we want to say that the difference Δx_1 is small *and* that the difference Δx_2 is small *and* that the difference Δx_3 is small, etc.

Similarly, we want to say that the differences $\Delta_x I_{ij}$ and $\Delta_y I_{ij}$ between image intensities at nearby points are small.

Among all the signals (images) which are consistent with the observations, i.e., for which $J \le c_0$ for some c_0, it is then reasonable to select a signal for which the degree d with which the above "and"-statement is satisfied is the largest possible.

5.2.4 What we need to do to formalize this statement

According to the above-mentioned fuzzy logic techniques, to formalize this statement,

- we need to describe what is small, and
- we also need to select an appropriate "and"-operation.

Let $\mu(x)$ describe the degree to which x is small. Then the degree d to which the whole "and"-statement is satisfied is equal to

$$d = f_\&(\mu(\Delta x_1), \mu(\Delta x_2), \mu(\Delta x_3), \ldots).$$

5.2.5 Selecting an "and"-operation

It is known (see, e.g., [32]) that each "and"-operation can be approximated, for any given accuracy $\varepsilon > 0$, by an *Archimedean* "and"-operation, i.e., by an "and"-operation of the type $f_{\&}(a, b) = f^{-1}(f(a)) \cdot f(b))$ for some increasing function $f(a)$ from $[0, 1]$ to $[0, 1]$, where f^{-1} denotes the inverse function.

Thus, without losing generality, we can safely assume that the actual "and"-operation has exactly this type.

5.2.6 The selection of an "and"-operation simplifies the corresponding optimization problem

For the above Archimedean "and"-operation, the above expression for d has the form

$$d = f^{-1}(f(\mu(\Delta x_1)) \cdot f(\mu(\Delta x_2)) \cdot f(\mu(\Delta x_3)) \cdot \ldots).$$

Since the function $f(x)$ is increasing, maximizing d is equivalent to maximizing the value

$$f(d) = f(\mu(\Delta x_1)) \cdot f(\mu(\Delta x_2)) \cdot f(\mu(\Delta x_3)) \cdot \ldots$$

Maximizing this product is equivalent to minimizing its negative logarithm

$$L \stackrel{\text{def}}{=} -\ln(d).$$

Since the logarithm of the product is equal to the sum of the logarithms, we get

$$L = -\sum_i \ln(f(\mu(\Delta x_i))),$$

i.e.,

$$L = \sum_i g(\Delta x_i),$$

where we denoted $g(x) \stackrel{\text{def}}{=} -\ln(f(\mu(x)))$.

In these terms, selecting a membership function is equivalent to selecting the related function $g(x)$.

5.2.7 Which function $g(x)$ should we select: idea

The value $\Delta x_i = 0$ is definitely small, so we should have $\mu(0) = 1$. Here, $f(1) = 1$, so $f(\mu(0)) = 1$ and thus, $g(0) = \ln(1) = 0$.

The numerical value of a difference Δx_i depends on the choice of a measuring unit. If we choose a measuring unit which is a times smaller than the original one, then instead of the original numerical value Δx_i, we will have a different numerical value $a \cdot \Delta x_i$.

It is reasonable to require that the requirement

$$\sum_i g(\Delta x_i) \to \min$$

should not change if we simply change a measuring unit. For example, if for two pairs (z_1, z_2) and (z'_1, z'_2), we have the same value of the sum $\sum\limits_i g(z_i)$, then this equality should remain true for all values $a > 0$. In precise terms, if we have

$$g(z_1) + g(z_2) = g(z'_1) + g(z'_2),$$

then we should have

$$g(a \cdot z_1) + g(a \cdot z_2) = g(a \cdot z'_1) + g(a \cdot z'_2).$$

5.2.8 Let us find the corresponding function $g(x)$

Let us consider the case when:

- z'_1 is close to z_1, i.e., when $z'_1 = z_1 + \Delta z$ for a small value Δz, and
- z'_2 is close to z_2, i.e.,

  $$z'_2 = z_2 + k \cdot \Delta z + o(\Delta z)$$

  for an appropriate k.

Substituting these values z'_1 and z'_2 into the above equality, we get

$$g(z_1) + g(z_2) = g(z_1 + \Delta z) + g(z_2 + k \cdot \Delta z).$$

Here,

$$g(z_1 + \Delta z) = g(z_1) + g'(z_1) \cdot \Delta z + o(\Delta z)$$

and

$$g(z_2 + k \cdot \Delta z) = g(z_2) + g'(z_2) \cdot k \cdot \Delta z + o(\Delta z),$$

so the above equality implies that

$$g'(z_1) \cdot \Delta z + g'(z_2) \cdot k \cdot \Delta z + o(\Delta z) = 0.$$

Diving both sides by Δz and taking $\Delta z \to 0$, we get

$$g'(z_1) + g'(z_2) \cdot k = 0,$$

hence

$$k = -\frac{g'(z_1)}{g'(z_2)}.$$

The condition

$$g(a \cdot z_1) + g(a \cdot z_2) = g(a \cdot z_1') + g(a \cdot z_2')$$

similarly takes the form

$$g'(a \cdot z_1) + g'(a \cdot z_2) \cdot k = 0,$$

i.e.,

$$g'(a \cdot z_1) - g'(a \cdot z_2) \cdot \frac{g'(z_1)}{g'(z_2)} = 0.$$

Thus,

$$g'(a \cdot z_1) = g'(a \cdot z_2) \cdot \frac{g'(z_1)}{g'(z_2)}.$$

By moving all the terms related to z_1 to the left-hand side and all other terms to the right-hand side, we get

$$\frac{g'(a \cdot z_1)}{g'(z_1)} = \frac{g'(a \cdot z_2)}{g'(z_2)}$$

for all a, z_1, and z_2.

This means that the ratio $\dfrac{g'(a \cdot z_1)}{g'(z_1)}$ does not depend on z_i, it only depends on a:

$$\frac{g'(a \cdot z_1)}{g'(z_1)} = F(a)$$

for some function $F(a)$.

For $a = a_1 \cdot a_2$, we have

$$F(a) = \frac{g'(a \cdot z_1)}{g'(z_1)} = \frac{g'(a_1 \cdot a_2 \cdot z_1)}{g'(z_1)} =$$

$$\frac{g'(a_1 \cdot (a_2 \cdot z_1))}{g'(a_2 \cdot z_1)} \cdot \frac{g'(a_2 \cdot z_1)}{g'(z_1)} = F(a_1) \cdot F(a_2),$$

i.e.,

$$F(a_1 \cdot a_2) = F(a_1) \cdot F(a_2).$$

It is known (see, e.g., [1]) that every continuous function satisfying this property has the form $F(a) = a^q$ for some real number q.

The condition $\dfrac{g'(a \cdot z_1)}{g'(z_1)} = F(a)$ now takes the form

$$g'(a \cdot z_1) = g'(z_1) \cdot F(a) = g'(z_1) \cdot a^p.$$

In particular, for $z_1 = 1$, we get

$$g'(a) = C \cdot a^q,$$

where $C \overset{\text{def}}{=} g'(1)$.

We have an expression for the derivative $g'(a)$ of the desired function $g(a)$. To get $g(a)$, we therefore need to integrate this derivative. For this integration, we have two different formulas: for $q = -1$ and for all other q.

Let us show that the value $q = -1$ is impossible. Indeed, if $q = -1$, we get $g(a) = C \cdot \ln(a) + \text{const}$, which contradicts to the above requirement that $g(0) = 0$.

Thus, we have $q \neq -1$. Therefore, integration leads to

$$g(a) = \frac{C}{q+1} \cdot a^{q+1} + \text{const}.$$

5.2.9 Conclusion: we have indeed justified the ℓ^p-method

For the above function $g(x)$, we have

$$\sum_i g(\Delta x_i) = \frac{C}{q+1} \cdot \sum_i |\Delta x_i|^{q+1} + \text{const}.$$

Minimizing this sum is equivalent to minimizing the sum

$$\sum_i |\Delta x_i|^{q+1}.$$

According to the Lagrange multiplier method, minimizing this sum under the constraint $J \le c$ is equivalent to minimizing the expression

$$J + \lambda \cdot \sum_i |\Delta x_i|^p,$$

for $p = q + 1$. Thus, for signals, we have indeed justified the ℓ^p-method.

Similar arguments explain the ℓ^p-method for images.

Chapter 6

The Idea of Rotation-Invariance Enables Us To Further Improve the Results of the State-of-the-Art Blind Deconvolution Techniques

6.1 Need for Improvement: Reminder

The current state-of-the-art method for blind image deconvolution is based on minimizing the sum

$$|\Delta_x I_{i,j}|^p + |\Delta_y I_{i,j}|^p \tag{6.1.1}$$

for some $p < 2$, where $\Delta_x I_{i,j} \stackrel{\text{def}}{=} I_{i,j} - I_{i-1,j}$, and $\Delta_y I_{ij} \stackrel{\text{def}}{=} I_{i,j} - I_{i,j-1}$. This is a discrete analog of the term

$$\left|\frac{\partial I}{\partial x}\right|^p + \left|\frac{\partial I}{\partial y}\right|^p. \tag{6.1.2}$$

In the traditional least squares approach, when $p = 2$, the corresponding expression

$$\left|\frac{\partial I}{\partial x}\right|^2 + \left|\frac{\partial I}{\partial y}\right|^2 \tag{6.1.3}$$

is rotation-invariant: namely, it describes the square of the length of the gradient vector

$$\nabla I \stackrel{\text{def}}{=} \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right). \tag{6.1.4}$$

However, for $p \neq 2$, the corresponding expression is no longer rotation-invariant.

Since the current state-of-the-art technique is not rotation-invariant, the result of using his techniques may change with rotation. Thus, if we rotate the image a little bit, the method, in general, leads to a different deconvolution result. So, even when the original reconstruction is optimal, the reconstruction of a rotated image will be different and, thus, not optimal.

6.2 How to Improve: Main Idea

As we have just mentioned, the main problem with the current state-of-the-art blind deconvolution techniques comes from the fact that these techniques are not rotation-invariant. To improve the quality of image decomposition, it is therefore desirable to modify the current state-of-the art techniques by making them rotation-invariant.

In other words, instead of the above non-rotational-invariant expression, we need a rotation-invariant one. Let us first consider the continuous approximation. In this approximation, the desired expression depends on the components $\dfrac{\partial I}{\partial x}$ and $\dfrac{\partial I}{\partial y}$ of the gradient vector (6.1.4). When we rotate the coordinate system, the components of the gradient vector change.

In general, a 2-D vector can be characterized by its length and its direction. When we rotate the coordinate system, the direction changes but the length remains unchanged. Thus, the only rotation-invariant characteristic of a vector $\vec{a}$ is its length $\|\vec{a}\|$. Thus, since we want the desired expression to be rotation-invariant, it must depend only on the length $\|\nabla I\|$ of the gradient vector, i.e., only on the expression

$$\|\nabla I\| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}. \tag{6.2.1}$$

The actual images are discrete. Thus, as we have mentioned earlier, instead of the derivatives, we have finite differences $\Delta_x I_{i,j}$ and $\Delta y I_{i,j}$, and instead of $\|\nabla I\|$, we have an expression

$$\sqrt{(\Delta_x I_{i,j})^2 + (\Delta_y I_{i,j})^2}. \tag{6.2.2}$$

So, the desired rotation-invariant expression E must have the form

$$E = f(\sqrt{(\Delta_x I_{i,j})^2 + (\Delta_y I_{i,j})^2}) \tag{6.2.3}$$

for an appropriate function $f(x)$.

To find the function $f(x)$, let us consider a degenerate case, in which the image, in effect, is 1-dimensional, i.e., when the intensity does not change in the y-direction, it only varies in the x-direction: $I(i, j) = I(i)$. In this degenerate case, $\Delta_x I_{i,j} = \Delta_x I_i$ and $\Delta_y I_{i,j} = 0$. Thus, the above expression (6.2.3) takes the form

$$E = f(\sqrt{(\Delta_x I_i)^2}) = f(|\Delta_x I_i|). \tag{6.2.4}$$

On the other hand, We have already discussed, in the previous section, that in the 1-D case, the corresponding expression should be proportional to $|\Delta_x I_i|^p$ for some p, i.e., it should take the form

$$E = c \cdot |\Delta_x I_i|^p \tag{6.2.5}$$

for some c and p. By comparing the formulas (6.2.4) and (6.2.5), we conclude that

$$f(x) = c \cdot |x|^p.$$

Substituting this expression for $f(x)$ into the formula (6.2.3) that describes the general 2-D case, we thus conclude that

$$E = c \cdot \left|\sqrt{(\Delta_x I_{i,j})^2 + (\Delta_y I_{i,j})^2}\right|^p = c \cdot ((\Delta_x I_{i,j})^2 + (\Delta y I_{i,j})^2)^{p/2}. \tag{6.2.6}$$

Thus, we arrive at the following conclusion: to make the blind deconvolution method rotation-invariant, we need to replace the non-rotation-invariant expression (6.1.1) with a rotation-invariant expression

$$E = c \cdot ((\Delta_x I_{i,j})^2 + (\Delta_y I_{i,j})^2)^{p/2}, \tag{6.2.7}$$

for an appropriate constant c.

6.3 From the Idea to the Algorithmic Details

How does the above change in the objective function affect the resulting blind deconvolution algorithm? In terms of the algorithm, we replace the sum

$$|\Delta_i^h(x)|^p + |\Delta_i^v(x)|^p \tag{6.3.1}$$

with the new expression proportional to

$$((\Delta_i^h(x))^2 + (\Delta_i^v(x))^2)^{p/2}. \tag{6.3.2}$$

According to the description of the state-of-the-art algorithm in Chapter 2, to minimize the expression (6.3.1), we represent it as

$$(\Delta_i^h(x))^2 \cdot v_{h,i}^{p/2-1} + (\Delta_i^v(x))^2 \cdot v_{v,i}^{p/2-1}, \tag{6.3.3}$$

where:

- $v_{h,i}$ is the value of $(\Delta_i^h(x))^2$ on the previous iteration, and
- $v_{v,i}$ is the value of $(\Delta_i^v(x))^2$ on the previous iteration.

We can apply the same idea to minimize the new expression (6.3.2). Specifically, to minimize this expression (6.3.2), we represent it as

$$((\Delta_i^h(x))^2 + (\Delta_i^v(x))^2) \cdot v^{p/2-1}, \tag{6.3.4}$$

where v is the value of the sum $(\Delta_i^h(x))^2 + (\Delta_i^v(x))^2$ on the previous iteration.

The expression (6.3.4) can be described in the form similar to (6.3.3), as

$$(\Delta_i^h(x))^2 \cdot (v'_{h,i})^{p/2-1} + (\Delta_i^v(x))^2 \cdot (v'_{v,i})^{p/2-1}, \tag{6.3.5}$$

where $v'_{h,i} = v'_{v,i}$ is proportional to the value of the sum $(\Delta_i^h(x))^2 + (\Delta_i^v(x))^2$ on the previous iteration. We have already denoted the values of the squares of differences $(\Delta_i^h(x))^2$ and $(\Delta_i^v(x))^2$ on the previous iteration by $v_{h,i}$ and $v_{v,i}$. Thus, we have

$$v'_{h,i} = v'_{v,i} = C \cdot (v_{h,i} + v_{v,i}). \tag{6.3.6}$$

In other words, at each spatial location i, instead of possible different values $v_{h,i} \neq v_{v,i}$, we apply equal weights $v'_{h,i} = v'_{v,i}$ to horizontal and vertical differences.

The current method has been tuned to work well. So, it makes sense to make the difference between the current method and its proposed modification to be as small as possible. When $v_{h,i} \neq v_{v,i}$, our new method differs from the current one, but when $v_{h,i} = v_{v,i}$, there is no reason for it to differ. It is therefore reasonable to select a constant C in such a way that when $v_{h,i} = v_{v,i}$, the new method will lead to exactly the same result as the current one. In other words, when $v_{h,i} = v_{v,i}$, we should have $v'_{h,i} = v_{h,i}$ and $v'_{v,i} = v_{v,i}$. Substituting these values into the formula (6.3.6), we conclude that $C = \frac{1}{2}$. Thus, the formula (6.3.6) takes the following final form:

$$v'_{h,i} = v'_{v,i} = \frac{1}{2} \cdot (v_{h,i} + v_{v,i}). \tag{6.3.7}$$

So, we arrive at the following modification of the state-of-the-art blind deconvolution algorithm.

6.4 Resulting Modification of the State-of-the-Art Blind Deconvolution Algorithm

The only modification is on Step 1(c), where after computing, for each spatial location i, the horizontal and vertical values $v_{h,i}$ and $v_{v,i}$, we then average these two values before performing further computations:

First, we select α, β, γ, τ, θ, η^1, and $0 < p < 1$. We then select some initial estimate $h^{1,1}$ of the blur, and the initial values of auxiliary vectors $v_d^{1,1}$, where $d \in D = \{h, v, hh, vv, hv\}$.

Then, for $k = 1, 2, \ldots$, we perform the following until the stopping criterion

$$\frac{\|x^k - x^{k-1}\|}{\|x^{k-1}\|} < \varepsilon = 0.01$$

is met:

1. For $\ell = 1, \ldots, L_0$ for some L_0:

   (a) Compute

   $$x^{k,l+1} = \left[\eta^k (\mathbf{H}^{k,l})^T (\mathbf{H}^{k,l}) + \alpha p \sum_{d \in D} 2^{1-o(d)} (\mathbf{\Delta}^d)^T \mathbf{B}_d^{k,l} \mathbf{\Delta}^d \right]^{-1} \cdot \eta^k (\mathbf{H}^{k,l})^T \mathbf{W} a^k,$$

   where $B_d^{k,\ell}$ is a diagonal matrix with entries $B_d^{k,\ell}(i,i) = \left(v_{d,i}^{k,\ell}\right)^{p/2-1}$ and $\mathbf{\Delta}^d$ is the convolution matrix of the difference operator $\Delta_i^d(\cdot)$. For solving this system of linear equations, the authors use the Fourier transform approach.

   (b) Compute

   $$h^{k,l+1} = \left[\eta^k (\mathbf{X}^{k,l})^T (\mathbf{X}^{k,l}) + \gamma \mathbf{C}^T \mathbf{C}\right]^{-1} \cdot \eta^k (\mathbf{X}^{k,l})^T \mathbf{W} a^k,$$

   where $\mathbf{X}^{k,\ell}$ is the convolution matrix of the image $x^{k,\ell}$. For solving this system of linear equations, the authors also use the Fourier transform approach.

   (c) For each $d \in \{h, v, hh, vv, hv\}$, calculate

   $$v_{d,i}^{k,\ell+1} = \left[\Delta_i^d (x^{k,\ell})\right]^2;$$

   for $d \in \{h, v\}$, calculate

   $$v_{h,i}^{k,\ell+1} = v_{v,i}^{k,\ell+1} = \frac{1}{2} \cdot \left(\left[\Delta_i^h (x^{k,\ell})\right]^2 + \left[\Delta_i^v (x^{k,\ell})\right]^2\right);$$

2. Set $x^k = x^{k,\ell+1}$, $h^k = h^{k,\ell+1}$, and $v_{d,i}^k = v_{d,i}^{k,\ell+1}$.

3. Now, we need to find a^{k+1} by minimizing the expression

   $$\frac{\beta}{2} \cdot \|y - \mathbf{W}a\|_2^2 + \frac{\eta}{2} \cdot \|\mathbf{W}a - \mathbf{H}x\|_2^2 + \tau \cdot \|a\|_1.$$

   This can be done by applying the *l1-ls* algorithm for minimizing the equivalent minimization problem

   $$\|y' - \mathbf{\Phi}' \mathbf{W} a^k\|^2 + \tau \|a^k\|_1,$$

where

$$y' = \begin{bmatrix} \sqrt{\frac{\beta}{2}} y \\ \\ \sqrt{\frac{\eta^k}{2}} \mathbf{H}^k x^k \end{bmatrix}$$

and

$$\mathbf{\Phi}' = \begin{bmatrix} \sqrt{\frac{\beta}{2}} \mathbf{I} \\ \\ \sqrt{\frac{\eta^k}{2}} \mathbf{I} \end{bmatrix}.$$

4. Set $\eta^{k+1} = \theta \eta^k$.

6.5 Testing the New Algorithm

To test the new method, we compared it with the original methods on the same "Cameraman" image on which the authors of the original paper [3] tested their method. In our application, we used the same values of the parameters that the authors of [3] used:

$$\{\alpha, \beta, \gamma, \tau, \eta^1\} = \{1, 1/\sigma^2, 5e5, 0.125, 1042\},$$

where σ^2 denotes the noise variance.

Following [3], we also applied, to the original image, the Gaussian blurring with the variance of 5.

We selected $\sigma = 0.001$, which is consistent with the signal-to-noise ratio used in [3]. Following [3], we used the mean square difference $\|x - \widehat{x}\|_2$ between the original image x and the reconstructed image $\widehat{x}$ to gauge the quality of deconvolution.

When comparing the results of the two algorithms, we need to take into account that both the original and the modified algorithms start with randomly selected initial values $v_d^{1,1}$. Because of this, the results of both algorithm may differ slightly when we re-apply the same algorithm to the same blurred image. Because of the statistical character of the results, to compare the two algorithms, we need to apply both algorithms to the same blurred image several times, and then use statistical criteria to decide which method is better.

To perform this comparison, we applied each of the two algorithms 30 times, and for each application, we computed the distance $\|x - \widehat{x}\|_2$. To make the results of the comparison more robust, for each of the algorithms, we eliminated the smallest and the largest value of this distance, and got a list of 28 values. For the original algorithm, we get the following results:

1192.44, 1192.97, 1202.01, 1196.93, 1191.03, 1195.04, 1195.28,
1204.42, 1194.01, 1192.15, 1195.05, 1191.27, 1190.42, 1192.78,
1192.20, 1196.84, 1202.12, 1194.88, 1192.15, 1195.05, 1189.90,
1189.36, 1191.27, 1190.42, 1192.78, 1192.20, 1196.84, 1202.12.

The average of these values is 1195.21.

For the modified method, we get the following 28 values:

1195.78, 1190.74, 1188.81, 1188.60, 1190.43, 1189.07, 1191.83
1187.20, 1189.36, 1191.77, 1189.05, 1189.24, 1189.36, 1198.58
1193.91, 1188.08, 1192.04, 1191.54, 1189.36, 1191.77, 1189.05
1189.24, 1189.36, 1198.58, 1193.91, 1188.08, 1192.03, 1191.54.

The average of these values is 1191.01, which is smaller than the average distance corresponding to the original algorithm.

To check whether this difference is statistically significance, we applied the t-test for two independent means. In this test, given two samples of sizes N_1 and N_2, we compute the corresponding sample means $\overline{X}_1$ and $\overline{X}_2$, sample variances s_1^2 and s_2^2, and the value

$$t = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\left(\frac{(N_1 - 1)\cdot s_1^2 + (N_2 - 1)\cdot s_2^2}{N_1 + N_2 - 2}\right)\cdot\left(\frac{1}{N_1} + \frac{1}{N_2}\right)}}.$$

Based on the value of this statistic, we decide whether the null hypothesis – that both samples comes from the populations with same mean – can be rejected.

For the two above samples, computations lead to rejection with $p = 0.002$. This is much smaller than the p-values 0.01 and 0.05 normally used for rejecting the null hypothesis. So, we can conclude that the null hypothesis can be rejected, and that, therefore, the *modified algorithm is statistically significantly better than the original one.*

Chapter 7

Towards Using Embedded Zerotree Wavelet (EZW) Algorithm

7.1 Theoretical Background of Embedded Zerotree Wavelet (EZW) Coding

Main idea behind EZW. As we have mentioned earlier, one of the important features enabling us to perform blind deconvolution is the image sparsity. Specifically, in the wavelet transform of an image, most coefficients are so small that they can be safely ignored. Thus, when we reconstruct the actual image, it is sufficient to only consider images in which the number of non-zero wavelet coefficients is small.

In the previous chapters, this is how – following [3] – we interpreted the sparsity requirement. However, in image processing in general, there are more complex methods of formalizing sparsity, methods that lead to more efficient image compression and image compressing. The main idea behind such methods is motivated by the fact that each coefficient of a wavelet transform is a linear combination the intensities in some part of an image. Each of the few low-frequency coefficients is a linear combination over a reasonably large portion of an image. As we move to coefficients representing higher and higher frequencies, the size of the corresponding part of the image decreases.

If a wavelet coefficient corresponding to some part of the image is 0, this probably means that this particular part of the image is not significant and can be ignored – and thus, wavelet coefficients corresponding to sub-parts of this part can be safely ignored as well.

Different wavelet coefficients can be naturally described as tree: for each coefficient describing some part of the image, its "children" are the coefficients describing subparts of this part. The above idea is that if a "parent" node is 0, then its children nodes – and children of these children notes, etc. should also be all zeros. Such a tree is called a *zerotree.* If we use a zerotree approach, then the whole subtree with all 0s is "embedded" into the representation of the original image. Because of this, methods that use such zerotrees are known as *Embedded Zerotree Wavelet* algorithms (EZW for short). The main ideas behind EZW were first described in [36]; see also [40].

What we do in this chapter. In this chapter, we will analyze the possibility to incorporate EZW techniques into blind deconvolution. We have proposed and tests two ways of such incorporation. So far, we have not yet got any statistically significant improvement of blind deconvolution, but we hope that our ideas will be further enhanced and lead to such an improvement.

To explain our ideas, we first need to explain the EZW techniques in more detail.

1-D wavelet transform of signals: the basis for 2-D wavelet transform of images. The usual 2-D wavelet transform of images is obtained by applying the 1-D wavelet transform to each row, and then applying the same transform to each column. Thus, to better understand the 2-D wavelet transform algorithm, let us first describe how the 1-D wavelet transform is usually performed.

Based on the 1-D signal $x_1, \ldots, x_n$, we first compute:

- its *low-pass* (low-frequency) component L_1 consisting of the values $x_1 + x_2$, $x_3 + x_4$, etc., and
- its *high-pass* (high-frequency) component H_1 consisting of the values $x_1 - x_2$, $x_3 - x_4$, etc.

The values H_1 remain unchanged, but to the low-frequency values L_1, we apply a similar transformation, replacing L_1 with two parts: low-pass part L_2 and high-pass part H_2. To the low-pass part L_2, we apply the same transformation, etc.

For example, if we have a signal

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8$$

consisting of eight values $x_1, \ldots, x_8$, then in the first pass, we transform this signal into the following sequence:

$$x_1 + x_2, x_3 + x_4, x_5 + x_6, x_7 + x_8, x_1 - x_2, x_3 - x_4, x_5 - x_6, x_7 - x_8.$$

On the second pass, we apply a similar transformation to the part

$$L_1 = \{x_1 + x_2, x_3 + x_4, x_5 + x_6, x_7 + x_8\},$$

transforming it into a new sequence

$$(x_1 + x_2) + (x_3 + x_4), (x_5 + x_6) + (x_7 + x_8), (x_1 + x_2) - (x_3 + x_4), (x_5 + x_6) - (x_7 + x_8).$$

On the third pass, we take a low-pass part of this new sequence

$$L_2 = \{(x_1 + x_2) + (x_3 + x_4), (x_5 + x_6) + (x_7 + x_8)\},$$

and apply to it a similar transformation. As a result, we get a new 2-number sequence

$$((x_1+x_2)+(x_3+x_4))+((x_5+x_6)+(x_7+x_8)), ((x_1+x_2)+(x_3+x_4))-((x_5+x_6)+(x_7+x_8)).$$

The new low-pass part L_3 consists of only one value, so the process stops. The resulting wavelet transform has the form

$$L_3, H_3, H_2, H_1,$$

where

$$L_3 = \{((x_1 + x_2) + (x_3 + x_4)) + ((x_5 + x_6) + (x_7 + x_8))\},$$

$$H_3 = \{((x_1 + x_2) + (x_3 + x_4)) - ((x_5 + x_6) + (x_7 + x_8))\},$$

$$H_2 = \{(x_1 + x_2) - (x_3 + x_4), (x_5 + x_6) - (x_7 + x_8)\},$$

and

$$H_1 = \{x_1 - x_2, x_3 - x_4, x_5 - x_6, x_7 - x_8\}.$$

In general, the 1-D wavelet transform can be schematically described as follows:

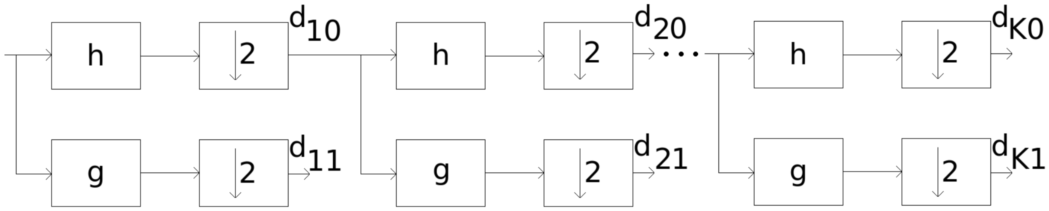

Figure 7.1: K-th Level Decomposition, 1-D Wavelet Decomposition

From 1-D to 2-D transforms. First, we apply one step of the 1-D wavelet transform to each row; as a result, instead of the original values, we get half low-pass values L and half high-pass values H. Then, to the resulting values, we apply the one-step transform to each column. As a result, we have four sets of values:

- values LL_1 that correspond to the low-pass filtering by row and by column;
- values HL_1 that correspond to high-pass filtering by row and low-pass filtering by column;
- values LH_1 that correspond to low-pass filtering by rows and high-pass filtering by column, and
- values HH_1 that correspond to high-pass filtering both by row and by column.

The result is described in Fig. 7.2.

After that, the regions HL_1, LH_1, and HH_1 remain unchanged, but to the region LL_1, we apply a similar transformation, dividing this region into four subregions LL_2, HL_2, LH_2, and HH_2; see Fig. 7.3.

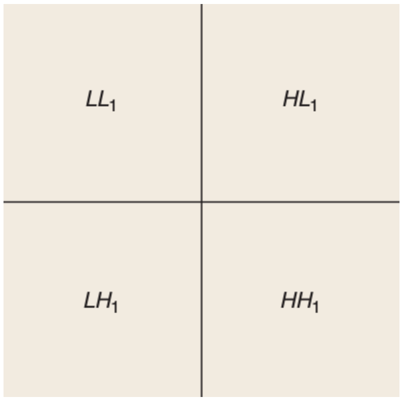

Figure 7.2: Labeling Scheme for a one level, 2-D wavelet transform [40]

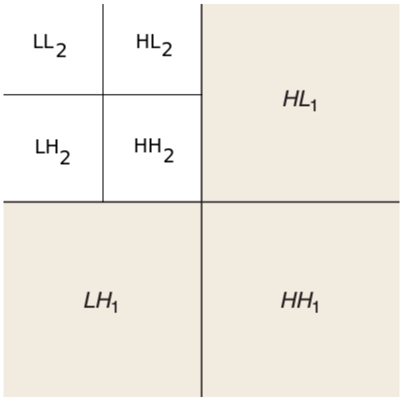

Figure 7.3: Labeling Scheme for a two level, 2-D wavelet transform [40]

If we apply again the 2-D wavelet transform to the LL_2 region in Fig. 7.3, we get the labeling scheme for the third level decomposition of the 2-D wavelet transform, shown in Fig. 7.4.

For the fourth level of decomposition, the 2-D wavelet transform is performed in the LL_3 region. This process continues until the desired level K of decomposition is reached.

Wavelet transforms is an efficient way of representing an image. For most practical signals and images, most of the energy of the original signal or image is concentrated in the low pass region L_K or LL_K. Moreover, most of the energy in the high-frequency bands HL_i, LH_i, and HH_i is concentrated on a relatively small number of coefficients; see, [40]. The probability distribution describing the frequency of different values of wavelet coefficients

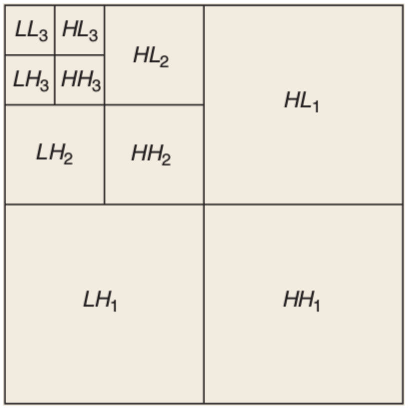

Figure 7.4: Labeling Scheme for a three level, 2-D wavelet transform [40]

has a Laplacian-like behavior, i.e.

$$f(x) = A \cdot \exp\left(-\left(\frac{|x|}{\sigma^2}\right)^{\beta}\right),$$

for some values σ^2 and β. Here, σ^2 is close to the variance, and β represents the rolloff of the distribution [40]. The 1-D Laplacian function is shown in Fig. 7.5.

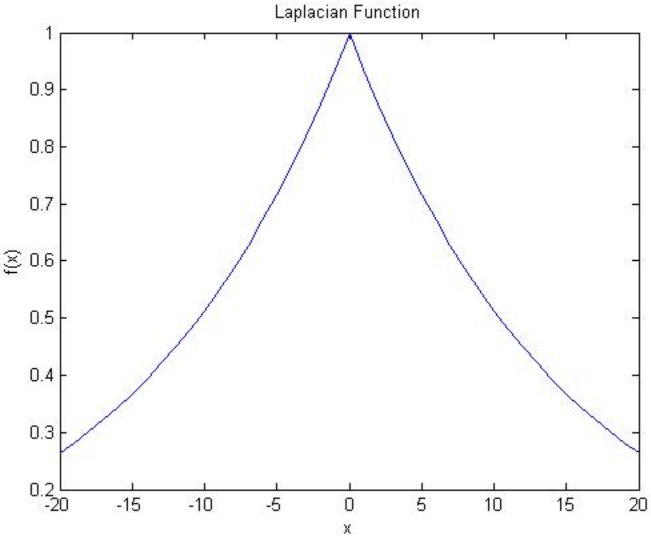

Figure 7.5: Laplacian Distribution Function with $A = 1, \sigma^2 = 15$, and $\beta = 1$

As a result of this empirical distribution, we can conclude that wavelet transforms of real-life signals and images have a small number of non-zero coefficients

$$\|x\|_0 \overset{\text{def}}{=} \#\{i | x_i \neq 0\}.$$

This sparsity property is used, e.g., in the compressive sensing (CS) approach to data compression, channel coding, data acquisition, inverse problems, etc.; see, e.g., [13]. Because of the sparsity property, wavelet transform forms the basis of the current image compression standard JPEG 2000. Specifically, JPEG 2000 uses the following EZW idea.

EZW idea. Based on the above detailed explanation of the wavelet transform, we can now explain, in detail, the EZW idea.

Let us start with the 1-D case. If the coefficient $(x_1 + x_2) - (x_3 + x_4)$ – that cover the locations 1, 2, 3, and 4 – is approximately equal to 0, then we expect all other coefficients covering these locations to be also equal to 0, i.e., we expect that $x_1 - x_2 \approx 0$ and $x_3 - x_4 \approx 0$.

In the 2-D case, we can form a similar hierarchical parent-child relation between the coefficients corresponding to different subregions of the original image; see Fig. 7.6.

7.1.1 How to Incorporate EZW Into Blind Deconvolution: First Idea

Since the EZW techniques are efficient in image processing, a natural idea is to use these techniques in blind deconvolution as well. How can we incorporate EZW ideas into the above blind deconvolution algorithms?

The main idea behind EZW approach is that if a certain wavelet coefficient is equal to 0, the all the coefficients corresponding to its "children" should also be equal to 0, as well as coefficients corresponding to "children" of "children" nodes, etc.

One possible way of enforcing this relation is to make sure that the difference between the wavelet coefficients corresponding to a node and to its children should not be too large. This way, if the wavelet coefficient corresponding to the original node is 0, the coefficients corresponding to its children should also be small.

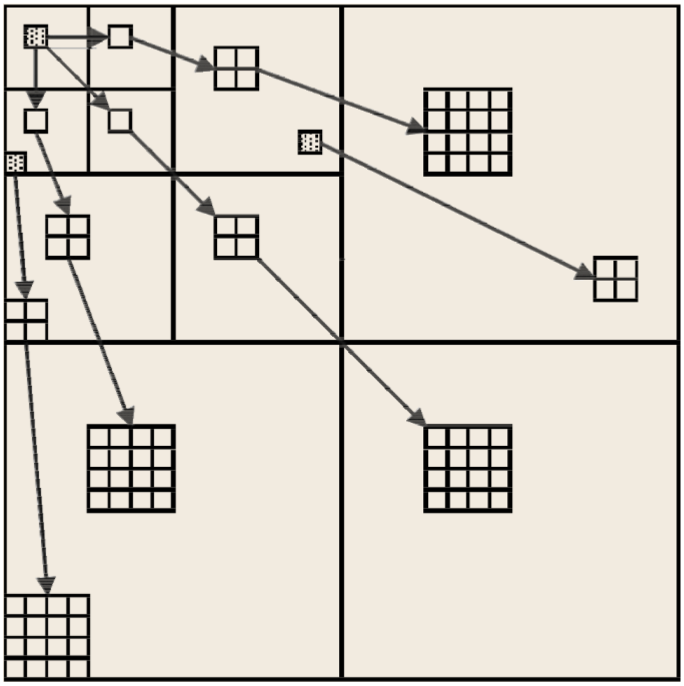

Figure 7.6: The hierarchial structure of the 3rd level of decomposition of the Wavelet transform [40]

So, in addition to minimizing the original objective function J, we should also minimize the differences $(q - q')^2$ between the corresponding pairs of values. A natural way to solve the corresponding multi-objective optimization problem is to combine all these objectives, with an appropriate weights. In other words, to the original objective function J, we add the differences $(q - q')^2$ multiplied by appropriate weights.

In our experiments, we use three levels of wavelet transforms, so we have two types of parent-child relations:

- between the level-3 nodes (LH_3, HL_3, and HH_3) and their level-2 children (LH_3, HL_3, and HH_3); and
- between the level-2 nodes (LH_2, HL_2, and HH_2) and their level-1 children (LH_1, HL_1, and HH_1).

We combine the differences between the level-3-level-2 pairs with some weight w_0, and the differences between the level-2-level-1 pairs with some weight w_1. Since there are many more level-2-level-1 pairs than level-3-level-2 pairs, we selected a larger weight $w_0 > w_1$ for level-3-level-2 pairs: otherwise, their contribution to the overall objective function would be too small in comparison with the joint contribution of level-2-level-1 pairs.

We selected the values of the weights empirically, as the values for which the reconstruction results were the best. Specifically, we selected $w_0 = 0.0004$ and $w_1 = 0.0003$.

Let us illustrate this idea on the simplified 1-D case; see Fig. 7.7.

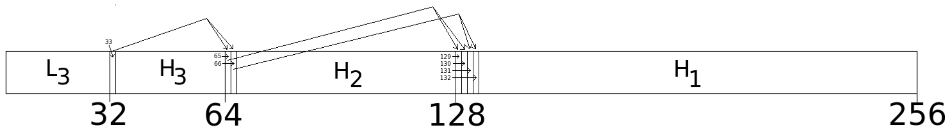

Figure 7.7: 1-D Wavelet hierarchy map

In this case, a level-3 coefficient located in position 33 is a parent to the two level-2 coefficients located in positions 65 and 66. So, to the original objective function, we add the squares of the two differences:

- the square of the difference between the values of the coefficients 33 and 65, and
- the square of the difference between the values of the coefficients 33 and 66.

Both squares are added with the weight w_0.

The level-2 coefficient in position 65 is, in its turn, a parent to the two level-1 coefficients located in positions 129 and 130. So, to the original objective function, we add the squares of the corresponding two differences:

- the square of the difference between the values of the coefficients 65 and 129, and
- the square of the difference between the values of the coefficients 65 and 130.

Both squares are added with the weight w_1.

Similarly, the level-2 coefficient in position 66 is, in its turn, a parent to the two level-1 coefficients located in positions 131 and 132. So, to the original objective function, we add the squares of the corresponding two differences:

- the square of the difference between the values of the coefficients 66 and 131, and
- the square of the difference between the values of the coefficients 66 and 132.

Both squares are also added with the weight w_1.

The resulting sum of squares of the differences can be described, in the matrix form, as $\|\mathbf{F}a\|_2^2$, where each row of the matrix $\mathbf{F}$ describes one of the differences. Each row of the the of the matrix $\mathbf{F}$ will have exactly two non-zero values: one value equal to the corresponding weight w_i and the other value equal to $-w_i$:

$$\begin{bmatrix} f_{11} & f_{12} & f_{13} & f_{14} & \cdots & f_{1N} \\ f_{21} & f_{22} & f_{23} & f_{24} & \cdots & f_{2N} \\ f_{31} & f_{32} & f_{33} & f_{34} & \cdots & f_{3N} \\ f_{41} & f_{42} & f_{43} & f_{44} & \cdots & f_{4N} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{M1} & f_{M2} & f_{M3} & f_{M4} & \cdots & f_{MN} \end{bmatrix},$$

where N is the size of the signal (or image), which in this 1-D case is 256, and M is the number of parent-child pairs. The non-zero entries of the matrix $\mathbf{F}$ are given in Table 7.1.

The resulting term is added to the objective function used in the blind deconvolution method. Specifically, at each step of the corresponding iterative method, instead of finding the next iteration a^{k+1} by optimizing the original objective function

$$\frac{\beta}{2} \cdot \|y - \mathbf{W}a\|_2^2 + \frac{\eta}{2} \cdot \|\mathbf{W}a - \mathbf{H}x\|_2^2 + \tau \cdot \|a\|_1,$$

we instead find a^{k+1} by optimizing the new expression

$$\frac{\beta}{2} \cdot \|y - \mathbf{W}a\|_2^2 + \frac{\eta}{2} \cdot \|\mathbf{W}a - \mathbf{H}x\|_2^2 + \frac{\zeta}{2} \cdot \|\mathbf{F}a\|_2^2 + \tau \cdot \|a\|_1. \tag{7.1.1}$$

For optimization purposes, it is convenient to reformulate this expression in the following equivalent form:

$$a^{k+1} = \arg\min_a \left(\frac{\beta}{2} \cdot \|y' - \rho' a\|_2^2 + \tau \cdot \|a\|_1 \right), \tag{7.1.2}$$

Table 7.1: Values of the matrix $\mathbf{F}$

i	j_1	f_{i,j_1}	j_2	f_{i,j_2}
1	33	w_0	65	$-w_0$
2	33	w_0	66	$-w_0$
3	34	w_0	67	$-w_0$
4	34	w_0	68	$-w_0$
5	35	w_0	69	$-w_0$
6	35	w_0	70	$-w_0$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
191	128	w_1	255	$-w_1$
192	128	w_1	256	$-w_1$

where we denoted

$$y' = \begin{bmatrix} \sqrt{\frac{\beta}{2}}y \\ \\ \sqrt{\frac{\eta^k}{2}}\mathbf{H}^k x^k \\ \\ \mathbf{0} \end{bmatrix}$$

and

$$\rho' = \begin{bmatrix} \sqrt{\frac{\beta}{2}}\mathbf{W} \\ \\ \sqrt{\frac{\eta^k}{2}}\mathbf{W} \\ \\ \sqrt{\frac{\zeta}{2}}\mathbf{F} \end{bmatrix}.$$

This form enables us to find the optimal a by using same MatLab algorithm *l1-ls* as is used in solving the original optimization problem; see [3].

7.1.2 How to Incorporate EZW Into Blind Deconolution: Second Idea

In our first idea, we add an additional term to the original objective function. An alternative approach is to appropriately modify the existing terms in this objective function. Specifically, in the original objective function, we have a term

$$\|a\|_1 = |a_1| + |a_2| + \ldots$$

that reflects the general sparsity requirement – that

- either $a_1 = 0$,
- or $a_2 = 0, \ldots$
- or $a_n = 0$,

and that as many of these terms as possible should be equal to 0.

Now, the sparsity requirement is somewhat modified. For example, in the above 1-D case, we are no longer simply requiring that either $a_{33} =$ or $a_{65} = 0$, etc: we also require that if $a_{33} = 0$, then all its children should be equal to 0 as well. In other words, we require that:

- either $a_{129} = 0$,
- or $a_{130} = 0$,
- or $a_{131} = 0$,
- or $a_{132} = 0$,
- or $a_{65} = a_{129} = a_{130} = 0$,

- or $a_{66} = a_{131} = a_{132} = 0$,
- or $a_{33} = a_{65} = a_{66} = a_{129} = a_{130} = a_{131} = a_{132} = 0$.

In practice, we may not achieve exact 0s, but at least we want these values to be as small as possible, i.e., smaller than some threshold δ. For two quantities q and q', the corresponding two conditions $|q| \le \delta$ and $|q'| \le \delta$ are equivalent to requiring that $\max(|q|, |q'|) \le \delta$. Thus, each requirement $q = q' = 0$ can be naturally described by adding the corresponding term $\max(|q|, |q'|)$ to the minimized expression $\|a\|_1$.

In the above example, this means that instead of the original sum

$$|a_{33}| + |a_{65}| + |a_{66}| + |a_{129}| + |a_{130}| + |a_{131}| + |a_{132}|,$$

we now minimize the modified expression

$$|a_{129}| + |a_{130}| + |a_{131}| + |a_{132}| +$$

$$\max(|a_{65}|, |a_{129}|, |a_{130}|) + \max(|a_{66}|, |a_{131}|, |a_{132}|) +$$

$$\max(|a_{32}, |a_{65}|, |a_{66}|, |a_{131}|, |a_{132}|).$$

Similarly to the first idea, we can use some weights to compensate for the difference in number of terms corresponding to different levels. In particular, in the 3-level case, we can this use the expression

$$w_2 \cdot (|a_{129}| + |a_{130}| + |a_{131}| + |a_{132}|) +$$

$$w_1 \cdot (\max(|a_{65}|, |a_{129}|, |a_{130}|) + \max(|a_{66}|, |a_{131}|, |a_{132}|)) +$$

$$w_0 \cdot \max(|a_{32}, |a_{65}|, |a_{66}|, |a_{131}|, |a_{132}|).$$

Let us show that this ideas indeed reflects the EZW intent. Indeed, what we want to avoid is a situation in which, e.g., $a_{65} = 0$ but $a_{129} = a_{130} \ne 0$. Let us therefore consider two possible solutions with the same number of non-zero coefficients:

- One solution in which the three non-zero coefficients are placed at random, e.g., $a_{66} = a_{129} = a_{130} = 0$, the rest are equal to 1.

- Another is a more desired solution, in which a 0 in a parent coefficient implies a 0 in all its children; as such a solution, we take $a_{65} = a_{129} = a_{130} = 0$, all the other coefficients are equal to 1.

For the first solution, the above objective function is equal to $2w_2+2w_1+w_0$. For the second solution, the value of this function is $2w_2+w_1+w_0$ which is smaller. Thus, indeed, the above objective function favors the EZW-compliant arrangements of the non-zero coefficient.

In general, we replace the term $\|a\|_1 = |a_1|+\ldots$ in the original objective function by the above-described weighted combination of max terms. To minimize the resulting expression, we used the same idea as is used to minimize the ℓ^p-terms in the original method: namely, for large P, $\max(|q|,\ldots,|q'|)$ is close to $(|q|^P+\ldots+|q'|^P)^{1/P}$. We can represent each term as the ratio

$$\frac{|q|^P+\ldots+|q'|^P}{(|q|^P+\ldots+|q'|^P)^{1-1/P}},$$

where the denominator can be computed based on the values from the previous iteration. Similarly, each term $|q|^P$ can be represented as the ratio

$$\frac{|q|^2}{|q|^{2-P}}.$$

Thus, we get the easy-to-optimize sum of squares, with coefficients determined by the previous iteration.

7.2 Numerical Results of Applying the Two Ideas

A general description. The results of testing both ideas are given in the following tables. We tested each of these ideas in two versions:

- by adding this idea to the original algorithm, and

- by adding this idea to the algorithm from Chapter 6 (that makes the procedure more rotation-invariant).

As we have mentioned earlier, the results are not statistically significantly better than the method without EZW, but we hope that eventually, the EZW-related ideas will lead to a better deconvolution.

Results of the first method, without rotation-invariance. For the first method, as we have mentioned, the weights were chosen as: $w_0 = 0.0004$, $w_1 = 0.0003$, and $w_2 = 0.0002$.

For the first method, we get the following 24 values of the mean square difference $\|x - \widehat{x}\|_2$ between the original image x and the reconstructed image $\widehat{x}$:

$$1194.89, \quad 1192.85, \quad 1197.91, \quad 1192.35, \quad 1201.54, \quad 1193.34, \quad 1203.41,$$

$$1209.80, \quad 1200.76, \quad 1196.44, \quad 1192.73, \quad 1199.98, \quad 1194.99, \quad 1192.17,$$

$$1202.06, \quad 1200.39, \quad 1192.62, \quad 1193.16, \quad 1192.73, \quad 1192.12, \quad 1195.23$$

$$1201.90, \quad 1191.25, \quad 1191.12.$$

The average of these values is 1196.24, which is larger than the average difference corresponding to the original algorithm (and thus, larger than the average different corresponding to the algorithm from Chapter 6).

Results of the first method, with rotation-invariance. If add the new idea to the rotation-invariant algorithm from Chapter 6, we get the following values of the difference:

$$1188.45, \quad 1193.27, \quad 1189.12, \quad 1190.34, \quad 1196.90, \quad 1192.13, \quad 1187.76$$

$$1191.22, \quad 1202.52, \quad 1189.89, \quad 1196.28, \quad 1190.20, \quad 1188.60, \quad 1187.66,$$

$$1189.09, \quad 1190.94, \quad 1187.82, \quad 1189.41, \quad 1187.74, \quad 1186.64, \quad 1188.81,$$

$$1194.34, \quad 1192.68, \quad 1195.79.$$

The average of these values is 1191.15, which is smaller than the average difference corresponding to the original algorithm, but larger than the average 1191.01 for the algorithm from Chapter 6.

Results of the second method, without rotation-invariance. For the second method, the optimal deconvolution was attained when we selected the weights $w_0 = 10^{-5}$, $w_1 = 5 \cdot 10^{-6}$, and $w_2 = 10^{-6}$.

For this method, we get the following 26 values:

1190.37, 1193.62, 1192.15, 1195.06, 1191.27, 1190.42, 1192.78

1192.17, 1196.84, 1202.14, 1194.89, 1192.26, 1222.88, 1198.26,

1191.24, 1191.92, 1195.36, 1198.59, 1191.92, 1201.53, 1193.35,

1195.91, 1203.40, 1209.80, 1190.79, 1200.74.

The average of these values is 1196.53, which is slightly better than for the first method but still larger than the average difference corresponding to the original algorithm (and thus, larger than for the algorithm from Chapter 6).

Results of the second method, with rotation-invariance. For this method, we get the following values of the difference:

1191.99, 1189.36, 1191.76, 1189.04, 1189.23, 1189.36, 1198.58

1193.91, 1188.08, 1192.04, 1191.55, 1200.88, 1191.15, 1195.78,

1190.74, 1188.81, 1188.60, 1190.42, 1189.07, 1191.83, 1188.29,

1187.36, 1189.76, 1188.82, 1193.69, 1199.31.

The average of these values is 1191.52, which is smaller than the average difference corresponding to the original algorithm, but larger than the average difference for the algorithm from Chapter 6.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In many practical situations, we need to reconstruct an image in situations in which we do not have any information about the blurring function. There exist empirically successful algorithms for such blind image deconvolution. However, the use of these methods is hindered by the lack of convincing theoretical justification for the use of the corresponding techniques of sparsity and ℓ^p-regularization in blind image deconvolution. Without such a theoretical justifications, potential users are not sure that these methods will work successfully on their images.

In this dissertation, we have provided such a theoretical justification. This will hopefully improve the acceptance and usage of the current blind image deconcovolution techniques.

Our theoretical analysis has also led us to the possibility to improve the current state-of-the-art blind image deconvolution techniques. As a result, we propose a new method which leads to a 10% more accurate image reconstruction.

8.2 Future Work

While the current methods are reasonably efficient, they are not yet perfect. For example, while the current methods correctly reconstructs the standard “Cameraman” image from its blurred version, when we rotated this image, the quality of the reconstruction drastically decreased.

We hope that our theoretical analysis will help in designing even better blind image decomposition techniques. For example, since, as we have shown, replacing the first-order regularization terms with a rotation-invariant expression improves the image, it may be a good idea to try a similar replacement for regularization terms containing second order differences.

References

[1] J. Aczél and J. Dhombres, *Functional Equations in Several Variables*, Cambridge University Press, 2008.

[2] B. Amizic, S. Derin Babacan, R. Molina, and A. K. Katsaggelos. "Sparse bayesian blind image deconvolution with parameter estimation", *Proceedings of the 2010 18th European Signal Processing Conference*, August 2010, pp. 626–630.

[3] B. Amizic, L. Spinoulas, R. Molina, and A. K. Katsaggelos, "Compressive blind image deconvolution", *IEEE Transactions in Image Processing*, 2013, Vol. 22, pp. 3994–4006.

[4] M. Argaez, C. Ramirez, and R. Sanchez, "An l_1-algorithm for underdetermined systems and applications. *Proceedings of the 2011 Annual Conference of the North American Fuzzy Information Processing Society*, March 2011, pp. 1–6.

[5] M. Argaez, R. Sanchez, and C. Ramirez, "Face recognition from incomplete measurements via l_1 optimization", *American Journal of Computational Mathematics*, 2012, Vol. 2, No. 4, pp. 287–294.

[6] D. P. Bertsekas, *Nonlinear Programming*, Athena Scientific, 1999.

[7] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge Unversity Press, 2004.

[8] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images", *SIAM Review*, 2009, Vol. 51, No. 1, pp. 34-81.

[9] J. Cai, H. Ji, C. Liu, and Z. Shen. "Blind motion deblurring from a single image using sparse approximation", *Proceedings on the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, June 2009, pp. 109–114.

[10] E. J. Candès, J. Romberg and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements", *Comm. Pure Appl. Math.*, 2006, Vol. 59, pp. 1207–1223.

[11] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information", *IEEE Transactions on Information Theory*, 2006, Vol. 52, No. 2, pp. 489–509.

[12] E. J. Candès and T. Tao, "Decoding by linear programming", *IEEE Transactions on Information Theory*, 2005, Vol. 51, No. 12, pp. 4203–4215.

[13] E. J. Candès and M. B. Wakin, "An Introduction to compressive sampling", *IEEE Signal Processing Magazine*, 2008, Vol. 25, No. 2, pp. 21–30.

[14] F. Cervantes, B. Usevitch, L. Valera, and V. Kreinovich, "Why sparse? fuzzy techniques explain empirical efficiency of sparsity-based data- and image-processing algorithms", *Proceedings of the 2016 World Congress on Soft Computing*, Berkeley, California, May 22–25, 2016, to appear.

[15] F. Cervantes, B. Usevitch, and V. Kreinovich, "Why ℓ_p-methods in signal and image processing: a fuzzy-based explanation", *Proceedings of the Annual Conference of the North American Fuzzy Information Processing Society NAFIPS'2016*, El Paso, Texas, October 31 – November 4, 2016, to appear.

[16] G. B. Dantzig and M. N. Thapa, *Linear Programming 1*, Springer-Verlag, New York, 1997.

[17] D. L. Donoho, "Compressed sensing", *IEEE Transactions on Information Theory*, 2005, Vol. 52, No. 4, pp. 1289–1306.

[18] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE Signal Processing Magazine*, 2008, Vol. 25, No. 2, pp. 83–91.

[19] T. Edeler, K. Ohliger, S. Hussmann, and A. Mertins, “Super-resolution model for a compressed-sensing measurement setup”, *IEEE Transactions on Instrumentation and Measurement*, 2012, Vol. 61, No. 5, pp. 1140–1148.

[20] M. Elad, *Sparse and Redundant Representations*, Springer Verlag, 2010.

[21] R. C. Gonzales and R. E. Woods, *Digital Image Processing*, Prentice Hall, 2008.

[22] E. T. Jaynes and G. L. Bretthorst, *Probability Theory: The Logic of Science*, Cambridge University Press, Cambridge, UK, 2003.

[23] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, “An interior-point method for large-scale l^1-regularized least squares”, *IEEE Journal on Selected Topics in Signal Processing*, 2007, Vol. 1, No. 4, pp. 606–617.

[24] G. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic*. Prentice Hall, Upper Saddle River, New Jersey, 1995.

[25] D. Krishnan and R. Fergus, “Fast image deconvolution using hyper-Laplacian priors”, In: Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta (eds.), *Proceedings of the 23th Annual Conference on Neural Information Processing Systems NIPS’2009*, Vancouver, Canada, December 7–12, 2009.

[26] A. Levin, R. Fergus, F. Durand, and W. T. Freeman, “Image and depth from a conventional camera with a coded aperture”, *ACM SIGGRAPH Transactions on Graphics*, 2007, Vol. 26, No. 3.

[27] J. Ma and F. X. Le Dimet, “Deblurring from highly incomplete measurements for remote sensing”, *IEEE Transactions on Geosciences Remote Sensing*, 2009, Vol. 47, No. 3, pp. 792–802.

[28] B. Mandelbrot, *The Fractal Geometry of Nature*, Freeman, San Francisco, California, 1983.

[29] L. McMackin, M. A. Herman, B. Chatterjee, and M. Weldon, "A high-resolution swir camera via compressed sensing", *Proceedings of SPIE*, 2012, Vol. 8353, No. 1, p. 8353-03.

[30] R. Molina, J. Mateos, and A. K. Katsaggelos, "Blind deconvolution using a variational approach to parameter, image, and blur estimation", *IEEE Transactions in Image Processing*, 2006, Vol. 15, No. 12, pp. 3715–3727.

[31] B. K. Natarajan, "Sparse approximate solutions to linear systems", *SIAM Journal on Computing*, 1995, Vol. 24, pp. 227–234.

[32] H. T. Nguyen, V. Kreinovich, and P. Wojciechowski, "Strict Archimedean t-Norms and t-Conorms as Universal Approximators", *International Journal of Approximate Reasoning*, 1998, Vol. 18, Nos. 3–4, pp. 239–249.

[33] H. T. Nguyen and E. A. Walker, *A First Course in Fuzzy Logic*, Chapman and Hall/CRC, Boca Raton, Florida, 2006.

[34] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Verlag, New York, 2006.

[35] C. Ramirez and M. Argaez, "An l_1 minimization algorithm for non-smooth regularization in image processing", *Signal and Image Processing*, 2015, Vol. 9, No. 2, pp. 373–386.

[36] J. M. Shapiro, "Embedded image coding using zerotrees of wavelet coefficients", *IEEE Transactions on Signal Processing*, 1993, Vol. 41, No. 12, pp. 3445–3462.

[37] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-Posed Problems*, V. H. Winston & Sons, Washington, DC, 1977.

[38] Y. Tsaig and D. Donoho, "Compressed sensing", *IEEE Transactions on Information Theory*, 2006, Vol. 52, No. 4, pp. 1289–1306.

[39] S. E. Umbaugh, *Computer Vision*, CRC Press, 2005.

[40] B. E. Usevitch, "A tutorial on modern lossy wavelet image compression: foundations of JPEG 2000", *IEEE Signal Processing Magazine*, 2001, Vol. 18, No. 5, pp. 22–35.

[41] L. Xiao, J. Shao, L. Huang, and Z. Wei, "Compounded regularization and fast algorithm for compressive sensing deconvolution", *Proceedings of the 2011 6th International Conference on Image and Graphics ICIG'2011*, Hefei, China, August 12–15, 2011, pp. 616–621.

[42] W. Yin, S. Morgan, J. Yang, and Y. Zhang, "Practical compressive sensing with Toeplitz and circulant matrices", *Proceedings of the 2010 SPIE Conference on Visual Communications and Image Processing*, July 2010.

[43] L. A. Zadeh, "Fuzzy sets", *Information and Control*, 1965, Vol. 8, pp. 338–353.

Appendix A

Matlab Code for the Original Algorithm and for Its Proposed Modifications

A.1 Matlab Code for the Original State-of-the-Art Image Deconvolution Algorithm

```
clc, clear all, close all;
% Creating the mage
f = double(imread('cameraman.tif'));
[m,n]=size(f);
x_orig = f;
N=m*n;
figure(1)
pause(0.01)
imagesc(f), colormap gray;
h1=fspecial('gaussian',17,sqrt(5));
h_orig = h1;
[Wav, L, H, k]=wavelet_factory4('db1', n,n/8);
W=Wav';
sigma=1e-3;
```

```
y=convolution(f,h1,1)+sigma*randn(m,n);
a=Wav*y;
figure(2)
imagesc(y), colormap gray;
pause(0.01)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
lag = fspecial('laplacian',0.0);

C=zeros(m,n);

primerIndiceX=floor((m-3)/2);
ultimoIndiceX=primerIndiceX+3;

primerIndiceY=floor((n-3)/2);
ultimoIndiceY=primerIndiceY+3;

C(primerIndiceX+1:ultimoIndiceX,primerIndiceY+1:ultimoIndiceY)=lag;

C=fftshift(C);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Using the program
% Estimating a new filter
h=fspecial('gaussian',17,sqrt(2));
h_est = h;
alpha=1;
p=0.8;
B=rand(N,5);
eta=1042;
```

```
gamma=5e5;

exp_a=0.8;

thr_e=0.0001;

beta=sigma^2;

tau=0.125;

theta=1.3;


dxf=[1 -1]; DXF=zeropad(dxf,m,n);

dyf=[1;-1]; DYF=zeropad(dyf,m,n);

dxxf=[-1, 2, -1]; DXXF=zeropad(dxxf,m,n);

dyyf=[-1; 2; -1]; DYYF=zeropad(dyyf,m,n);

dxyf=[-1 1;1 -1]; DXYF=zeropad(dxyf,m,n);


xkl=y;

for k=1:10


  xklold=xkl;

  Wa=W*a;

  for l=1:2


   weight_x=reshape(B(:,1),m,n);weight_x=weight_x.^(p/2-1);

   weight_y=reshape(B(:,2),m,n);weight_y=weight_y.^(p/2-1);

   weight_xx=reshape(B(:,3),m,n);weight_xx=weight_xx.^(p/2-1);

   weight_yy=reshape(B(:,4),m,n);weight_yy=weight_yy.^(p/2-1);

   weight_xy=reshape(B(:,5),m,n);weight_xy=weight_xy.^(p/2-1);


   H=zeropad(h,m,n);
```

```
Ax=eta*conj(fft2(H)).*fft2(H);

Ax=Ax+alpha*p*conj(fft2(DXF)).*ifft2(fft2(fft2(weight_x)).*...
              fft2(fft2(DXF)))/(m*n);
Ax=Ax+alpha*p*conj(fft2(DYF)).*ifft2(fft2(fft2(weight_y)).*...
              fft2(fft2(DYF)))/(m*n);
Ax=Ax+0.5*alpha*p*conj(fft2(DXXF)).*ifft2(fft2(fft2(weight_xx)).*...
              fft2(fft2(DXXF)))/(m*n);
Ax=Ax+0.5*alpha*p*conj(fft2(DYYF)).*ifft2(fft2(fft2(weight_yy)).*...
              fft2(fft2(DYYF)))/(m*n);
Ax=Ax+0.5*alpha*p*conj(fft2(DXYF)).*ifft2(fft2(fft2(weight_xy)).*...
              fft2(fft2(DXYF)))/(m*n);

bx=eta*conj(fft2(H)).*fft2(Wa);

xklf=bx./Ax;
xkl=real(ifft2(xklf));

xkl=reshape(xkl,m,n);

xklT=xkl;
xklT=fftshift(xklT);
Ahf=eta*conj(fft2(xklT)).*fft2(xklT)+gamma*conj(fft2(C)).*fft2(C);
bhf=eta*conj(fft2(xklT)).*fft2(Wa);
hf=bhf./Ahf;
h=real(ifft2(hf));
h=h(120:136,120:136);
figure(4)
```

```
 subplot(1,2,1)
 mesh(h1)
 subplot(1,2,2)
 mesh(h);
 pause(0.1)

 dx=convolution(xkl,dxf,1);
 dy=convolution(xkl,dyf,1);
 dxx=convolution(xkl,dxxf,1);
 dyy=convolution(xkl,dyyf,1);
 dxy=convolution(xkl,dxyf,1);

 W1=max(abs(dx).^2,thr_e); B(:,1)=W1(:);
 W2=max(abs(dy).^2,thr_e); B(:,2)=W2(:);
 W3=max(abs(dxx).^2,thr_e);B(:,3)=W3(:);
 W4=max(abs(dyy).^2,thr_e);B(:,4)=W4(:);
 W5=max(abs(dxy).^2,thr_e);B(:,5)=W5(:);
end

norma=norm(xkl-xklold)/norm(xklold);
if norma<1e-5
   break
end
figure(3)
imagesc(xkl); colormap gray
pause(0.01)
HX=convolution(xkl,h,1);
```

```
  for i=1:n
    Y(:,i)=[sqrt(beta/2)*y(:,i);sqrt(eta/2)*HX(:,i)];
    PHI=[sqrt(beta/2)*eye(n);sqrt(eta/2)*eye(n)];
    A=PHI*W;
    b=Y(:,i);

    [a(:,i),status,history] = l1_ls(A,b,tau);
  end

  eta=eta*theta;
end

disp(['The d(xrec,xorig) ', num2str(norm(x_orig-xkl))]);
disp(['The d(hrec,horig) is ', num2str(norm(h_orig-h_est))]);
```

A.2 Matlab Code for the Newly Proposed Rotation-Invariant Modification of the State-of-the-Art Image Deconvolution Algorithm

```
clc clear all, close all;
% Creating the mage
f = double(imread('cameraman.tif'));
[m,n]=size(f);
x_orig = f;
N=m*n;
figure(1)
```

```
pause(0.01)
imagesc(f), colormap gray;
h1=fspecial('gaussian',17,sqrt(5));
h_orig = h1;
[Wav, L, H, k]=wavelet_factory4('db1', n,n/8);
W=Wav';
sigma=1e-3;
y=convolution(f,h1,1)+sigma*randn(m,n);
a=Wav*y;
figure(2)
imagesc(y), colormap gray;
pause(0.01)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
lag = fspecial('laplacian',0.0);

C=zeros(m,n);


primerIndiceX=floor((m-3)/2);
ultimoIndiceX=primerIndiceX+3;


primerIndiceY=floor((n-3)/2);
ultimoIndiceY=primerIndiceY+3;


C(primerIndiceX+1:ultimoIndiceX,primerIndiceY+1:ultimoIndiceY)=lag;


C=fftshift(C);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Using the program
% Estimating a new filter
h=fspecial('gaussian',17,sqrt(2));
h_est = h;
alpha=1;
p=0.8;
B=rand(N,5);
eta=1042;
gamma=5e5;
exp_a=0.8;
thr_e=0.0001;
beta=sigma^2;
tau=0.125;
theta=1.3;

dxf=[1 -1]; DXF=zeropad(dxf,m,n);
dyf=[1;-1]; DYF=zeropad(dyf,m,n);
dxxf=[-1, 2, -1]; DXXF=zeropad(dxxf,m,n);
dyyf=[-1; 2; -1]; DYYF=zeropad(dyyf,m,n);
dxyf=[-1 1;1 -1]; DXYF=zeropad(dxyf,m,n);

xkl=y;
for k=1:10
  xklold=xkl;
  Wa=W*a;
  for l=1:2

   weight_x=reshape(B(:,1),m,n);weight_x=weight_x.^(p/2-1);
   weight_y=reshape(B(:,2),m,n);weight_y=weight_y.^(p/2-1);
```

```
weight_xx=reshape(B(:,3),m,n);weight_xx=weight_xx.^(p/2-1);
weight_yy=reshape(B(:,4),m,n);weight_yy=weight_yy.^(p/2-1);
weight_xy=reshape(B(:,5),m,n);weight_xy=weight_xy.^(p/2-1);

H=zeropad(h,m,n);

Ax=eta*conj(fft2(H)).*fft2(H);

Ax=Ax+alpha*p*conj(fft2(DXF)).*ifft2(fft2(fft2(weight_x)).*...
              fft2(fft2(DXF)))/(m*n);
Ax=Ax+alpha*p*conj(fft2(DYF)).*ifft2(fft2(fft2(weight_y)).*...
              fft2(fft2(DYF)))/(m*n);
Ax=Ax+0.5*alpha*p*conj(fft2(DXXF)).*ifft2(fft2(fft2(weight_xx)).*...
              fft2(fft2(DXXF)))/(m*n);
Ax=Ax+0.5*alpha*p*conj(fft2(DYYF)).*ifft2(fft2(fft2(weight_yy)).*...
              fft2(fft2(DYYF)))/(m*n);
Ax=Ax+0.5*alpha*p*conj(fft2(DXYF)).*ifft2(fft2(fft2(weight_xy)).*...
              fft2(fft2(DXYF)))/(m*n);

bx=eta*conj(fft2(H)).*fft2(Wa);

xklf=bx./Ax;
xkl=real(ifft2(xklf));

xkl=reshape(xkl,m,n);

xklT=xkl;
xklT=fftshift(xklT);
```

```
 Ahf=eta*conj(fft2(xklT)).*fft2(xklT)+gamma*conj(fft2(C)).*fft2(C);
 bhf=eta*conj(fft2(xklT)).*fft2(Wa);
 hf=bhf./Ahf;
 h=real(ifft2(hf));
 h=h(120:136,120:136);
 figure(4)
 subplot(1,2,1)
 mesh(h1)
 subplot(1,2,2)
 mesh(h);
 pause(0.1)

 dx=convolution(xkl,dxf,1);
 dy=convolution(xkl,dyf,1);
 dxx=convolution(xkl,dxxf,1);
 dyy=convolution(xkl,dyyf,1);
 dxy=convolution(xkl,dxyf,1);

 W1=.5*max(abs(dx).^2+abs(dy).^2,thr_e); B(:,1)=W1(:);
 W2=W1; B(:,2) = W2(:);
 W3=max(abs(dxx).^2,thr_e);B(:,3)=W3(:);
 W4=max(abs(dyy).^2,thr_e);B(:,4)=W4(:);
 W5=max(abs(dxy).^2,thr_e);B(:,5)=W5(:);
end

norma=norm(xkl-xklold)/norm(xklold);

if norma<1e-2
```

```
     break
  end
  figure(3)
  imagesc(xkl); colormap gray
  pause(0.01)
  HX=convolution(xkl,h,1);

  for i=1:n
    Y(:,i)=[sqrt(beta/2)*y(:,i);sqrt(eta/2)*HX(:,i)];
    PHI=[sqrt(beta/2)*eye(n);sqrt(eta/2)*eye(n)];
    A=PHI*W;
    b=Y(:,i);
    [a(:,i),status,history] = l1_ls(A,b,tau);
  end

  eta=eta*theta;
end

disp([’The d(xrec,xorig) ’, num2str(norm(x_orig-xkl))]);
disp([’The d(hrec,horig) is ’, num2str(norm(h_orig-h_est))]);
```

A.3 Matlab Code for the Method Using the First EZW Idea

```
clc, clear all, close all;
% Creating the mage
```

```
f = double(imread('cameraman.tif'));

[m,n]=size(f);

x_orig = f;

N=m*n;

figure(1)

pause(0.01)

imagesc(f), colormap gray;

h1=fspecial('gaussian',17,sqrt(5));

h_orig = h1;

level = 3;

[Wav, Low, High, k]=wavelet_factory4('db1', n,n/(2^level));

W=Wav';

sigma=1e-3;

y=convolution(f,h1,1)+sigma*randn(m,n);

a=Wav*y;

figure(2)

imagesc(y), colormap gray;

pause(0.01)


% construir matriz F

index = [];

start_index = n/(2^level)+1;

for i = start_index:n/2

    index = [index;i;i];

end


index(:,2)=[n/((level-1)^2)+1:n]';
```

```
F=zeros(size(index,1),n);
P = ones(size(F,1),1);
P(65:end) = .0002;
P(32:64) = .0003;
P(1:31) = .0004;
for i=1:length(index(:,1))
    F(i,index(i,1))=1;
    F(i,index(i,2))=-1;
end
F = diag(P)*F;
%%%%%%%%%%%%%%%%%%%%%%%%%%
lag = fspecial('laplacian',0.0);

C=zeros(m,n);

primerIndiceX=floor((m-3)/2);
ultimoIndiceX=primerIndiceX+3;

primerIndiceY=floor((n-3)/2);
ultimoIndiceY=primerIndiceY+3;

C(primerIndiceX+1:ultimoIndiceX,primerIndiceY+1:ultimoIndiceY)=lag;

C=fftshift(C);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Using the program
% Estimating a new filter
h=fspecial('gaussian',17,sqrt(2));
```

```
h_est = h;
alpha=1;
p=0.8;
B=rand(N,5);
eta=1042;
gamma=5e5;
exp_a=0.8;
thr_e=0.0001;
beta=sigma^2;
tau=0.125;
theta=1.3;
omicron = 5;

dxf=[1 -1]; DXF=zeropad(dxf,m,n);
dyf=[1;-1]; DYF=zeropad(dyf,m,n);
dxxf=[-1, 2, -1]; DXXF=zeropad(dxxf,m,n);
dyyf=[-1; 2; -1]; DYYF=zeropad(dyyf,m,n);
dxyf=[-1 1;1 -1]; DXYF=zeropad(dxyf,m,n);

tic;
xkl=y;
for k=1:10

  xklold=xkl;
  Wa=W*a;

  for l=1:2
```

```
weight_x=reshape(B(:,1),m,n);weight_x=weight_x.^(p/2-1);
weight_y=reshape(B(:,2),m,n);weight_y=weight_y.^(p/2-1);
weight_xx=reshape(B(:,3),m,n);weight_xx=weight_xx.^(p/2-1);
weight_yy=reshape(B(:,4),m,n);weight_yy=weight_yy.^(p/2-1);
weight_xy=reshape(B(:,5),m,n);weight_xy=weight_xy.^(p/2-1);

H=zeropad(h,m,n);

Ax=eta*conj(fft2(H)).*fft2(H);

Ax=Ax+alpha*p*conj(fft2(DXF)).*ifft2(fft2(fft2(weight_x)).*...
              fft2(fft2(DXF)))/(m*n);
Ax=Ax+alpha*p*conj(fft2(DYF)).*ifft2(fft2(fft2(weight_y)).*...
              fft2(fft2(DYF)))/(m*n);
Ax=Ax+0.5*alpha*p*conj(fft2(DXXF)).*ifft2(fft2(fft2(weight_xx)).*...
              fft2(fft2(DXXF)))/(m*n);
Ax=Ax+0.5*alpha*p*conj(fft2(DYYF)).*ifft2(fft2(fft2(weight_yy)).*...
              fft2(fft2(DYYF)))/(m*n);
Ax=Ax+0.5*alpha*p*conj(fft2(DXYF)).*ifft2(fft2(fft2(weight_xy)).*...
              fft2(fft2(DXYF)))/(m*n);

bx=eta*conj(fft2(H)).*fft2(Wa);

xklf=bx./Ax;
xkl=real(ifft2(xklf));

xkl=reshape(xkl,m,n);
```

```
 xklT=xkl;
 xklT=fftshift(xklT);
 Ahf=eta*conj(fft2(xklT)).*fft2(xklT)+gamma*conj(fft2(C)).*fft2(C);
 bhf=eta*conj(fft2(xklT)).*fft2(Wa);
 hf=bhf./Ahf;
 h=real(ifft2(hf));
 h=h(120:136,120:136);
 figure(4)
 subplot(1,2,1)
 mesh(h1)
 subplot(1,2,2)
 mesh(h);
 pause(0.1)
 dx=convolution(xkl,dxf,1);
 dy=convolution(xkl,dyf,1);
 dxx=convolution(xkl,dxxf,1);
 dyy=convolution(xkl,dyyf,1);
 dxy=convolution(xkl,dxyf,1);

 W1=max(abs(dx).^2,thr_e); B(:,1)=W1(:);
 W2=max(abs(dy).^2,thr_e); B(:,2)=W2(:);
 W3=max(abs(dxx).^2,thr_e);B(:,3)=W3(:);
 W4=max(abs(dyy).^2,thr_e);B(:,4)=W4(:);
 W5=max(abs(dxy).^2,thr_e);B(:,5)=W5(:);
end

norma=norm(xkl-xklold)/norm(xklold);
```

```
  if norma<1e-2
     break
  end

  figure(3)
  imagesc(xkl); colormap gray
  pause(0.01)

  HX=convolution(xkl,h,1);

  for i=1:n
    Y(:,i)=[sqrt(beta/2)*y(:,i);sqrt(eta/2)*HX(:,i);zeros(size(F,1),1)];
    PHI=[sqrt(beta/2)*eye(n)*W;sqrt(eta/2)*W;sqrt(omicron/2)*F];

    A=PHI;
    b=Y(:,i);

    [a(:,i),status,history] = l1_ls(A,b,tau);
  end

  eta=eta*theta;
end
toc;
disp(['d(xorig,xrec) = ', num2str(norm(f-xkl))]);
```

A.4 Matlab Code for the Method Using the First EZW Idea Combined With Rotation-Invariance

```
clc, clear all, close all;
% Creating the mage
f = double(imread('cameraman.tif'));
[m,n]=size(f);
x_orig = f;
N=m*n;
figure(1)
pause(0.01)
imagesc(f), colormap gray;
h1=fspecial('gaussian',17,sqrt(5));
h_orig = h1;
level = 3;
[Wav, Low, High, k]=wavelet_factory4('db1', n,n/(2^level));
W=Wav';
sigma=1e-3;
y=convolution(f,h1,1)+sigma*randn(m,n);
a=Wav*y;
figure(2)
imagesc(y), colormap gray;
pause(0.01)

% construir matriz F
index = [];
start_index = n/(2^level)+1;
for i = start_index:n/2
```

```
    index = [index;i;i];
end

index(:,2)=[n/((level-1)^2)+1:n]’;

F=zeros(size(index,1),n);
P = ones(size(F,1),1);
P(65:end) = .0002;
P(32:64) = .0003;
P(1:31) = .0004;
for i=1:length(index(:,1))
    F(i,index(i,1))=1;
    F(i,index(i,2))=-1;
end
F = diag(P)*F;
%%%%%%%%%%%%%%%%%%%%%%%%%%
lag = fspecial(’laplacian’,0.0);

C=zeros(m,n);

primerIndiceX=floor((m-3)/2);
ultimoIndiceX=primerIndiceX+3;

primerIndiceY=floor((n-3)/2);
ultimoIndiceY=primerIndiceY+3;

C(primerIndiceX+1:ultimoIndiceX,primerIndiceY+1:ultimoIndiceY)=lag;
```

```
C=fftshift(C);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Using the program
% Estimating a new filter
h=fspecial(’gaussian’,17,sqrt(2));
h_est = h;
alpha=1;
p=0.8;
B=rand(N,5);
eta=1042;
gamma=5e5;
exp_a=0.8;
thr_e=0.0001;
beta=sigma^2;
tau=0.125;
theta=1.3;
omicron = 5;

dxf=[1 -1]; DXF=zeropad(dxf,m,n);
dyf=[1;-1]; DYF=zeropad(dyf,m,n);
dxxf=[-1, 2, -1]; DXXF=zeropad(dxxf,m,n);
dyyf=[-1; 2; -1]; DYYF=zeropad(dyyf,m,n);
dxyf=[-1 1;1 -1]; DXYF=zeropad(dxyf,m,n);

tic;
xkl=y;
for k=1:10
  xklold=xkl;
```

```
Wa=W*a;
for l=1:2

 weight_x=reshape(B(:,1),m,n);weight_x=weight_x.^(p/2-1);
 weight_y=reshape(B(:,2),m,n);weight_y=weight_y.^(p/2-1);
 weight_xx=reshape(B(:,3),m,n);weight_xx=weight_xx.^(p/2-1);
 weight_yy=reshape(B(:,4),m,n);weight_yy=weight_yy.^(p/2-1);
 weight_xy=reshape(B(:,5),m,n);weight_xy=weight_xy.^(p/2-1);

 H=zeropad(h,m,n);

 Ax=eta*conj(fft2(H)).*fft2(H);

 Ax=Ax+alpha*p*conj(fft2(DXF)).*ifft2(fft2(fft2(weight_x)).*...
              fft2(fft2(DXF)))/(m*n);
 Ax=Ax+alpha*p*conj(fft2(DYF)).*ifft2(fft2(fft2(weight_y)).*...
              fft2(fft2(DYF)))/(m*n);
 Ax=Ax+0.5*alpha*p*conj(fft2(DXXF)).*ifft2(fft2(fft2(weight_xx)).*...
              fft2(fft2(DXXF)))/(m*n);
 Ax=Ax+0.5*alpha*p*conj(fft2(DYYF)).*ifft2(fft2(fft2(weight_yy)).*...
              fft2(fft2(DYYF)))/(m*n);
 Ax=Ax+0.5*alpha*p*conj(fft2(DXYF)).*ifft2(fft2(fft2(weight_xy)).*...
              fft2(fft2(DXYF)))/(m*n);

 bx=eta*conj(fft2(H)).*fft2(Wa);

 xklf=bx./Ax;
 xkl=real(ifft2(xklf));
```

```
xkl=reshape(xkl,m,n);

xklT=xkl;
xklT=fftshift(xklT);
Ahf=eta*conj(fft2(xklT)).*fft2(xklT)+gamma*conj(fft2(C)).*fft2(C);
bhf=eta*conj(fft2(xklT)).*fft2(Wa);
hf=bhf./Ahf;
h=real(ifft2(hf));
h=h(120:136,120:136);
figure(4)
subplot(1,2,1)
mesh(h1)
subplot(1,2,2)
mesh(h);
pause(0.1)

dx=convolution(xkl,dxf,1);
dy=convolution(xkl,dyf,1);
dxx=convolution(xkl,dxxf,1);
dyy=convolution(xkl,dyyf,1);
dxy=convolution(xkl,dxyf,1);

W1=.5*max(abs(dx).^2+abs(dy).^2,thr_e); B(:,1)=W1(:);
W2=W1; B(:,2) = W2(:);
W3=max(abs(dxx).^2,thr_e);B(:,3)=W3(:);
W4=max(abs(dyy).^2,thr_e);B(:,4)=W4(:);
W5=max(abs(dxy).^2,thr_e);B(:,5)=W5(:);
```

```
  end

  norma=norm(xkl-xklold)/norm(xklold);
  if norma<1e-2
     break
  end
  figure(3)
  imagesc(xkl); colormap gray
  pause(0.01)
  HX=convolution(xkl,h,1);

  for i=1:n
    Y(:,i)=[sqrt(beta/2)*y(:,i);sqrt(eta/2)*HX(:,i);zeros(size(F,1),1)];
    PHI=[sqrt(beta/2)*eye(n)*W;sqrt(eta/2)*W;sqrt(omicron/2)*F];

    A=PHI;
    b=Y(:,i);

    [a(:,i),status,history] = l1_ls(A,b,tau);
  end

  eta=eta*theta;
end
toc;
disp(['d(xorig,xrec) = ', num2str(norm(f-xkl))]);
```

A.5 Matlab Code for the Method Using the Second EZW Idea

```
clc, clear all, close all; warning off;
% Creating the mage
[Wav,high,low,k]=wavelet_factory4(’db1’, 256,256/8); W=Wav’; f =
double(imread(’cameraman.tif’)); [m,n]=size(f); N=m*n; figure(1)
pause(0.01) imagesc(f), colormap gray;
h1=fspecial(’gaussian’,17,sqrt(5)); sigma=1e-3;
y=convolution(f,h1,1)+sigma*randn(m,n); a=Wav*y; figure(2)
imagesc(y), colormap gray; pause(0.01)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
lag = fspecial(’laplacian’,0.0);

C=zeros(m,n);

primerIndiceX=floor((m-3)/2); ultimoIndiceX=primerIndiceX+3;

primerIndiceY=floor((n-3)/2); ultimoIndiceY=primerIndiceY+3;

C(primerIndiceX+1:ultimoIndiceX,primerIndiceY+1:ultimoIndiceY)=lag;

C=fftshift(C);
% w(1) is w0, and it should have the largest weight
% w(2) is w1, and it should a number between w0 and w2 (w0 > w1 > w2)
% w(3) is w2, and it should have the smallest weight
w = 1e-6*[10 5 1];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Using the program
% Estimating a new filter
h=fspecial('gaussian',17,sqrt(2));

alpha=1; p=0.8; B=rand(N,5); eta=1042; gamma=5e5; exp_a=0.8;
thr_e=0.0001; beta=sigma^2; tau=0.125; theta=1.3;

dxf=[1 -1]; DXF=zeropad(dxf,m,n); dyf=[1;-1];
DYF=zeropad(dyf,m,n); dxxf=[-1, 2, -1]; DXXF=zeropad(dxxf,m,n);
dyyf=[-1; 2; -1]; DYYF=zeropad(dyyf,m,n); dxyf=[-1 1;1 -1];
DXYF=zeropad(dxyf,m,n);

tic; xkl=y; for k=1:10

  xklold=xkl;
  Wa=W*a;

  for l=1:2

   weight_x=reshape(B(:,1),m,n);weight_x=weight_x.^(p/2-1);
   weight_y=reshape(B(:,2),m,n);weight_y=weight_y.^(p/2-1);
   weight_xx=reshape(B(:,3),m,n);weight_xx=weight_xx.^(p/2-1);
   weight_yy=reshape(B(:,4),m,n);weight_yy=weight_yy.^(p/2-1);
   weight_xy=reshape(B(:,5),m,n);weight_xy=weight_xy.^(p/2-1);

   H=zeropad(h,m,n);

   Ax=eta*conj(fft2(H)).*fft2(H);
```

```
Ax=Ax+alpha*p*conj(fft2(DXF)).*ifft2(fft2(fft2(weight_x)).*...
              fft2(fft2(DXF)))/(m*n);
Ax=Ax+alpha*p*conj(fft2(DYF)).*ifft2(fft2(fft2(weight_y)).*...
                  fft2(fft2(DYF)))/(m*n);
Ax=Ax+0.5*alpha*p*conj(fft2(DXXF)).*ifft2(fft2(fft2(weight_xx)).*...
              fft2(fft2(DXXF)))/(m*n);
Ax=Ax+0.5*alpha*p*conj(fft2(DYYF)).*ifft2(fft2(fft2(weight_yy)).*...
              fft2(fft2(DYYF)))/(m*n);
Ax=Ax+0.5*alpha*p*conj(fft2(DXYF)).*ifft2(fft2(fft2(weight_xy)).*...
              fft2(fft2(DXYF)))/(m*n);

bx=eta*conj(fft2(H)).*fft2(Wa);

xklf=bx./Ax;
xkl=real(ifft2(xklf));

xkl=reshape(xkl,m,n);

xklT=xkl;
xklT=fftshift(xklT);
Ahf=eta*conj(fft2(xklT)).*fft2(xklT)+gamma*conj(fft2(C)).*fft2(C);
bhf=eta*conj(fft2(xklT)).*fft2(Wa);
hf=bhf./Ahf;
h=real(ifft2(hf));
h=h(120:136,120:136);
figure(4)
subplot(1,2,1)
```

```
 mesh(h1)
 subplot(1,2,2)
 mesh(h);
 pause(0.1)

 dx=convolution(xkl,dxf,1);
 dy=convolution(xkl,dyf,1);
 dxx=convolution(xkl,dxxf,1);
 dyy=convolution(xkl,dyyf,1);
 dxy=convolution(xkl,dxyf,1);

 W1=max(abs(dx).^2,thr_e); B(:,1)=W1(:);
 W2=max(abs(dy).^2,thr_e); B(:,2)=W2(:);
 W3=max(abs(dxx).^2,thr_e);B(:,3)=W3(:);
 W4=max(abs(dyy).^2,thr_e);B(:,4)=W4(:);
 W5=max(abs(dxy).^2,thr_e);B(:,5)=W5(:);
end

norma=norm(xkl-xklold)/norm(xklold);

if norma<1e-2
   break
end

figure(3)
imagesc(xkl); colormap gray
pause(0.01)
```

```
  HX=convolution(xkl,h,1);


  for i=1:n
    ff = @(a) minimizar_a2(beta,eta,W,a,HX(:,i),y(:,i),n,w);
    a(:,i) = fminunc(ff,a(:,i));
  end


  eta=eta*theta;
end toc; disp(['d(xorig,xrec) = ', num2str(norm(f-xkl))]);
```

A.6 Matlab Code for the Method Using the Second EZW Idea Combined With Rotation-Invariance

```
clc, clear all, close all;
warning off;
% Creating the mage
[Wav,high,low,k]=wavelet_factory4('db1', 256,256/8);
W=Wav';
f = double(imread('cameraman.tif'));
[m,n]=size(f);
N=m*n;
figure(1)
pause(0.01)
imagesc(f), colormap gray;
h1=fspecial('gaussian',17,sqrt(5));
sigma=1e-3;
y=convolution(f,h1,1)+sigma*randn(m,n);
```

```
a=Wav*y;
figure(2)
imagesc(y), colormap gray;
pause(0.01)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
lag = fspecial('laplacian',0.0);

C=zeros(m,n);


primerIndiceX=floor((m-3)/2);
ultimoIndiceX=primerIndiceX+3;


primerIndiceY=floor((n-3)/2);
ultimoIndiceY=primerIndiceY+3;


C(primerIndiceX+1:ultimoIndiceX,primerIndiceY+1:ultimoIndiceY)=lag;


C=fftshift(C);
% w(1) is w0, and it should have the largest weight
% w(2) is w1, and it should a number between w0 and w2 (w0 > w1 > w2)
% w(3) is w2, and it should have the smallest weight
w = 1e-6*[10 5 1];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Using the program
% Estimating a new filter
h=fspecial('gaussian',17,sqrt(2));

alpha=1;
```

```
p=0.8;

B=rand(N,5);

eta=1042;

gamma=5e5;

exp_a=0.8;

thr_e=0.0001;

beta=sigma^2;

tau=0.125;

theta=1.3;


dxf=[1 -1]; DXF=zeropad(dxf,m,n);

dyf=[1;-1]; DYF=zeropad(dyf,m,n);

dxxf=[-1, 2, -1]; DXXF=zeropad(dxxf,m,n);

dyyf=[-1; 2; -1]; DYYF=zeropad(dyyf,m,n);

dxyf=[-1 1;1 -1]; DXYF=zeropad(dxyf,m,n);


tic;

xkl=y;

for k=1:10

  xklold=xkl;

  Wa=W*a;

  for l=1:2


    weight_x=reshape(B(:,1),m,n);weight_x=weight_x.^(p/2-1);

    weight_y=reshape(B(:,2),m,n);weight_y=weight_y.^(p/2-1);

    weight_xx=reshape(B(:,3),m,n);weight_xx=weight_xx.^(p/2-1);

    weight_yy=reshape(B(:,4),m,n);weight_yy=weight_yy.^(p/2-1);

    weight_xy=reshape(B(:,5),m,n);weight_xy=weight_xy.^(p/2-1);
```

```
H=zeropad(h,m,n);

Ax=eta*conj(fft2(H)).*fft2(H);

Ax=Ax+alpha*p*conj(fft2(DXF)).*ifft2(fft2(fft2(weight_x)).*...
          fft2(fft2(DXF)))/(m*n);
Ax=Ax+alpha*p*conj(fft2(DYF)).*ifft2(fft2(fft2(weight_y)).*...
          fft2(fft2(DYF)))/(m*n);
Ax=Ax+0.5*alpha*p*conj(fft2(DXXF)).*ifft2(fft2(fft2(weight_xx)).*...
          fft2(fft2(DXXF)))/(m*n);
Ax=Ax+0.5*alpha*p*conj(fft2(DYYF)).*ifft2(fft2(fft2(weight_yy)).*...
          fft2(fft2(DYYF)))/(m*n);
Ax=Ax+0.5*alpha*p*conj(fft2(DXYF)).*ifft2(fft2(fft2(weight_xy)).*...
          fft2(fft2(DXYF)))/(m*n);

bx=eta*conj(fft2(H)).*fft2(Wa);

xklf=bx./Ax;
xkl=real(ifft2(xklf));

xkl=reshape(xkl,m,n);

xklT=xkl;
xklT=fftshift(xklT);
Ahf=eta*conj(fft2(xklT)).*fft2(xklT)+gamma*conj(fft2(C)).*fft2(C);
bhf=eta*conj(fft2(xklT)).*fft2(Wa);
hf=bhf./Ahf;
```

```
 h=real(ifft2(hf));
 h=h(120:136,120:136);
 figure(4)
 subplot(1,2,1)
 mesh(h1)
 subplot(1,2,2)
 mesh(h);
 pause(0.1)

 dx=convolution(xkl,dxf,1);
 dy=convolution(xkl,dyf,1);
 dxx=convolution(xkl,dxxf,1);
 dyy=convolution(xkl,dyyf,1);
 dxy=convolution(xkl,dxyf,1);

 W1=.5*max(abs(dx).^2+abs(dy).^2,thr_e); B(:,1)=W1(:);
 W2=W1; B(:,2) = W2(:);
 W3=max(abs(dxx).^2,thr_e);B(:,3)=W3(:);
 W4=max(abs(dyy).^2,thr_e);B(:,4)=W4(:);
 W5=max(abs(dxy).^2,thr_e);B(:,5)=W5(:);
end

norma=norm(xkl-xklold)/norm(xklold);
if norma<1e-2
   break
end
figure(3)
imagesc(xkl); colormap gray
```

```
  pause(0.01)
  HX=convolution(xkl,h,1);


  for i=1:n
    ff = @(a) minimizar_a(beta,eta,W,a,HX(:,i),y(:,i),n,w);
    a(:,i) = fminunc(ff,a(:,i));


  end


  eta=eta*theta;
end
toc;
disp(['d(xorig,xrec) = ', num2str(norm(f-xkl))]);
```

Curriculum Vita

Fernando Cervantes was born in El Paso, Texas, the son of Elizabeth Duran and Dr. Jose Luis Cervantes Mendez. He graduated from Hanks High School in 2003. While in high school, he enrolled in advanced mathematical courses since he knew that he would like to do something related to math in the future. In Fall 2003, he enrolled at the University of Texas at El Paso. The first few years working towards his Bachelor's Degree in Electrical Engineering, he did not maintain a good grade point average since he struggled to adapt to the university. The last years at the university, however, his grades went up, and he even got funded to help him pay for his tuition. He graduated with his bachelor's degree in 2008 and enrolled in the Master of Science degree in Engineering at the University of Texas at El Paso. During that time, he concentrated in biomedical engineering courses, such as biomedical imaging, bioinformatics, image processing, and biomedical instrumentation design. He was also funded by the Texas Instruments Endowed Scholarship for several years, and maintained an excellent grade point average. At the end of his master's studies, he did a project involving image restoration. He received his Master's degree in December 2010, and entered the PhD program at the University of Texas at El Paso in January 2011.

Permanent Address:
11769 Tony Tejeda Dr.
El Paso, Texas, 79936
fcervantes@miners.utep.edu