2017-01-01

Compressive Vector Reconstruction: HypoThesis For Blind Image Deconvolution

Alonso Orea Amador
University of Texas at El Paso, aorea@miners.utep.edu

COMPRESSIVE VECTOR RECONSTRUCTION: HYPOTHESIS FOR BLIND IMAGE DECONVOLUTION

ALONSO OREA AMADOR

Master's Program in Electrical Engineering

APPROVED:

Bryan Usevitch, Ph.D., Chair

Sergio Cabrera, Ph.D.

Berenice Verdin, Ph.D.

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

Dedication

To my dear family and friends

COMPRESSIVE VECTOR RECONSTRUCTION: HYPOTHESIS FOR BLIND IMAGE DECONVOLUTION

by

ALONSO OREA AMADOR, B.S. in E.E.

THESIS

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

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

THE UNIVERSITY OF TEXAS AT EL PASO

May 2017

Acknowledgements

This work would not be possible without the continued support of my advisor Dr. Bryan Usevitch, whom I sincerely thank. His hard work as professor and mentor really inspired me to give my best. In addition, the initial part of the code presented here was possible thanks to Dr. Fernando Cervantes who generously shared his recreation code from which this present work grew.

Moreover, I would like to thank my professors who helped and tough me along the path to my undergraduate and graduate studies. Special thanks go to Dr. von Borries and Dr. Verdin who gave me my first formal research opportunity as an undergraduate student and showed me the best professional practices in research.

Thanks to the members of my committee, Dr. Cabrera and Dr. Verdin, for taking the time to read and evaluate this work. Their insights helped to increase the quality of this work.

Finally, this research was supported in part by the Texas Instruments Foundation Endowed Scholarship Program. My sincere appreciation and gratitude for their support. I hope they continue helping students to achieve their dreams in advanced degree.

Abstract

Alternative imaging devices propose to acquire and compress images simultaneously. These devices are based on the compressive sensing (CS) theory. A reduction in the measurement required for reconstruction without a post-compression sub-system allows imaging devices to become simpler, smaller, and cheaper. In this research, we propose a new algorithm to compress and reconstruct blurred images for CS imaging devices. Blur effect in images is common due to relative motion, lens, limited aperture dimensions, lack of focus, and/or atmospheric turbulence. Our intention is to compress a blurred image with CS techniques and then reconstruct a blur-free version using the proposed algorithm. To assess the performance of the proposed algorithm in comparison to other CS based compression schemes, we have used the Peak-Signal-to-Noise-Ratio (PSNR). Our algorithm is based on the previous work of compressive blind image deconvolution (BID) [1] and in a new way of organizing wavelet coefficients [2]. We can see an improvement up to 2 dBs in the PSNR for the two highest compression rates comparing the proposed algorithm with the one presented in [1].

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Motivation

Advances in technology have increased the use of digital imaging devices in many different technical areas. Traditional imaging devices create an image in two sequential phases. In the acquisition phase, the challenge is to acquire the highest possible spatial resolution. In general, imaging devices achieve this goal by using a series of 2D sensors array, which average the intensity of the light coming towards the lens of the device. The light is projected by the lens reaching the sensor. Then it is converted into a voltage or current value. This voltage or current value represents the intensity value of a dot in the final image. The higher the intensity value the brighter the dot; the lower the intensity value the darker the dot [3]. Frequently, increasing the spatial resolution of the image implies an increase of the number of sensors to increase the sampling rate. In terms of signal processing, the Nyquist theorem specifies that to accurately preserve the information contained in a signal the sampling rate must be at minimum twice the bandwidth of the signal of interest. In terms of imaging, a slow sampling rate produces a low-resolution image. The higher sampling rate, the higher the resolution of the image. Increasing the resolution of an imaging system translates into the increase on the number of sensors (dots) and thus the cost and complexity of the system. For example, a megapixel camera requires several millions of sensors to produce a megapixel image [4].

Storing the raw data generated by the sensors is impractical due to the memory capacity required and the redundancy of the information in the sampled image. Therefore, in the second phase of a standard imaging system, the image's data is post-processed for compression and storage. Here, the requirement is to use an efficient compression technique to preserve key information and reliably reproduce or reconstruct when required. Usually, the process requires

going to a transformation domain such as Fourier or Wavelet [5]. In standard imaging systems, inside the imaging device there is an embedded microprocessor which performs a discrete cosine (Fourier) transform or a discrete wavelet transform before then compressing using JPEG or JPEG2000 techniques, respectively. Both techniques discard a lot of small coefficient of the original transformation retaining only significant coefficient and thus reducing the data to be stored [6].

1.2 Problem Statement

A two-phase digital imaging process is extremely wasteful in time and energy for massive data acquisitions. For example, two very important requirements in applications such as satellites and remote sensing are energy efficiency and time acquisition [7]. Moreover, having two separate processes increases the complexity and the cost of imaging devices, especially in expensive systems such as X-rays, infrared, or millimeter wave cameras. The challenges increase as the resolution of the image is increased due to the direct translation into more sensors being required, more data being generated, and more post-processing power needed.

Alternative imaging devices have been proposed to acquire and compress images simultaneously [5], [8]. These devices are based on the Compressive Sensing (CS) theory. CS allows an image to be acquired with far fewer samples than required by standard techniques based on the Nyquist theorem. Thus, the post-processing step to compress can be skipped, saving power consumption, simplifying the system's size, and reducing costs.

In this thesis work, we propose a new algorithm to compress and reconstruct blurred images for CS imaging devices. Blur degradation in images is common due to relative motion, lens, limited aperture dimensions, lack of focus, and/or atmospheric turbulence. Our intention is to compress a blurred image with CS techniques and then reconstruct a blur-free version using the proposed

algorithm. To assess the performance of the proposed algorithm in comparison with other CS based compression schemes, we have used the Peak-Signal-to-Noise-Ratio (PSNR). Our algorithm is based on the previous work of compressive blind image deconvolution (BID) [9], and in a new way of organizing wavelet coefficients [2].

Chapter 2: Background

This chapter explores the fundamental concepts on which the algorithm presented in this work is based. We review the Compressive Sensing (CS) theory and its requirements for practical application. Next, a short overview of wavelets and their importance in CS applications is presented. Finally, we present the two main research papers where the idea for the algorithm presented here was born.

2.1 Compressive Sensing

Traditional Nyquist Theory states that to preserve the information in a signal of interest, it must be sampled at a rate of at least twice its bandwidth. Usually, this process generates vast amounts of sample coefficients from which many are redundant information. In contrast, CS does not rely on the bandwidth. Instead, samples are obtained by an inner product of the signal with a measurement matrix. This happens in real time in the acquisition phase [8] making the post-compression stage unnecessary.

CS imaging device systems generate important advantages. Having a single-stage imaging system reduces the acquisition time and the power consumption due to the unrequired post-processing compression stage. In addition, the number of sensors required by the digital image device to generate a high-resolution image decreases. All previous advantages translate into a significant reduction of the cost and complexity of imaging device systems making them smaller, faster, and cheaper. To understand how this is possible, it is necessary to explore the theory behind CS.

2.1.1 General Theory

The general idea of CS is as follows. Let **x** be the signal of interest with dimension $N \times 1$ or $\mathbf{x} \in \mathbb{R}^N$. The sampling process is achieved by multiplying **x** by a measurement (or sampling) matrix **Φ** as

$$\mathbf{y} = \mathbf{\Phi x}, \tag{1}$$

where **Φ** is dimension $M \times N$ in $R^{M \times N}$ and **y** is the observation vector or compressed measurement vector with dimension $M \times 1$ in $R^{M \times 1}$. The compression is possible because $M \ll N$. **Figure 2.1** shows a visual representation of CS general ideal.

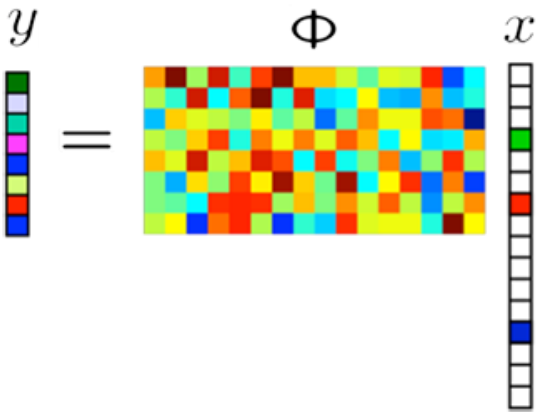


Figure 2.1: Visual illustration of compressive sensing idea. Image retrieved from [10]

In imaging systems, **x** represents the coefficients of the image in lexicographical order, and **y** is the observations vector – the compressed version of **x**. From observations vector **y** the original image is intended to be reconstructed. The reconstruction requirements are presented below.

2.1.2 Requirements for Compressive Sensing Reconstruction

CS theory states that to recover the original signal $\mathbf{x}$ from $\mathbf{y}$, two conditions must be satisfied. First, the original vector signal $\mathbf{x}$ must be sparse or approximate sparse in the time (spatial) domain or in some transformation domain - Fourier or Wavelet [4]. That is, the signal of interest must have mostly zero coefficient values in natural form or at least in some transformation domain. Most natural images are not sparse in natural form. Thus, a transformation domain is required.

The original image $\mathbf{x}$ can be obtained from the transformation domain by using the inverse transformation matrix $\boldsymbol{\Psi}$. That is,

$$\mathbf{x} = \boldsymbol{\Psi}\mathbf{s} = \sum_{j=1}^{N} \boldsymbol{\Psi}_j \mathbf{s}_j, \tag{2}$$

where $\mathbf{s} = \{\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_N\}$ is the transformation of $\mathbf{x}$ with $N \times 1$ coefficient vector with $\mathbf{s}_j = \langle \boldsymbol{\Psi}_j', \boldsymbol{x} \rangle$ and $\boldsymbol{\Psi}_{N\times N} = \{\boldsymbol{\Psi}_1, \boldsymbol{\Psi}_2, \dots, \boldsymbol{\Psi}_N\}$ being $\boldsymbol{\Psi}_j$ the j^{th} column vector of basis matrix $\boldsymbol{\Psi}$ [2]. Notice that the transformation matrix $\boldsymbol{\Psi}'$ is orthogonal. That is, $\boldsymbol{\Psi}\times\boldsymbol{\Psi}' = \boldsymbol{\Psi}'\times\boldsymbol{\Psi} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

In this way, the CS observation $\mathbf{y}$ are obtained from $\boldsymbol{s}$ as follows

$$\mathbf{y} = \boldsymbol{\Phi}\boldsymbol{\Psi}\mathbf{s} = \mathbf{A}\mathbf{s} \tag{3}$$

where $\boldsymbol{\Phi}$ represents the measurement matrix, $\boldsymbol{\Psi}$ represents the inverse transformation matrix, $\boldsymbol{A}$ is the $M\times N$ sensing matrix, and, $\mathbf{s}$ is K-sparse (that is, it has K nonzero values). Figure 2.2 displays a visual representation of CS with a transformation domain.

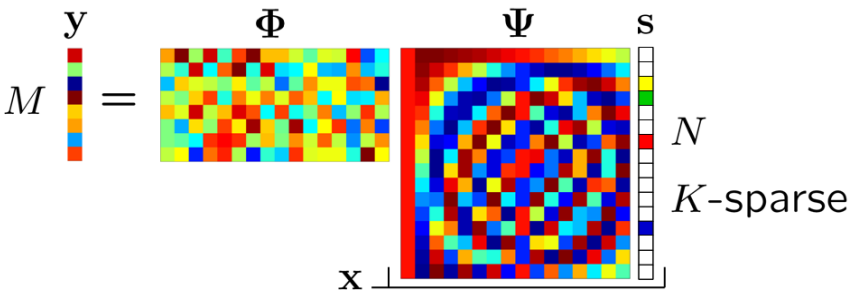


Figure 2.2: Visual representation of general idea of CS with the signal move to a Transformation domain. Image retrieved from [11]

The second condition required by CS to reliably reconstruct the original signal is the incoherence between the sensing matrix **Φ** and the transformation matrix **Ψ**. Incoherence determines the minimum number of samples K required for reconstruction. Incoherence refers to the fact that there is no correlation between the rows of **Φ** and the columns of **Ψ**.The greater the incoherence is, the smaller the number of measurements is needed for reconstruction. To test for incoherence, $\mathbf{A} = \mathbf{\Phi\Psi}$ must satisfy the Restricted Isometry Property (RIP). The matrix $\boldsymbol{A}$ would satisfy the RIP of order K and constant $\delta_S \in (0,1)$ if

$$(1 - \delta_k)\left||x|\right|_2^2 \leq ||\boldsymbol{A}_T\mathbf{x}|| \leq (1 + \delta_k)\left||x|\right|_2^2 \tag{4}$$

Here $\boldsymbol{A}_T, \subset 1, \ldots, N$ denotes the $K \times |T|$ submatrix obtained by extracting the columns of **Φ** corresponding to the indexes in T, where T is an index set of columns from the measurement matrix $\boldsymbol{A}$ [12].

For practical purposes, the simplest way to satisfy RIP is by choosing the sampling matrix, **Φ,** randomly. Frequently, the measurement matrix is created from random matrices, because they are incoherent to most sparse transforms. Thus, the RIP conditions hold. Usually, the random values for **Φ** comes from (i.i.d) Gaussian or Bernoulli random variable.

2.1.3 Compressive Sensing Reconstruction

When the signal is sparse, the reconstruction step for CS requires searching for sparse solution $\tilde{\boldsymbol{s}}$ as an under-determined system of linear equations, M$\ll N$ [13]. That is, finding a solution with the fewest non-zero entries or lowest ℓ_0. The recovery using ℓ_0-minimization can be formulated as

$$\tilde{\mathbf{s}} = arg\min_{\mathbf{s}} \left|\left|\mathbf{s}\right|\right|_0 \text{subject to: } \mathbf{y} = \mathbf{A}\mathbf{s} \tag{5}$$

where $\left|\left|\boldsymbol{s}\right|\right|_0$ corresponds to non-zero entries in vector $\boldsymbol{s}$. The recovery problem is a non-deterministic polynomial-time hard (NP-hard). This kind of problem has a non-convex nature of ℓ_0-minimization. Furthermore, its nature is ill-conditioned and difficult to solve, since it requires an exhaustive search to find the most sparse solution [14]. Nonetheless, the unique sparse solution can be found exactly using ℓ_1-minimization [15]. The solution to the reformulated reconstruction problem as a convex problem can be found using linear programing [16].

$$\tilde{\mathbf{s}} = arg\min_{\mathbf{s}} ||\mathbf{s}||_1 \text{subject to: } \mathbf{y} = \mathbf{A}\mathbf{s} \tag{6}$$

where

$$||\mathbf{s}||_1 = \sum_{j=1}^{N} |\boldsymbol{s}_j| \tag{7}$$

The above being the simplest expression. However, finding the solution of CS algorithms is still a very active research area. Different methods of solution include proposing different ways of stating the original problem. In this work, we will use the following representation as in [1].

$$\min_{\boldsymbol{s}} \left|\left|\mathbf{y} - \boldsymbol{\Phi}\boldsymbol{\Psi}\mathbf{s}\right|\right|_2^{\,2} + \tau\left|\left|\mathbf{s}\right|\right|_1 \ , \tag{8}$$

where $||\cdot||_2$ denotes the Euclidean norm, $||\cdot||_1$ denotes the ℓ_1-norm, and τ is a non-negative regularization parameter.

2.2 Blur Distortion

Blur effects in images and signals are common in many applications such as medical, optical, astronomical, and physical. Blur is often the consequence of lens, limited aperture dimensions, and lack of focus, motion, insufficient light, atmospheric turbulences, or combinations of the above. Removing the effect of blur is an inverse problem and is usually called deblurring or deconvolution.

2.2.1 The Blur Problem

The blur problem can be stated as follows

$$\mathbf{y_h} = \mathbf{Hx} + \mathbf{n}, \tag{9}$$

where $\mathbf{x}$ is the original discrete signal of interest size $N\times1$; $\mathbf{H}$ is an $N\times N$ low-pass linear blurring operator created from a point spread function (PSF), representing the blur from lens, atmosphere, motion, etc.; $\boldsymbol{n}$ is the measurement error or noise; and $\boldsymbol{y}$ is the observed signal.

The deblurring techniques goal is to obtain $\mathbf{x}$ from $\mathbf{y}$. An initial approach focus on finding the inverse of the low-pass filter $\mathbf{H}^{-1}$ and solve for the appreciation $\tilde{\mathbf{x}}$ as

$$\tilde{\mathbf{x}} = \mathbf{H}^{-1}\mathbf{y_h} = \mathbf{H}^{-1}\mathbf{x} + \mathbf{H}^{-1}\mathbf{n} \tag{10}$$

Ideally, $\tilde{\mathbf{x}} = \mathbf{x}$. However, the deblurring problem is a nonlinear ill-posed problem. As it is shown, the noise is affected by $\mathbf{H}^{-1}$, as well causing the error to increase degrading the quality of the reconstruction. Therefore, a second approach is to find a new inverse operator $\mathbf{P}^{-1}$ such that

$$\tilde{\mathbf{x}} = \mathbf{P}^{-1}\mathbf{y_h} = \mathbf{P}^{-1}\mathbf{Hx} + \mathbf{P}^{-1}\mathbf{n} \tag{11}$$

with the constraint that $\mathbf{P}^{-1}\mathbf{Hx} \approx \mathbf{x}$ and $\mathbf{P}^{-1}\mathbf{n} \approx 0$.

Different methods have been proposed to find $\mathbf{P}^{-1}$. A complete literary review about the subject can be found in [17].

2.2.2 Compressive Sensing Deconvolution

One of the major objectives in this work is not only to find the best $\mathbf{P}^{-1}$ such that it satisfies the constraints mentioned above, but to find it when the measurements are incomplete. That is, reconstruct a blur-free image which has been previously compressed using CS.

When CS is implemented to a blurred signal, the CS observation now becomes

$$\mathbf{y} = \mathbf{\Phi H x} + \mathbf{n} \tag{12}$$

An interesting consequence of the blur operator **H** is that it reduces the sparsity of the signal in the time or spatial domain. However, in the transformation domain, the number of nonzero coefficients reduces, and the sparsification of the signal increases further. Such increase in the sparsification allows an easier reconstruction of its blurred version. Of course, there is a limit in the blur degradation allowed before it becomes impossible to recover the original signal. The author in [18] proposes a lower boundary marking the limit before reconstruction become unachievable.

2.2.3 Compressive Sensing Deconvolution Problem Statement

Applying CS to a blurred image, **Hx,** with transform representation **a,** redefine **y** as

$$\mathbf{y} = \mathbf{\Phi \Psi a} + \mathbf{n}, \tag{13}$$

where $\mathbf{Hx} = \mathbf{\Psi a}$. The sparse signal $\mathbf{a}$ represents the transformation coefficients corresponding to N basis vectors which span the column space of the $N \times N$ matrix $\mathbf{\Psi}'$, i.e. $\mathbf{a} = \mathbf{\Psi}'\mathbf{Hx}$. Letting $\mathbf{\Psi} = \boldsymbol{W}$, to emphasize the presence of blur in the signal of intertest and then

$$\mathbf{y} = \mathbf{\Phi W a} + \mathbf{n}, \tag{14}$$

Reconstruction of $\mathbf{x}$ from $\mathbf{y}$ problem is called compressive sensing (CS) deconvolution [12]. Deconvolution for compressed measurement is required when the blurred images prevent the viewing of the image or further the processing. A solution for CS deconvolution is presented in [9]. In Section 2.4, the solution is explored further.

2.3 Wavelet Transform

A mentioned before, the transformation domain can sparsify a non-sparse signal or at least make it approximately sparse. Thus, allowing significant compression by CS. In this way, many non-sparse natural images can be recovered from CS observation.

In this thesis, the transform domain of use is wavelets. The wavelet transform is ideal for image processing, because it allows one to keep track of the changes in frequency in the images. In contrast, the Fourier Transform (FT) uses sinusoids and cannot keep track of the change in frequency with respect to the location. The FT requires dividing the image into small pieces to have a more accurate mapping of the change in frequency to the original signal. This introduces blocking artifacts when each part is put back together to form the reconstructed image, while the wavelet transform can work with the whole image at once.

The single-stage two-dimensional Discrete Wavelet Transform (DWT) for an image size $M \times N$ is calculated from the one-dimensional DWT. The process is as follows: The rows of the

image are transformed using two one-dimensional filters - a low-pass filter and high-pass filter. Then the resulting output is down sampled by a factor of two. This produces two $M \times N/2$ down-sampled filtered images, one obtained from the low-pass filter, and the other obtained from the high-pass filter. Next, the process is repeated for both down-sampled filtered images, but this time columnwise. Finally, the resulting output is decimated, resulting in the single-stage two-dimensional DWT with four dimensions: LL (low-pass low-pass), LH (low-pass high-pass), HL (high-pass low-pass), and HH (high-pass high-pass). The process can be extended to higher levels by repeating the filtering for LL sub-band, until achieving the required levels. That is, the LL sub-band is decomposed into four more levels. In general, the wavelet transform results from the filtering of the signals by high-pass and low-pass filters in sequence time resulting in an approximation signal and several details signals. The image quality greatly depends upon the low-frequency components. The LL sub-band carries most of the image energy.

To illustrate sparsification you can observe Figure 2.3. The Figure contains the matlab standard cameraman on the top. On the bottom, it is a histogram of its the intensity values. As you can observe, the image is not sparse because its intensity coefficients are all over the range 0-255.

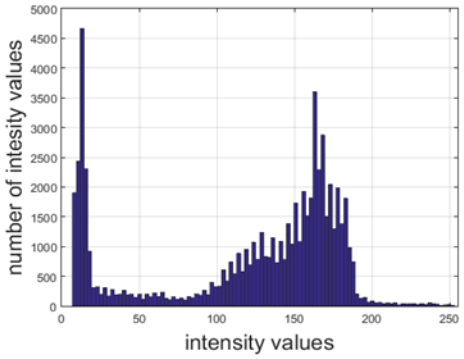


Figure 2.3: Cameraman and histogram of its intensity values

On the other hand, Figure 2.4 shows the wavelet transform of the Cameraman image with 2 levels of decomposition. We can observe how the wavelet coefficients follow an approximate sparse histogram having mostly zero values.

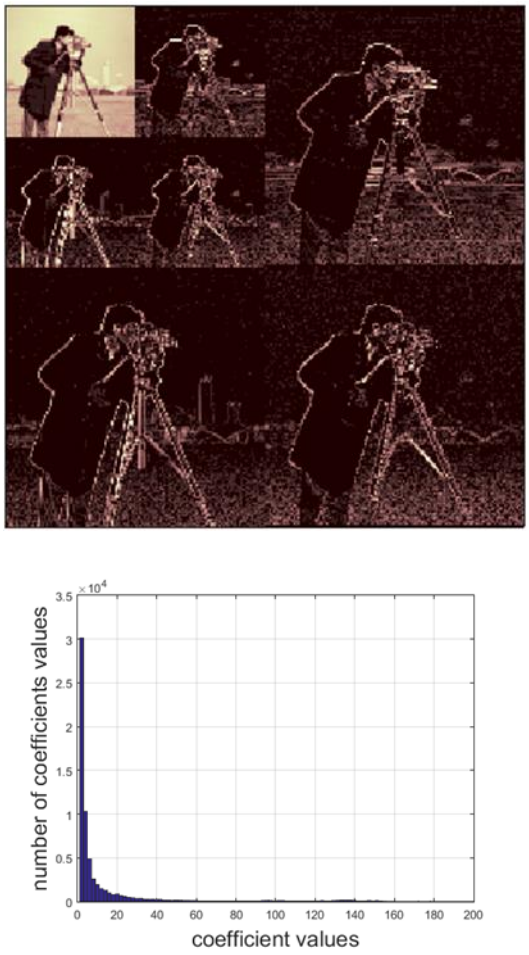


Figure 2.4: Sparsification of Cameraman using wavelets as Transformation Domain.

2.4 Related Work

2.4.1 Compressive Blind Image Deconvolution

Authors in [1] present a solution for the image deconvolution equation (14). They present a new algorithm that reconstructs the original image and removes the blur at the same time. That is, the algorithm presented can reconstruct a blur-free image from CS measurements obtained from the blurred image. In this paper, the algorithm is tested for two major cases, blind and nonblind deconvolution. In the first case, the Point Spread Function (PSF) representing the blur is assumed

to be unknown and the problem becomes a blind image deconvolution. In this case, the algorithm returns an estimate of the original blur-free image **x** and an estimate of the PSF **h** which is the blur filter approximation. In a second experiment, the PSF is assumed to be known and the problem becomes a nonblind deconvolution problem. Thus, their algorithm attempts to reconstruct only the original blur-free image **x** using an assumed known PSF **h**.

In the experiment, they test different compression ratios, blur, and noise degradations. They use the Peak-Signal-to-Noise-Ration (PSNR) between the reconstructed image $\hat{\mathbf{x}}$ and the original blur-free image **x** to assess the quality of the reconstruction. Not surprisingly, the nonblind deconvolution problem shows better results since the PSF is known. However, in some cases, the reconstruction for the nonblind and blind deconvolution shows very close PSNR. In such cases, the compression ratio was high using only 20% - 40% of the original measurements to reconstruct **x**.

Nonetheless, the algorithm proposed by the authors shows better performance in most cases when compared with most state-of-art algorithms such as l1-ls [19], GPSR [20], NESTA [21], YALL1 [22], and CoSaMP [23]. In the case of this algorithm, they only solve the nonblind case limiting their use.

In fundamental sense, our work intends to improve on the algorithm reconstruction by following the ideas presented in the following reference.

2.4.2 Compressive Sensing and Wavelet Coefficient as Sparse Vectors

The work in [2] presents an ingenious idea in which the coefficients in the transformation domain are rearranged in vectors. Because of the rearrangement specifications, the vectors are highly sparse. The vector arrangement takes advantage of the characteristics of the transformation domain, the wavelet transform, which facilitate the sparsification of the vectors. In addition, the

CS reconstruction problem becomes a simple 1D vector reconstruction, because a single vector is reconstructed once at a time. To maximize the compression and the reconstruction, authors in [2] normalize the measurement matrix with the weighted average Root Mean Squared energies of the different wavelet sub-bands. The results present a significant improvement over existing CS-based compression methods such as RWS [24], CRP+AS [25], WBCT [26], and others [2].

To illustrate the vector arrangement let us consider the following example. Figure 2.5 shows the discrete wavelet transform 2-levels deep of the standard Matalab image Cameraman. Remember that the quality of the reconstruction depends greatly on the LL sub-band, since this band is the one that has the most energy of the discrete wavelet transform. The remainder of the Wavelet sub-bands contain low energy coefficients. That is, their coefficient values are mostly zero or close to zero. Then the idea is to create highly sparse vectors by taking only one coefficient from the LL sub-band and several coefficients from the other bands to form the vectors.

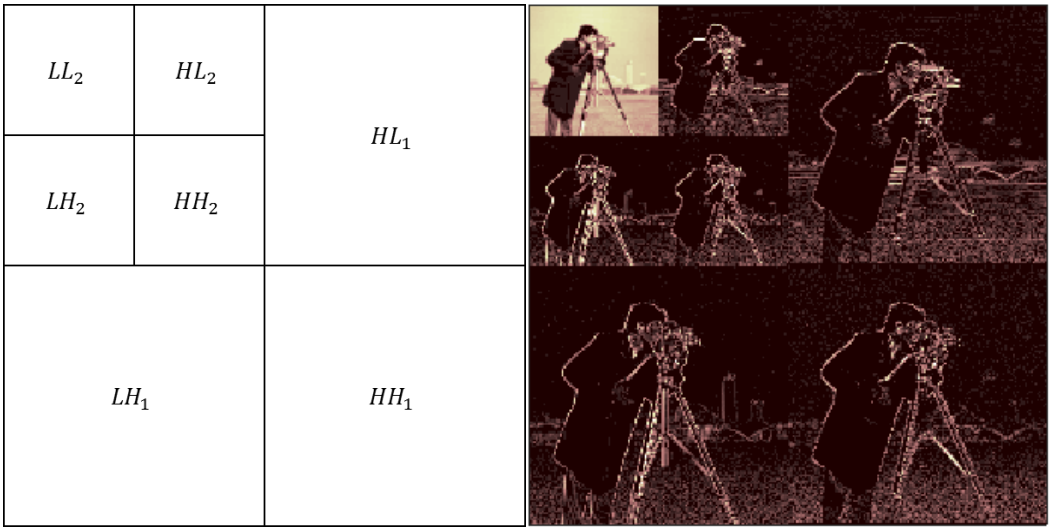


Figure 2.5: Cameraman 2-Levels Deep Wavelet Transform

As shown in Figure 2.6, the idea of [2] is to divide the discrete wavelet trasnform in sub-blocks of the same size. Then from each block, one wavlet coefficent is taken to form a single

vector. The total number of vectors is determined by the size of the origianl image and the size of the smallest sub-bands. In this example, we use 2-Level wavelet transfrom resulting in the smallest sub-band, the LL_2, HL_2, LH_2, and HH_2. Hower, the wavelet level can be further.

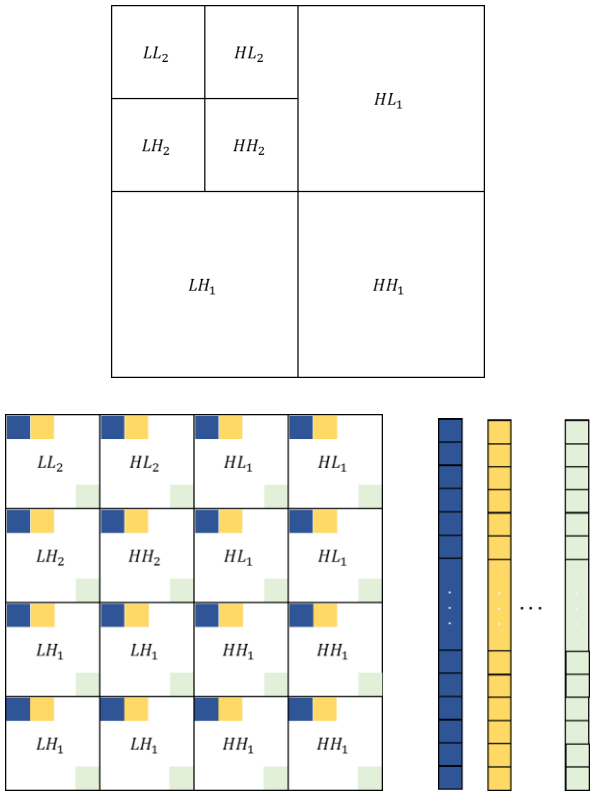


Figure 2.6: Wavelet transform rearranged in vector form by [2] method. Original wavelet transform sub-bands (top). Sub-block distribution and vector creation (Bottom).

Once the sparse vectors are created, the measurement matrix is weighted to compresses them. Authors in [2] propose to normalize the measurement matrix $\boldsymbol{\Phi}$ with the weighted average Root Mean Squared (RMS) energies of the different wavelet sub-bands as follows. Let $\boldsymbol{v_j}$ be the sparse vectors obtained from the wavelet transform for $j = 1,2, \ldots q$, where q represents the total number of sparse vectors and $\boldsymbol{v_j}$ is in $\mathbb{R}^B$ or length B. Then we have

$$\mathbf{E}_{\mathrm{i}} = \sqrt{\sum_{j=1}^{q} |\boldsymbol{v}_j^i|^2} \quad \text{for } i = 1,2,\dots,B \tag{15}$$

Having all vectors next to each other columnwise, the result of (15) returns **E**, the RMS energy of the sparse vectors row-wise. That is, E_1 is the RMS energy of all vectors entry one or $\boldsymbol{v_1} = v_{1,}^1, v_1^2, \dots, v_{1,}^q$; E_2 is the RMS energy of all vectors entry two or $\boldsymbol{v_2} = v_{2,}^1, v_2^2, \dots, v_{2,}^q$, and so on. Then the measurement matrix $\boldsymbol{\Phi}$ is normalized by the weighting matrix $\boldsymbol{\Omega} = [\boldsymbol{\omega}; \boldsymbol{\omega}; \dots, \boldsymbol{\omega}]$, $\boldsymbol{\omega} = [E_1, E_2, .. E_B]$ as follows

$$\boldsymbol{\Phi}_E = orth((\boldsymbol{\Phi}.\times\boldsymbol{\Omega})^T)^T, \tag{16}$$

where (.) represents entry-wise multiplication. According to [2] to satisfy the RIP conditions, the rows of the sensing matrix $\boldsymbol{\Phi}_E$ are normalized.

From these two works, our intention here is to answer the question: Can we improve the algorithm presented on [1] using the suggested rearrangement vector form of the Wavelet coefficient presented in [2]? As we will present in the following sections, the answer to this question is not a clear 'yes' or 'no'. We were able to make improvements, but the improvements were not uniform to all compression ratios.

Chapter 3: Experimentation

The major difference between the blind compressive sensing (CS) algorithm in [1] and the proposition of sparse vectors from the wavelet coefficients in [2] is the reconstruction of the vectors. Being as this is our first experimentation to test our hypothesis, our main objective is to combine the two ideas while trying to keep most of the original algorithms. Therefore, it is not our main priority to reduce the complexity of the proposed algorithm. We first want to test if the combinatorial hypothesis presents an improvement over previous works.

To have a comparison base, we tested the algorithm presented here with parameters similar as in [1]. The parameters are two series of experimentations using non-blind and blind reconstructions. The images used are the Matlab standard Lena and Cameraman. For blur degradation, the PSF is obtained from the Gaussian smoothing kernel filter with three degradation levels: variances 3, 5, and 9, respectively; the CS vectors are degraded with two different noise levels: 30dB and 40dB; and the wavelet transform is 3-levels deep.

In contrast with [1] , the wavelet filters used here are the Coiflets 5 (coif5). The specific coif5 are the coefficients that perform the best when vector reconstruction is involved [2].

3.1 Sparse Vector Creation

To create the sparse vectors, we start by obtaining the 3-level discrete wavelet transform of each blurred image. Figure 3.1 displays the two experimental images degraded with the Gaussian blur of variance 5 and their respective wavelet transforms as an example. We can see some circular shift in the wavelet transform. This is the result of using periodic padding when the wavelet transform is obtained. The reason to use periodic padding is to keep the same size as the original image. Otherwise, the wavelet transform results in a bigger image than original. Changing the size of the wavelet transform with respect the original image caused difficulties in the

application of sparse vectors construction using the *[2]* method. A series of experiments proves that the circular shifting does not affect the reconstruction of the original images. In fact, correcting the shifting in the wavelet transform inserts artifacts in the final reconstruction. In addition, we can observe in Figure 3.1 that the blur effect smooths the images causing more wavelet coefficients to become zero. Such smoothness in the image increases the sparsification of the wavelet transform allowing further sparsification of the vector when they are created, and facilitating the eventual reconstruction after CS is implemented.

Figure 3.1: Three-level wavelet transform of the two experimentation images Cameraman, and Lena. Both images were degraded with Gaussian blur of variance 5. The wavelet transform was obtained using filters coif5 and periodic-padding to keep original size.

The number of sparse vectors is determined by the image size and the depth of the wavelet transform. Having an image with size mxn and wavelet transform level L, the total number of sparse vectors is $\left(\frac{\mathrm{m}}{2^{\mathrm{L}}} \times \frac{\mathrm{n}}{2^{\mathrm{L}}}\right) = p$. Our experiments use a 3- level wavelet transform, and the image size is 256×265 resulting in 1024 vectors.

Next, the wavelet transform is divided into blocks of equal size (Figure 3.2). From each block a coefficient is taken to form a sparse vector. The size of each block is the same as the smallest Wavelet sub-bands. The vector's length is equal to the number of sub-blocks. [2] presents the following equation to calculate the number of blocks

$$\#\ \text{of Blocks} = 1 + 3\sum_{i=1}^{L} 4^{i-1} \tag{17}$$

Let $\boldsymbol{v}_j$ be the sparse set of vectors from a given wavelet transform, and we have

$$\boldsymbol{v}_j = \{\boldsymbol{v}_1^b, \boldsymbol{v}_2^b \ldots \boldsymbol{v}_j^b\} \tag{18}$$

where $\boldsymbol{v}_j^b$ represent a single sparse vector with length b. The length is emphasized with constant b because can change if the image size or the wavelet depth change.

In this work, the number of blocks is equal to 64. Thus, the length of each sparse vector is 64 as well. We can see an example of the block divisions in Figure 3.2. The Figure displays the blurred Cameraman image of the wavelet transform (left); the wavelet transform sub-bands divisions (center); and the block division of same size as the smallest wavelet sub-bands (right). As mentioned, before a single coefficient is taken from each sub-band to form a sparse vector. Thus, only one coefficient from the sub-band & -block containing the highest energy (sub-band LLL_3) goes to each vector. The rest of the coefficients are obtained from sub-sequential blocks with lower energy. Many of them have zero values. In this way, the sparsity of the vectors is maximized.

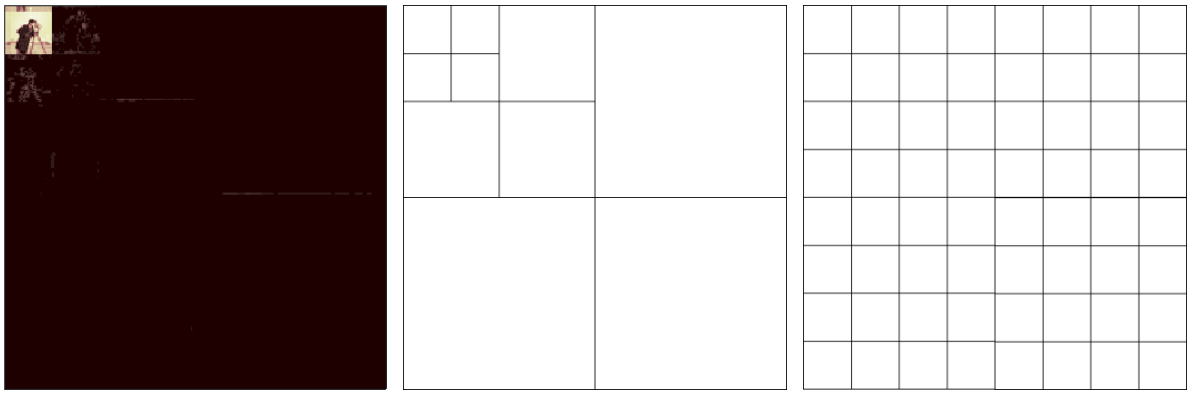


Figure 3.2: Sub-blocks selection to create sparse vector, each vector has a coefficient per block

Once the sparse vectors are formed, the CS observations **y** are obtained by simple

$$\mathbf{y} = \mathbf{\Phi}_{\mathrm{E}} \boldsymbol{v_j}, \tag{19}$$

where $\mathbf{\Phi}_E$ is the weighted energy measurement matrix obtained by equation (16).

We tested over different compression ratio ranging from .2 to 1, using .2 step increments. That is, we use CS to make reconstructions of the vectors from only 20%, 40%, ..., 100% of the original vectors length.

3.2 Blur Reduction and Vector Reconstruction

The reconstruction of the vectors should take in consideration the removal of the blur. To achieve the reconstruction and the blur reduction of the image, we followed the techniques presented in [1], and adapted them for vector reconstruction. Let us call their original algorithm in [1] Blind Image Deconvolution (BID) algorithm. There is not Matlab code provided by the authors. However, we use the recreation of Dr. Fernando Cervantes in [27] to have an initial starting point.

The BID algorithm starts by estimating the values of **x** and **h** from CS observations made to the wavelet transform of the blurred image. In our case, we first obtain the initial estimate of **a**, the wavelet transform of **Hx** (the original blurred image), from the CS vector observations **y**. The

initial estimate of the vector set $\widehat{\boldsymbol{v}}_{\boldsymbol{j}}$ is obtained by multiplying the transpose normalized measurement matrix $\boldsymbol{\Phi}_E{}'$ as follows

$$\widehat{\boldsymbol{v}}_j = \boldsymbol{\Phi}_E{}'\boldsymbol{y}\text{ , for } j = 1, 2, \ldots, 1024 \tag{20}$$

By rearranging $\widehat{\boldsymbol{v}}_j$ from vector form to wavelet transform, we can obtain an initial estimate of **a,** the wavelet transform of **Hx**. Then by simply obtaining the inverse wavelet transform, we obtain an estimation of the blurred image **Hx** . This initial estimate is used to have a first estimation of the reconstructed blur-free image **x** and the PSF filter **h** following the BID algorithm steps. That is, we use the equations (21) and (22) below

$$\mathbf{x}^{k,l} = [\eta^k(\mathbf{H}^{k,l})^t(\mathbf{H}^{k,l}) + \alpha p \sum_{\mathrm{d}} 2^{1-\mathrm{o}(d)}(\boldsymbol{\Delta}^d)^t\mathbf{B}_d^k(\boldsymbol{\Delta}^d)]^{-\mathbf{1}} \times \eta^k(\mathbf{H}^{k,l})^t\mathbf{W}\mathbf{a}^k \tag{21}$$

where, k and l represent internal iteration counters; η^k,α, p, and $o(d)$ are constants; $\mathbf{H}^{k,l}$ is the convolutional matrix of **h** for iteration k, l; $\mathbf{B}_d^k$ is a diagonal matrix with entries $\mathbf{B}_d^k(i,i) = \left(\boldsymbol{v}_{d,i}^{k.l}\right)^{\frac{p}{2}-1}$; $\boldsymbol{v}_{d,i}^{k.l} = [\,\boldsymbol{\Delta}_i^d(\mathbf{x}^{k,l})]^{\mathbf{2}}$; $\boldsymbol{\Delta}_i^d(\cdot)$ is the convolutional matrix of the difference operator, $d \in D = \{h, v, hh, vv, hv\}$, and $\boldsymbol{\Delta}_i^h(\mathbf{x})$, $\boldsymbol{\Delta}_i^v(\mathbf{x})$, $\boldsymbol{\Delta}_i^{hh}(\mathbf{x})$ ... correspond to the horizontal and vertical first- and second-order difference at pixel i, respectively. All previous are just parameters used to estimate **x**. Further details can be found in [1], [28]**,** and [29].

In similar fashion the estimation of PSF **h** is done using

$$\mathbf{h}^{k,l} = [\eta^k(\mathbf{X}^{k,l})^t(\mathbf{X}^{k,l}) + \gamma\mathbf{C}^t\mathbf{C}]^{-\mathbf{1}} \times \eta^k(\mathbf{X}^{k,l})^t\mathbf{W}\mathbf{a}^k \tag{22}$$

where η^k, and γ are constants; $\mathbf{X}^{k,l}$ is the convolutional matrix of the estimated image $\mathbf{x}^{k,l}$, and $\mathbf{C}$ is a regularization parameter [1], [30]

In the Matlab implementation, we decided to use the Fourier transform to make the convolutional matrices $\mathbf{H}, \mathbf{X}, \mathbf{C}$, and $\boldsymbol{\Delta}^d$ a simple multiplication operation. For the transpose version of these convolutional matrices we used the conjugate Fourier transform. For example,

$\mathbf{H}^t = conj(\boldsymbol{H}_{fft})$, where $\boldsymbol{H}_{fft}$ is the Fourier transform of $\mathbf{h}$ with size $M \times N = 256 \times 256$. Then the operation in the special domain of the convolutional matrices $\mathbf{H}^t\mathbf{H}$ can be substituted by $conj(\boldsymbol{H}_{fft})\boldsymbol{H}_{fft}$ in terms of the Fourier transform.

With $\mathbf{x}^k$and $\mathbf{h}^k$ we obtain our first estimation of the blurred image $\mathbf{H}^k\mathbf{x}^k$. The next step in the BID algorithm is the reconstructions of the CS measurements by equation (23)

$$\mathbf{a}^{k+1} = \arg\min_{\mathbf{a}} \frac{\beta}{2}\|\mathbf{y} - \mathbf{\Phi W a}\|^2 + \frac{\eta^k}{2}\|\mathbf{Wa} - \mathbf{H}^k\mathbf{x}^k\|^{\mathbf{2}} + \tau\|\mathbf{a}\|_{\mathbf{1}} \tag{23}$$

Here we insert our modification to reconstruct vectors instead as

$$\widehat{\boldsymbol{\nu}}_j^{k+1} = \arg\min_{\mathbf{a}} \frac{\beta}{2}\|\boldsymbol{y_j} - \mathbf{\Phi}\boldsymbol{\nu}_j\|^2 + \frac{\eta^k}{2}\left\|\boldsymbol{\nu_j} - \boldsymbol{\nu}_{\boldsymbol{j}}^{\mathbf{H}^k\mathbf{x}^k}\right\|^{\mathbf{2}} + \tau\|\boldsymbol{\nu_j}\|_{\mathbf{1}} \tag{24}$$

where $\boldsymbol{\nu}_{\boldsymbol{j}}^{\boldsymbol{Hx}}$ are the sparse vectors obtained from reconstruction $\mathbf{H}^k\mathbf{x}^k$. In this way, the reconstruction uses $\boldsymbol{\nu}_{\boldsymbol{j}}^{\boldsymbol{Hx}}$ as the blur removal parameter. We draw this possibility following the ideas presented in the BID deconvolution.

Equation (24) is simplified as

$$\widehat{\boldsymbol{\nu}}_j^{k+1} = \arg\min_{\mathbf{a}}\|\mathbf{y}' - \boldsymbol{\Phi}_E{}'\boldsymbol{\nu}_j\|^2 + \tau\|\boldsymbol{\nu}_j\|_{\mathbf{1}} \tag{25}$$

Where $\mathbf{y}' = \begin{bmatrix} \sqrt{\frac{\beta}{2}}\mathbf{y} \\ \sqrt{\frac{\eta^k}{2}}\boldsymbol{\nu}_j^{\mathbf{H}^k\mathbf{x}^k} \end{bmatrix}$ and $\mathbf{\Phi}' = \begin{bmatrix} \sqrt{\frac{\beta}{2}}\boldsymbol{\Phi}_E \\ \sqrt{\frac{\eta^k}{2}}\mathbf{I} \end{bmatrix}$

The rest of the code follows the BID algorithm steps using the vector values $\boldsymbol{\nu}_j$ to improve over the next iteration until the stopping criteria is met. It probably can be observed that we do several operations going back and forth from vector form to wavelet transform to time domain to Fourier transform, but remember that our intention in this work is to test the potential of sparse vector reconstruction as CS blur removal for the blind and non-blind case.

The quality of the reconstruction compared to the original blur-free image was measured using the error PSNR,

$$\text{PSNR} = 10 \log_{10} \frac{NL^2}{\|\mathbf{x} - \hat{\mathbf{x}}\|^2} \tag{26}$$

where $\mathbf{x}$ and $\hat{\mathbf{x}}$ are the blur-free original and estimated images, respectively, and the constant L represents the maximum possible intensity values in image $\mathbf{x}$.

Chapter 4: Results and Discussion

The initial experimentation used (25) to reconstruct the CS vector observations in mode of one vector at time. Figure 4.1 presents an illustration of the coefficients of vector one with respect to the wavelet transform of an image. The single vector reconstruction cleared does not take into consideration the neighbor coefficients

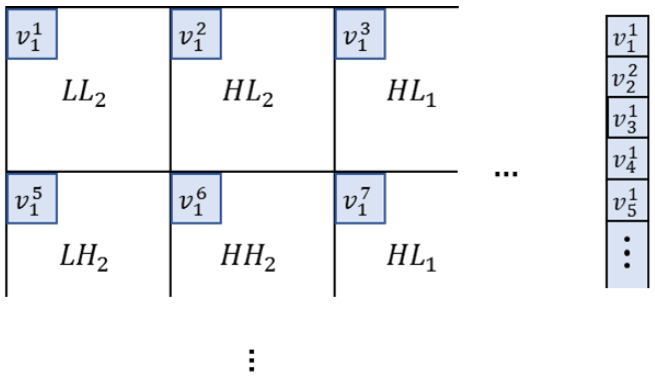


Figure 4.1: Example of vector one selection for reconstruction assuming 2-level wavelet transform

When single vector reconstruction was made, sometimes artifacts appear during the reconstruction. In some image reconstructions, the artifacts cause degradation in the image estimation. Making the reconstruction worse in each iteration instead of improving it. For example, Figure 4.2 displays the reconstruction of the Lena image degraded with Gaussian blur filter variance 9, 30 dB noise, and a CS ratio of 0.6. Through the iterations, a series of artifacts with dot-star shape appear in random places of the image making the end results a noisy reconstruction.

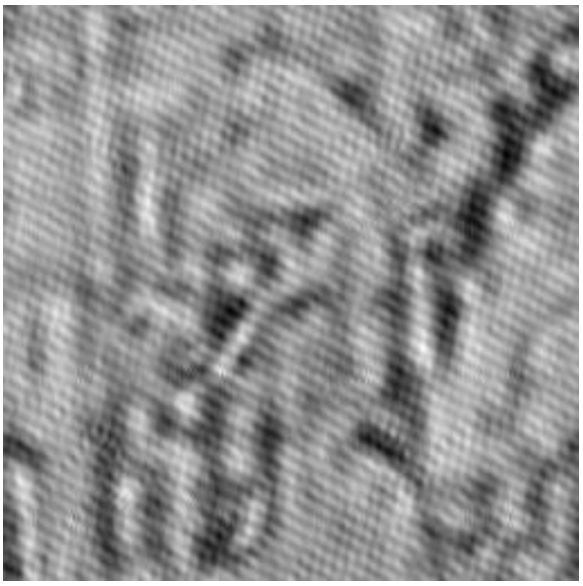


Figure 4.2 Lena reconstruction using single vector reconstruction. Initial image degraded with Gaussian blur filter variance 9, noise 30 dB, and CS ratio of 0.6

In a different experiment, the reconstruction was made possible, but dark dots appear in the final reconstruction. An example can be observed in Figure 4.3 which displays the Cameraman image reconstruction after being degraded with a Gaussian blur filter of variance 9, 40 dB noise, and a CS ratio of 0.8. The proposed algorithm seems to "contain" the degradation this time into a specific area of the image.

Figure 4.3 Cameraman reconstruction of original image degradation with Gaussian blur of variance 9, 40dB noise, and vector compression ratio of .8

The artifact and degradations appear randomly as we concluded several tests using different combinations of parameters and repetition of experiments. For example, in contrast with the images displayed in Figure 4.2 and Figure 4.3, the images displayed in Figure 4.4 have the same conditions except that we increase the compression ratio to 0.4, in Lena image, and 0.3, in the cameraman image. We were expecting greater deformation of the reconstructions. However, the final reconstruction did not show any artifact or degradation beside the ones we intended to correct as part of the experiment.

Figure 4.4: Lena same parameters as in **Figure 4.2** but, compression ratio of .4 (Top). Camera man same degrade parameters as in **Figure 4.3** but, compression ratio of .2

We corrected the appearance of artifacts and image degeneration by reconstructing four neighbor vectors at time as illustrated in Figure 4.5. Following such a configuration in the vector

reconstruction caused the artifacts to disappear preventing further degradation. In addition, we expected an improvement in the blur removal. Thus, we set four vector reconstruction as our standard form of reconstruction in the final set of experiments for the blind and non-blind deconvolution tests.

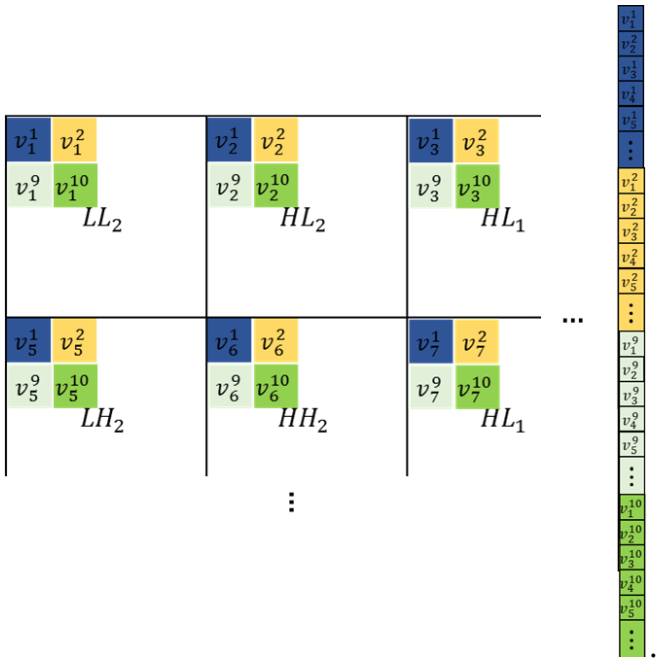


Figure 4.5: Selection of four neighbor vectors to reconstruction and count for blur effect.

Finally, to increase the accuracy in the results, we repeated each experiment ten times and obtained the mean average PSNR for each set of parameters. The results of the reconstruction are summarized in Table 4-2 and Table 4-1 for the Cameraman and Lena, respectively. The Tables display the PSNR for each reconstruction with respect the original blur-free image. In addition, we copied the results obtained from the BID algorithm for direct comparison.

To visually illustrate the reconstructions, Figure 4.6 to Figure 4.17 show the image reconstruction and the filter estimate for the blind and nonblind cases. We can see images Lena and Cameraman with examples of reconstructions with the three different blur degradations of

variance 3, 5, and 9; compression ratio of 0.2, and no compression ratio; and 30 dB noise. Following the results presented in Table 4-2 and Table 4-1, we present only two compression ration extremes and one noise degradation because our algorithm was able to improve few or none regardless the compression ratio and noise. We present our thoughts about this observation in the conclusions section.

Table 4-1: PSNR of the reconstruction for image Lena with different blur, noise, and compression ratio. The results in [1] are displayed in the table as well.

			Lena								
		Blur Var.	*3*			*5*			*9*		
		SNR(noise)	*0dB*	*30dB*	*40dB*	*0dB*	*30dB*	*40dB*	*0dB*	*30dB*	*40dB*
BID Algorithm	*Non-Blind*	*0.2*	-	21.30	20.75	-	20.86	20.66	-	20.22	20.37
		0.4	-	26.98	27.79	-	26.00	26.97	-	24.80	25.77
		0.6	-	28.74	30.55	-	27.35	28.88	-	25.83	27.10
		0.8	-	29.48	31.41	-	27.94	29.52	-	26.30	27.55
		1	-	29.89	31.83	-	28.29	29.85	-	26.60	27.82
	Blind	*0.2*	-	21.07	19.37	-	20.68	19.62	-	19.40	19.28
		0.4	-	26.00	25.25	-	25.19	25.31	-	23.88	24.64
		0.6	-	27.25	26.53	-	26.18	27.70	-	24.59	26.06
		0.8	-	27.69	29.00	-	26.63	28.28	-	24.93	26.56
		1	-	27.90	29.42	-	26.93	28.68	-	25.20	26.78
Proposed	*Non-Blind*	*0.2*	24.25	**24.40**	**24.40**	23.16	**23.45**	**23.45**	21.65	**22.23**	**22.23**
		0.4	24.39	24.44	24.44	23.15	23.46	23.46	21.69	22.23	22.23
		0.6	24.41	24.44	24.44	23.16	23.46	23.46	21.69	22.23	22.23
		0.8	24.43	24.44	24.44	23.14	23.46	23.46	21.70	22.23	22.23
		1	24.44	24.44	24.44	23.14	23.46	23.46	21.70	22.23	22.23
	Blind	*0.2*	24.24	**24.40**	**24.37**	**23.19**	**23.13**	**23.14**	**21.65**	**21.69**	**21.69**
		0.4	24.40	24.44	24.44	23.14	23.12	23.12	21.67	21.69	21.69
		0.6	24.41	24.44	24.44	23.13	23.12	23.12	21.68	21.69	21.69
		0.8	24.43	24.44	24.44	23.14	23.12	23.12	21.69	21.69	21.69
		1	24.43	24.44	24.44	23.14	23.12	23.12	21.70	21.69	21.69

Table 4-2. PSNR of the reconstruction for image Cameraman with different blur, noise, and compression ratio. The results in [1] are displayed in the table as well.

			Cameraman								
		Blur Var.	*3*			*5*			*9*		
		SNR(noise)	*0dB*	*30dB*	*40dB*	*0dB*	*30dB*	*40dB*	*0dB*	*30dB*	*40dB*
BID Algorithm	*Non-Blind*	*0.2*	-	19.86	19.95	-	19.55	19.59	-	19.09	19.17
		0.4	-	24.58	25.05	-	23.68	24.09	-	22.67	22.99
		0.6	-	25.44	26.10	-	24.28	24.76	-	23.10	23.47
		0.8	-	25.68	26.42	-	24.46	24.98	-	23.25	23.63
		1	-	25.83	26.59	-	24.57	25.09	-	23.34	23.74
	Blind	*0.2*	-	19.05	19.10	-	19.01	19.06	-	18.46	18.53
		0.4	-	22.07	22.16	-	21.88	21.75	-	21.65	21.46
		0.6	-	21.83	21.02	-	23.50	23.72	-	22.93	23.25
		0.8	-	22.29	20.80	-	24.00	24.41	-	23.11	23.43
		1	-	21.77	20.34	-	24.28	24.68	-	23.20	23.51
Proposed	*Non-Blind*	*0.2*	22.25	**22.38**	**22.40**	21.46	**21.58**	**21.57**	19.22	**20.27**	**20.22**
		0.4	22.33	22.44	22.44	21.49	21.58	21.60	19.23	20.26	20.24
		0.6	22.40	22.46	22.44	21.47	21.59	21.57	19.23	20.23	20.26
		0.8	22.43	22.43	22.43	21.49	21.57	21.57	19.23	20.18	20.17
		1	22.43	22.43	22.43	21.48	21.57	21.57	19.21	20.18	20.17
	Blind	*0.2*	22.20	**22.35**	**22.36**	21.42	**21.50**	**21.51**	19.23	**19.31**	**19.33**
		0.4	22.31	**22.44**	**22.41**	21.47	21.56	21.56	19.23	19.29	19.35
		0.6	22.35	**22.40**	**22.50**	21.50	21.58	21.53	19.22	19.30	19.31
		0.8	22.37	**22.43**	**22.43**	21.50	21.53	21.53	19.21	19.28	19.27
		1	22.38	**22.43**	**22.43**	21.51	21.53	21.53	19.21	19.27	19.27

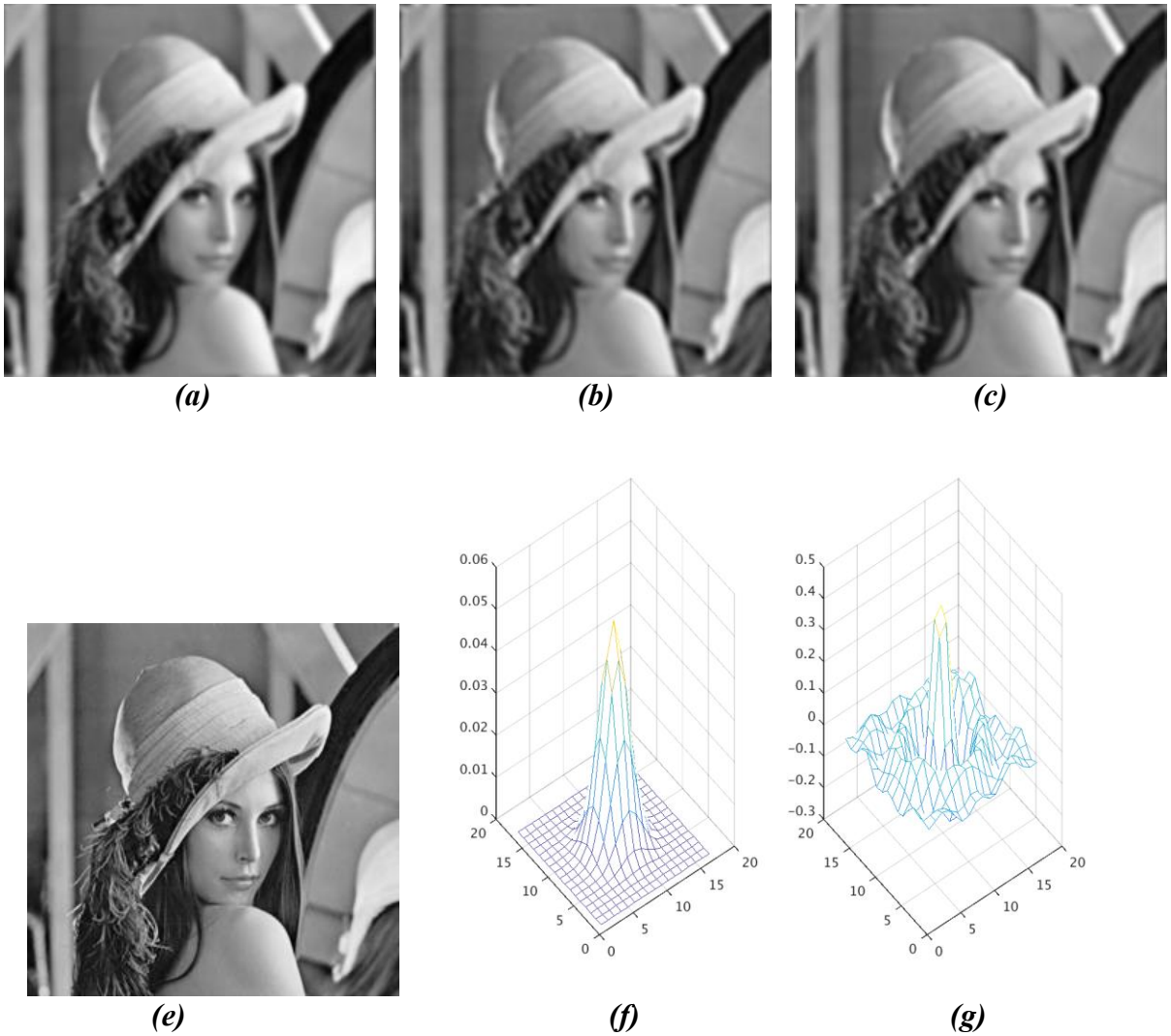


Figure 4.6: Blind Lena reconstruction. **(a)** Initial image degraded with Gaussian blur filter variance 3 and noise 30 dB. **(b)** Reconstruction after using compression ratio of 0.2. **(c)** Reconstruction using no compression ratio. **(e)** Original Lena image. **(f)** Original Gaussian blur filter variance 3. **(g)** Gaussian blur filter estimation

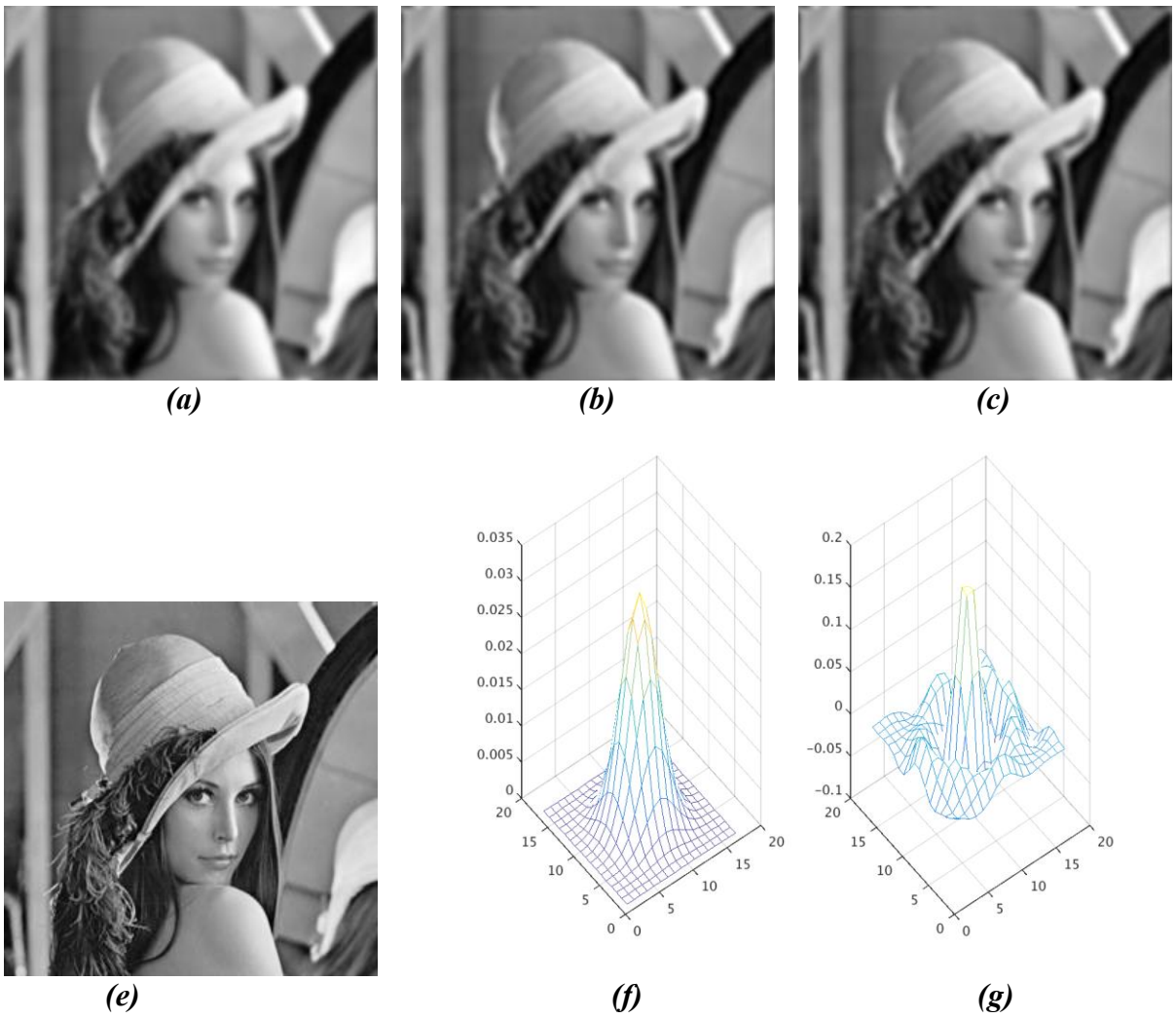


Figure 4.7: Blind Lena reconstruction. **(a)** Initial image degraded with Gaussian blur filter variance 5 and noise 30 dB. **(b)** Reconstruction after using compression ratio of 0.2. **(c)** Reconstruction using no compression ratio. **(e)** Original Lena image. **(f)** Original Gaussian blur filter variance 3. **(g)** Gaussian blur filter estimation.

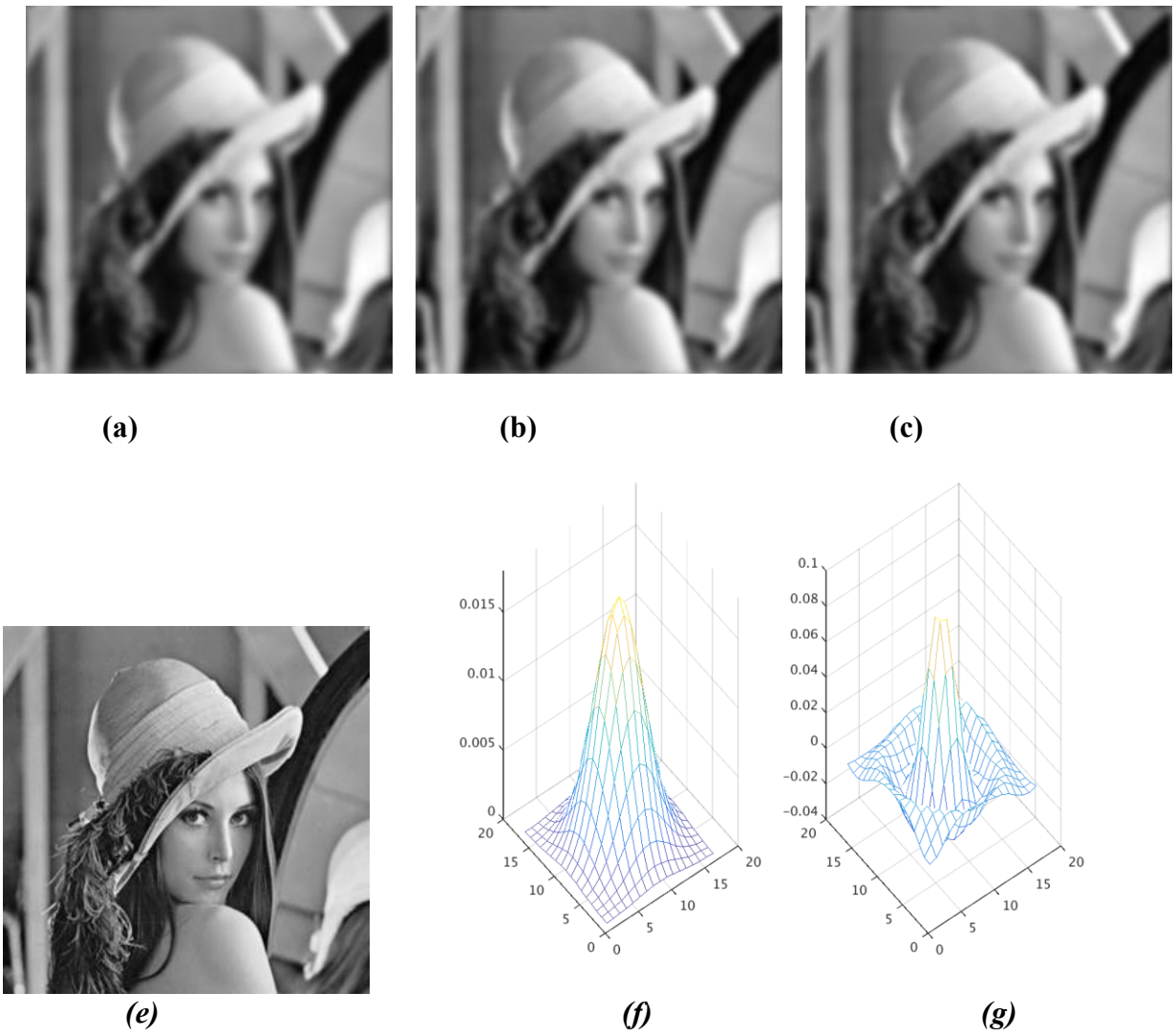


Figure 4.8: Blind Lena reconstruction. **(a)** Initial image degraded with Gaussian blur filter variance 9 and noise 30 dB. **(b)** Reconstruction after using compression ratio of 0.2. **(c)** Reconstruction using no compression ratio. **(e)** Original Lena image. **(f)** Original Gaussian blur filter variance 3. **(g)** Gaussian blur filter estimation.

(a) ***(b)*** ***(c)***

(e)

Figure 4.9: Nonblind Lena reconstruction. **(a)** Initial image degraded with Gaussian blur filter variance 3 and noise 30 dB. **(b)** Reconstruction after using compression ratio of 0.2. **(c)** Reconstruction using no compression ratio **(e)** Original Lena image.

(a) ***(b)*** ***(c)***

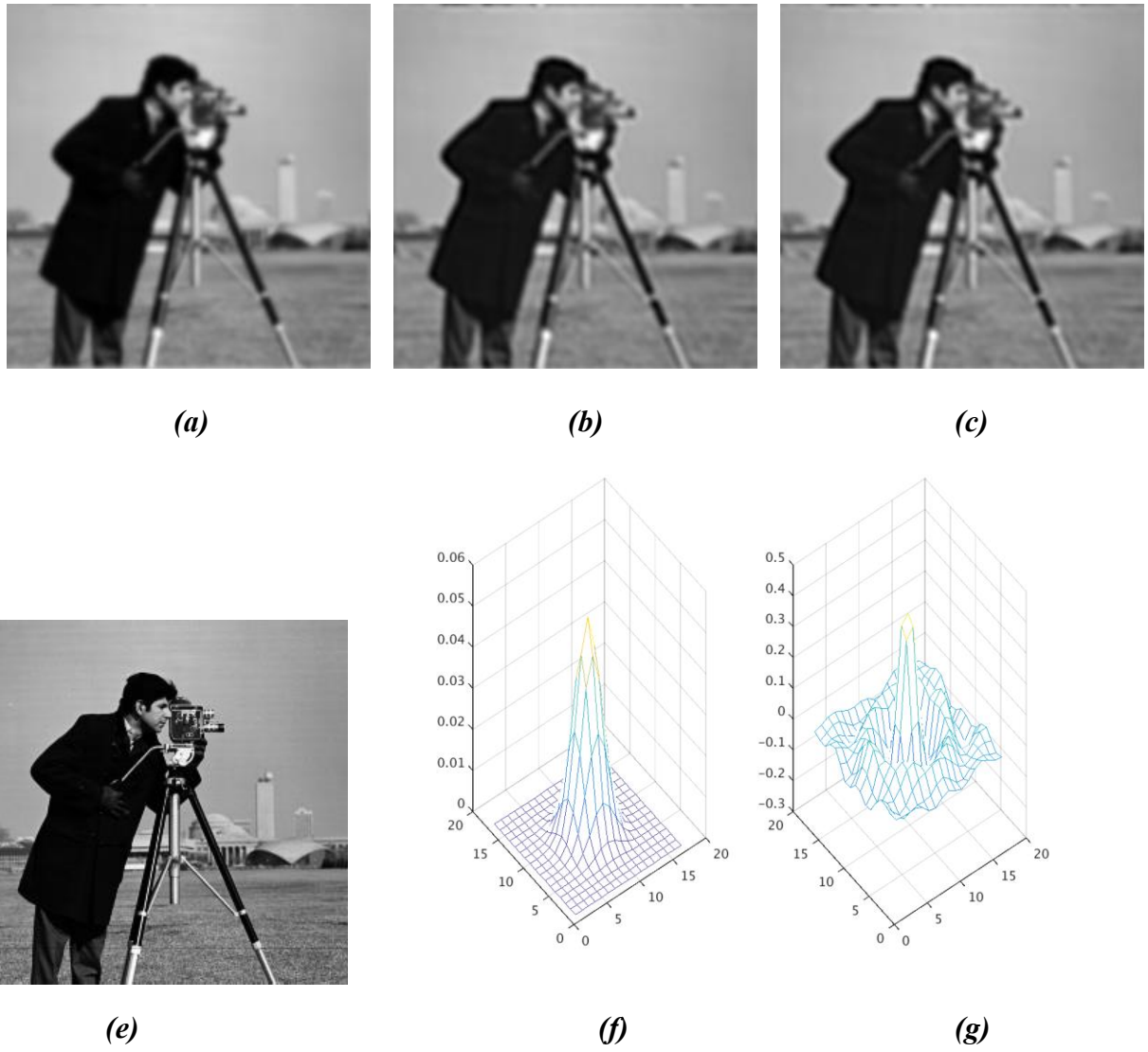
(e)

Figure 4.10: Nonblind Lena reconstruction. **(a)** Initial image degraded with Gaussian blur filter variance 5 and noise 30 dB. **(b)** Reconstruction after using compression ratio of 0.2. **(c)** Reconstruction using no compression ratio. **(e)** Original Lena image**.**

(a) ***(b)*** ***(c)***

(e)

Figure 4.11: Nonblind Lena reconstruction. **(a)** Initial image degraded with Gaussian blur filter variance 9 and noise 30 dB. **(b)** Reconstruction after using compression ratio of 0.2. **(c)** Reconstruction using no compression ratio. **(e)** Original Lena image.

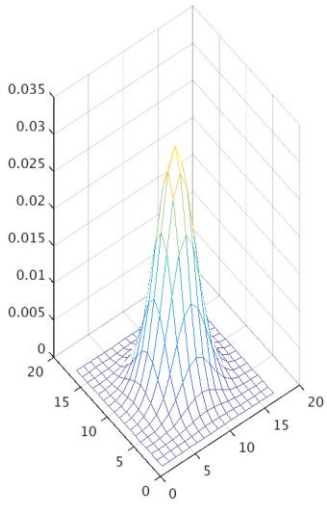


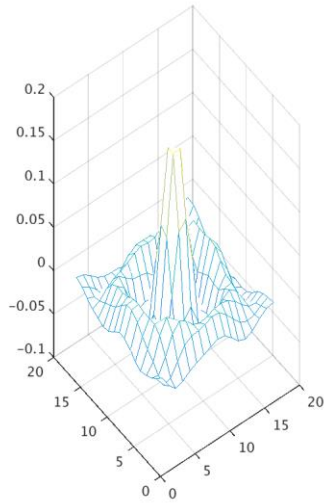
(a) ***(b)*** ***(c)***

(e) ***(f)*** ***(g)***

Figure 4.12: Blind Cameraman reconstruction. **(a)** Initial image degraded with Gaussian blur filter variance 3 and noise 30 dB. **(b)** Reconstruction after using compression ratio of 0.2. **(c)** Reconstruction using no compression ratio. **(e)** Original Cameraman image. **(f)** Original Gaussian blur filter variance 3. **(g)** Gaussian blur filter estimation.

(a) *(b)* *(c)*





(e) *(f)* *(g)*

Figure 4.13: Blind Lena reconstruction. **(a)** Initial image degraded with Gaussian blur filter variance 5 and noise 30 dB. **(b)** Reconstruction after using compression ratio of 0.2. **(c)** Reconstruction using no compression ratio. **(e)** Original Cameraman image. **(f)** Original Gaussian blur filter variance 3. **(g)** Gaussian blur filter estimation.

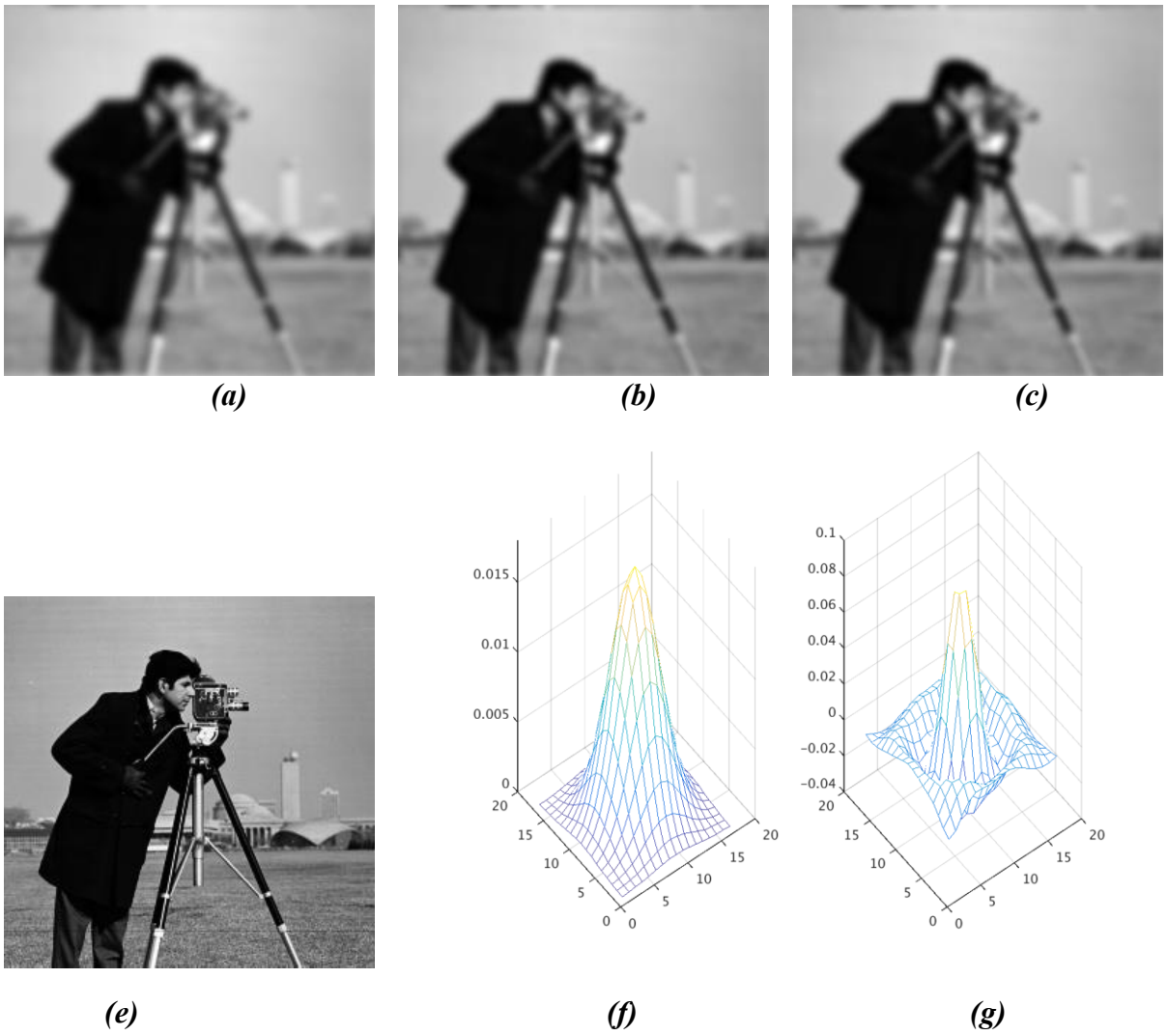


Figure 4.14: Blind Lena reconstruction. **(a)** Initial image degraded with Gaussian blur filter variance 9 and noise 30 dB. **(b)** Reconstruction after using compression ratio of 0.2. **(c)** Reconstruction using no compression ratio. **(e)** Original Cameraman image. **(f)** Original Gaussian blur filter variance 3. **(g)** Gaussian blur filter estimation.

(a) ***(b)*** ***(c)***

(e)

Figure 4.15: Nonblind Cameraman reconstruction. **(a)** Initial image degraded with Gaussian blur filter variance 3 and noise 30 dB. **(b)** Reconstruction after using compression ratio of 0.2. **(c)** Reconstruction using no compression ratio. **(e)** Original Cameraman image**.**

(a) ***(b)*** ***(c)***

(e)

Figure 4.16: Nonblind Cameraman reconstruction. **(a)** Initial image degraded with Gaussian blur filter variance 5 and noise 30 dB. **(b)** Reconstruction after using compression ratio of 0.2. **(c)** Reconstruction using no compression ratio. **(e)** Original Cameraman image**.**

(a) ***(b)*** ***(c)***

(e)

Figure 4.17: Nonblind Cameraman reconstruction. **(a)** Initial image degraded with Gaussian blur filter variance 9 and noise 30 dB. **(b)** Reconstruction after using compression ratio of 0.2. **(c)** Reconstruction using no compression ratio. **(e)** Original Cameraman image**.**

Chapter 5: Conclusion and Future Work

We can observe an improvement in the reconstruction with the highest compression ratio (0.2) with respect to BID algorithm. However, as we decrease the compression ratio, the PSNR tends to converge to a specific value regardless of the increase or decrease of the compression ratio. We conclude from this observation that the proposed modification of the BID algorithm to do vector reconstruction was able to improve the reconstruction of the original blur image, but it was not able to significantly remove the blur effect at the same time. As it is usually in the research, not all the ideas in the research field were made to be right and the ideas proposed here had potential in theory. At least this time, it did not perform as expected.

We did a quick comparison of different image reconstructions with their original blur image and the PSNR between them was significantly higher PSNR about 140 – 240 dBs. Following this observation, future work could include dividing the reconstruction of the CS measurement and deblurring stage into two separate stages. Having a more accurate reconstruction of the initial blurred image could translate into a better deblurring post-stage. Another possible idea to test could be to increase the number of vectors to be reconstrued from four to nine or sixteen neighbor's vectors. This include increasing the wavelet depth to reduce the block size and thus the vector length.

References

[1] B. Amizic, L. Spinoulas, R. Molina and A. K. Katsaggelos, "Compressive Blind Image Deconvolution," *IEEE Transactions on Image Processing,* vol. 22, no. 10, pp. 3994 - 4006, Oct. 2013.

[2] M. Ali Qureshi and M. Deriche, "A new wavelet based efficient image compression algorithm using compressive sensing," *Springer,* 2015.

[3] R. C. Gonzalez, R. E. Woods and S. L. Eddins, Digital Image Processing Using MATLAB, Gatesmark Publishing, 2010.

[4] E. J. Candes and M. B. Wakin, "An Introduction to Compressive Sampling," *IEEE Signal Processing Magazine,* vol. 25, no. 2, pp. 21-30, 2008.

[5] M. F. Duarte, M. A. Devenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly and R. G. Baraniuk, "Single-Pixel Imaging via Compressive Sampling," *IEEE Signal Processing Magazine ,* vol. 25, no. 2, pp. 83-91, 2008.

[6] C. Hegde and R. G. Baraniuk, "Compressive Sensing of Streams of Pulses," in *Allerton Conference on Communication, Control, and Computing*, Ilinois, 2009.

[7] J. Ma and F.-X. Le Dimet, "Deblurring From Highly Incomplete Measurements for Remote Sensing," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 47, no. 3, pp. 792-802, 2009.

[8] L. McMackin, M. A. Herman, B. Chatterjee and M. Weldon, "A high-resolution SWIR camera via compressed sensing," *Proc. of SPIE,* vol. 8353, no. 1, p. 83530, 2012.

[9] B. Amizic, L. Spinoulas, R. Molina and A. K. Katsaggelos, "Compressive sampling with unknown blurring function: Application to passive millimeter-wave imaging," in *19th IEEE International Conference on Image Processing (ICIP)*, Orlando, FL, USA, 2012.

[10] InView, "Compressive Sensing," InView, [Online]. Available: http://inviewcorp.com/applications/compressive-sensing/#.WPweddLys2w. [Accessed 17 April 2017].

[11] J. R. Stinnett and J. Gillenwater, "Compressive Imaging," [Online]. Available: https://cnx.org/contents/BEpEBtWm@1/Compressive-Imaging. [Accessed 17 April 2017].

[12] L. Xiao, J. Shao, L. Huang and Z. Wei, "Compounded Regularization and Fast Algorithm for Compressive Sensing Deconvolution," in *Proc. 6th Int. Conf. Image and Graphics*, 2011.

[13] R. Chartrand and W. Yin, "Iteratively Reweighted Algorithms for Compressive Sensing," in *IEEE international Conference on Acoustics, Speech and Signal Processing*, Las Vegas, NV, 2008.

[14] E. Candes and T. Tao, "Decoding by Linear Programming," *IEEE Transactions on Information Theory,* vol. 51, no. 12, pp. 4203 - 4215, 2005.

[15] E. Candes, J. Romberg and T. Tao, "Stable Signal Recovery from Incomplete and Inaccurate Measurements," *Communications on Pure and Applied Mathematics,* vol. 59, no. 8, pp. 1207-1223, 2006.

[16] E. J. Candes and T. Tao, "Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies?," *IEEE Transactions on Information Theory,* vol. 52, no. 12, pp. 5406 - 5425, 2006.

[17] T. E. Bishop, D. S. Babacan, B. Amizic, A. K. Katsaggelos, T. Chan and R. Molina, Blind image deconvolution: problem formulation and existing approaches, Cleveland, OH, USA: CRC Press, 2007.

[18] E. Candès and J. Romberg, "Practical Signal Recovery from Random Projections," in *Wavelet Application in Signal and Image Processing XI*, 2005.

[19] S.-J. Kim, K. Koh, M. Lustig, S. Boyd and D. Gorinevsky, "An interior-point method for large-scale l1-regularized least squares," *IEEE Journal on Selected Topics in Signal Processing,* vol. 1, no. 4, pp. 606-617, 2007.

[20] M. Figueiredo, R. Nowak and S. Wright, "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *IEEE J. Sel. Topics Signal Process.,* vol. 1, no. 4, pp. 586-597, 2007.

[21] S. Becker, J. Bobin and E. J. Candés, "NESTA: A fast and accurate first-order method for sparse recovery," *SIAM J. Imag. Sci.,* vol. 4, no. 1, pp. 1-39, 2011.

[22] J. Yang and Y. Zhang, "Alternating direction algorithms for l1-problems in compressive sensing," *SIAM J. Sci. Comput.,* vol. 1, pp. 250-278, 2011.

[23] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Commun. ACM,* vol. 53, no. 12, pp. 93-100, 2010.

[24] Y. Y.-S and A. M. G, "Compressed sensing technique for high-resolution radar imaging," *SPIE Defense and Security Symposium, International Society for Optics and Photonics,* vol. 6968, no. 69, pp. 681A.1- 681A.10, 2008.

[25] G. Z, X. C, D. L and Z. C, "Image representation using block compressive sensing for compression applications," *Vis. Commun. Image Represent,* vol. 24, no. 7, p. 885–894, 2013.

[26] D. M, Z. H, Z. C and L. B, "The application of wavelet-based contourlet transform on compressed sensing," in *Multimedia and Signal Processing. Communications in Computer and Information Science.*, Springer, Berlin, Heidelberg, 2012.

[27] F. Cervantes, "Blind image deconvolution based on sparsity : theoretical justification and improvement of state-of-the-art techniques," The University of Texas at El Paso, El Paso , 2016.

[28] D. Krishnan and R. Fergus, "Fast image deconvolution using hyper-Laplacian priors," in *Neural Information Processing Systems*, Vancouver, 2009.

[29] A. Levin, R. Fergus, F. Durand and W. T. Freeman, "Image and depth from a conventional camera with a coded aperture," *ACM Trans. Graph.,* vol. 26, no. 3, pp. 1-8, 2007.

[30] R. Molina, J. Mateos and A. Katsaggelos, "Blind Deconvolution Using a Vriational Approach to Parameter, Image, and Blur Estimation," *IEEE Transactions on Image Processing,* vol. 15, no. 12, pp. 3715 - 3727, 2006.

Appendix

A.1 List of Symbols

$\mathbf{x}$	signal of interest (image) in $\mathbb{R}^N$
$\tilde{\mathbf{x}}$	estimate of original signal (image) in $\mathbb{R}^N$
$\mathbf{Hx}$	blurred signal (image)
$\mathbf{\Phi}$	measurement matrix in $\mathbb{R}^{M\times N}$, $M \ll N$
$\mathbf{y}$	Compressive sensing observation in $\mathbb{R}^{M\times N}$ (either, from a blur or blur-free image)
$\mathbf{\Psi} = \boldsymbol{W}$	inverse transformation matrix in $\mathbb{R}^{N\times N}$. In this work, it represents wavelet inverse matrix. $\boldsymbol{W}$ is used to emphases that the signal of interest was initialized degraded with blur.
$\mathbf{\Psi}'$	wavelet transformation matrix in $\mathbb{R}^{N\times N}$
$\mathbf{s}$	Wavelet coefficient of a blur-free signal
$\tilde{\mathbf{s}}$	estimate of $\mathbf{s}$
$\mathbf{a}$	wavelet transform of signal of interest degraded with blur
$\mathbf{A} = \mathbf{\Phi\Psi}$	sensing matrix in $\mathbb{R}^{M\times N}$
$\mathbf{y_h}$	blur signal (blur image) with noise
$\mathbf{H}$	low-pass linear blurring operator
$\mathbf{H}^{-1}$	inverse of the low-pass linear operator
$\mathbf{n}$	measurement error or noise
$\mathbf{P}^{-1}$	inverse operator (equivalent but not equal to $\mathbf{H}^{-1}$) with constraints $\mathbf{P}^{-1}\mathbf{Hx} \approx \mathbf{x}$ and $\mathbf{P}^{-1}\mathbf{n} \approx 0$.
$\mathbf{h}$	function modeling blur called Point Spread Function (PSF)

$\boldsymbol{v}_j$	set of sparse vectors obtained from the wavelet transform
j	vector index
B	length of sparse vectors ($\mathbb{R}^B$)
q	Total number of sparse vector
$\boldsymbol{\Omega}$	weighted matrix used to normalize measurement matrix Φ, where $\Omega = [\omega; \omega; \dots, \omega]$ and $\omega = [E_1, E_2, .. E_B]$. The row vector **E** is defined in equation (15)
$\boldsymbol{\Phi}_E$	weighted measurement matrix
$m\times$n	(lowercase) image size
L	wavelet transform depth level
k and l	represent internal iteration counters
$\eta,\alpha, p, \gamma, \beta, \tau$	constants
	$o(d) \in \{1,2\}$
	$d \in D = \{h, v, hh, vv, hv\}$ see reference [1] for details

A.2 Matlab Code

```
% MATLAB CODE in which it is intended to combine the ideas of papers:
% 1. "A new wavelet based efficient image compression algorithm using
%      compressive sensing" by Qureshi and Deriche
% AND
% 2. "Compressive Blind Image Deconvolution"(CBID) by Amizic, Spinoulas,
% Molina,
%     and Katsaggelos
% The code use as starting point the recreation of CBID By Fernando Cervantes
% in “Blind image deconvolution based on sparsity : theoretical justification
% and improvement of state-of-the-art techniques”
%
% Modification made by Alonso Orea Amador

clear all; close all;

% Load image
    %% Image 1
      x = double(imread('cameraman.tif'));
      name = 'cameraman';
      [m,n] = size(x);
      %figure; imagesc(x); axis image; colormap(pink);
for exp = 1:10
% Adding blur and some noise
blurVec = [3,5,9];
for inxBlur = 1:3
    blurVar = blurVec(inxBlur) ;
    h1=fspecial('gaussian',17,sqrt(blurVar));
    %h1=fspecial('gaussian',17,sqrt(5));
    %h1=fspecial('gaussian',17,sqrt(9));
    sigVec = [1e-3,1e-4];
for sigInd = 1:2;
    sigma=sigVec(sigInd);
    noiseSRN = 10*log10(1/sigma);
    filter = 'coif5';
    L = 3; % wavelet level -> i.e. level depth of the decomposition
    NumofVec =  m*n/2^(2*L);
    f_bn =
convolution(x,h1,1)+sigma*randn(m,n);%imfilter(x,h1,'replicate');%+sigma*rand
n(m,n);
       % figure; imagesc(f_bn); axis image; colormap gray;
% Wavelet transform 3 level decomposition
    a = im2wav(f_bn,filter,L);
% Form sparse vectors
    vectors = w2vec(a,L);
% normalize the Gaussian measuement matrix using the energy weighting matrix
   % Let's test Measurement ration (MR) = 0.2 -> 13 measumentes
    rmse = sqrt(sum(abs(vectors).^2,2)); % Single vector. Equation (10)

    B = 1+3*sum(4.^(0:L-1)); % equalt to B length of the vectors
%% For loop for compression ration
ratio = [.2,.4,.6,.8,1];
for temp1 = 1:5
    comprRatio = ratio(temp1);
    K = round(B*comprRatio);
    phi = randn(K, B); % Define Gaussian measument matrix
```

```
    phi = orth(phi')';
    RMSE = zeros(K,B); % Allocate space for to input energy wighted matrix
value
    % Creting energy weighting matrix
    for i = 1:K
    RMSE(i,:) = rmse';
    end
    wphi = orth((phi.*RMSE)')'; % Weighted phi

% obtain the compressed measurements.
    y = wphi*vectors;
%% Reconstruction code
    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);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %constants
    h=fspecial('gaussian',17,sqrt(3)); % Initial guess for h1
    eta=1042;
    gamma=5e5;
    exp_a=0.8;
    thr_e=0.0001;
    beta=sigma^2;
    theta=1.3;
    alpha=1;
    p=0.8;
    tau=0.125;
 % Difference operator
    dxf=[1 -1];          D_xf = zeropad(dxf,m,n); % correction left
    dyf=[1;-1];          D_yf = zeropad(dyf,m,n); % correction above
    dxxf=[-1, 2, -1];    D_xxf= zeropad(dxxf,m,n);
    dyyf=[-1; 2; -1];    D_yyf= zeropad(dyyf,m,n);
    dxyf=[-1 1;1 -1];    D_xyf= zeropad(dxyf,m,n);

    av0 =wphi'*y; % Initial guess for a0
    ak = vec2w(av0,m,n,L);
    %    figure; imagesc(ak); axis image; colormap pink;

%    Inverse Wavelet transform and plot reconstructed image
    Wa0 = wav2im(ak,filter,L);
    %    figure; imagesc(Wa0); axis image; colormap pink;

    % Matrix
    temp_N=256^2;
    %V=rand(temp_N,5);
    V = zeros(temp_N,5);


    N2 = 256*2;
    Wak =Wa0; % = Hx, inverse wavelet transform  ????
```

```
    xkl = Wa0; % Initial image guess
    thr_e=0.0001;

 %Initializing
        Xdx = convolution(xkl,dxf,1);
        Xdy = convolution(xkl,dyf,1);
        Xdxx= convolution(xkl,dxxf,1
        Xdyy= convolution(xkl,dyyf,1
        Xdxy= convolution(xkl,dxyf,1

        W1= max(diag(diag(abs(Xdx).^2)),thr_e);  V(:,1)=W1(:);
        W2= max(diag(diag(abs(Xdy).^2)),thr_e);  V(:,2)=W2(:);
        W3= max(diag(diag(abs(Xdxx).^2)),thr_e); V(:,3)=W3(:);
        W4 = max(diag(diag(abs(Xdyy).^2)),thr_e); V(:,4)=W4(:);
        W5= max(diag(diag(abs(Xdxy).^2)),thr_e); V(:,5)=W5(:);
        xklold = ones(256,256);
        xklT = xklold;

while(norn(xkl-xklod)/norm(xklold)) > 1e-2)
        xklold = xkl;
        Wak = wav2im(ak,filter,L);

        B_x  = reshape(V(:,1),m,n); B_x = B_x.^(p/2-1
        B_y  = reshape(V(:,2),m,n); B_y = B_y.^(p/2-1
        B_xx = reshape(V(:,3),m,n); B_xx= B_xx.^(p/2-1
        B_yy = reshape(V(:,4),m,n);  B_yy= B_yy.^(p/2-1
        B_xy = reshape(V(:,5),m,n);  B_xy= B_xy.^(p/2-1

        H = zeropad(h,m,n);
        H_H = eta*conj(fft2(H)).*fft2(H);

        Delta_x  =
alpha*p*conj(fft2(D_xf)).*ifft2(fft2(fft2(B_x)).*fft2(fft2(D_xf)))/(m*n);
        Delta_y  =
alpha*p*conj(fft2(D_yf)).*ifft2(fft2(fft2(B_y)).*fft2(fft2(D_yf)))/(m*n);
        Delta_xx =
alpha*p*0.5*conj(fft2(D_xxf)).*ifft2(fft2(fft2(B_xx)).*fft2(fft2(D_xxf)))/(m*
n);
        Delta_yy =
alpha*p*0.5*conj(fft2(D_yyf)).*ifft2(fft2(fft2(B_yy)).*fft2(fft2(D_yyf)))/(m*
n);
        Delta_xy =
alpha*p*0.5*conj(fft2(D_xyf)).*ifft2(fft2(fft2(B_xy)).*fft2(fft2(D_xyf)))/(m*
n);

        part1 = H_H +Delta_x+Delta_y+Delta_xx+Delta_yy+Delta_xy;
        part2 = eta*conj(fft2(H)).*fft2(Wak);
        xklf=part2./part1;
        xkl=real(ifft2(xklf));
        xkl=xkl(1:256,1:256);
        xkl = circshift(xkl,[9 9]);
        xkl=reshape(xkl,m,n);
         %   figure; imagesc(xkl); colormap('pink'); axis image;

       xklT=xkl;
```

```
        xklT=fftshift(xklT);
        Ahf=eta*conj(fft2(xklT)).*fft2(xklT)+gamma*conj(fft2(C)).*fft2(C);
        bhf=eta*conj(fft2(xklT)).*fft2(Wak);
        hf=bhf./Ahf;
        h=real(ifft2(hf));
        %figure; imagesc(h);
        h=h(120:136,120:136);
        % h = h1; WHEN Non-Blind Deconvolution, we simple set h=h1


    % Update values
        Xdx = convolution(xkl,dxf,1);
        Xdy = convolution(xkl,dyf,1);
        Xdxx= convolution(xkl,dxxf,1);
        Xdyy= convolution(xkl,dyyf,1);
        Xdxy= convolution(xkl,dxyf,1);

        W1 = max(diag(diag(abs(Xdx).^2)),thr_e);  V(:,1)=W1(:);
        W2 = max(diag(diag(abs(Xdy).^2)),thr_e);  V(:,2)=W2(:);
        W3 = max(diag(diag(abs(Xdxx).^2)),thr_e); V(:,3)=W3(:);
        W4 = max(diag(diag(abs(Xdyy).^2)).^2,thr_e); V(:,4)=W4(:);
        W5 = max(diag(diag(abs(Xdxy).^2)),thr_e); V(:,5)=W5(:);


HX=convolution(xkl,h,1);%imfilter(xkl,h,'replicate');%conv2(xkl,h,'same');
        hx = im2wav(HX,filter,L);
        vecHXdwt = w2vec(hx,L);
        akvec =  zeros(B,NumofVec);%zeros(B,numbOfVectors);

%%%% Single Vector Reconstruction %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%         for i=1: NumofVec
%             Y = [sqrt(beta/2)*y(:,i);sqrt(eta/2)*vecHXdwt(:,i)];
%             PHI=[sqrt(beta/2)*wphi;sqrt(eta/2)*eye(B)];
%             A = PHI;
%             b = Y;
%             [akvec(:,i),status,history] = l1_ls(A,b,tau);
% %%       end

% Four neighbor vector reconstruction.%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        NumRec = 4;
        for i =0:15
            indx = (i*64)+1;
            for j=1:31
               RecVec = [indx,indx+1,indx+32,indx+33];
               block_y = y(:,RecVec);
               blcok_H = vecHXdwt(:,RecVec);
               Y=[sqrt(beta/2)*block_y(:);sqrt(eta/2)*blcok_H(:)];
               block_wphi = [wphi,
zeros(K,3*64);zeros(K,64),wphi,zeros(K,64*2);zeros(K,2*64),wphi,zeros(K,64);z
eros(K,3*64),wphi];
               PHI=[sqrt(beta/2)*block_wphi;sqrt(eta/2)*eye(NumRec*64)];
                A = PHI;
                b = Y;
                [block_akvec,status,history] = l1_ls(A,b,tau);
                tempRec =reshape(block_akvec,64,NumRec);
                akvec(:,RecVec) = (akvec(:,RecVec)+tempRec);
```

```
                if indx > 1 && indx <= 31
                    akvec(:,RecVec(1)) = akvec(:,RecVec(1))/2 ;
                    akvec(:,RecVec(3)) = akvec(:,RecVec(3))/2;
                end

                indx = indx+1;
            end
        end

        ak = vec2w(akvec,m,n,L);
        eta=eta*theta;
        fm = xkl-min(xkl(:));
    fmax = max(max(fm));
    x_apx = 255*fm/fmax;
    x_rec = uint8(round(x_apx));
    x_orig = uint8(x);
    PSNRval = psnr(x_rec, x_orig);
%    figure; imagesc(x_rec); colormap('gray'); axis image;
end

%figure; imagesc(x_rec); colormap('gray'); axis image;
% Filters

% Nomarlize xkl to range [0,255]
% Images
imwrite(x_rec,[name,'_exp',int2str(exp),'_Blur',int2str(blurVar),'_noiseSNR',
int2str(noiseSRN),'_ratio',int2str(comprRatio*10),'.tif']);
% PSNR
save([name,'PSNRexp',int2str(exp),'_Blur',int2str(blurVar),'_noiseSNR',int2st
r(noiseSRN),'_ratio',int2str(comprRatio*10),'.mat'],'PSNRval');
   %  figure; imagesc(x_rec); colormap('gray'); axis image;
    % figure; imagesc(x_orig); colormap('gray'); axis image;
figure;
subplot(1,2,1)
mesh(h1)
subplot(1,2,2)
mesh(h);
saveas(gcf,[name,'Filters_exp',int2str(exp),'_Blur',int2str(blurVar),'_noiseS
NR',int2str(noiseSRN),'_ratio',int2str(comprRatio*10),'.fig']);
saveas(gcf,[name,'Filters_exp',int2str(exp),'_Blur',int2str(blurVar),'_noiseS
NR',int2str(noiseSRN),'_ratio',int2str(comprRatio*10),'.png']);
close
end
end
end
end
```

A.3 Matlab Special Functions

```
% Obtain image's Wavelet Transform same size as 'im'
% im - input image
% filter - wavelet filter - ex - 'haar', 'db2', 'coif5'
function [ wav ] = im2wav(im,filter,L)
    %'im' -  image you want to transform
    %'filter' - kind of filter you want to implement
    %'L' - Level of decomposition
    [n,m] = size(im);
    wav = zeros(n,m);
    cAi = im;
    for i = 1:L
        [cAi,cHi,cVi,cDi] = dwt2(cAi,filter,'mode','per');
        wav(1:n/2^i,n/2^i+1:n/2^(i-1)) = cHi;
        wav(n/2^i+1:n/2^(i-1),1:n/2^i) = cVi;
        wav(n/2^i+1:n/2^(i-1),n/2^i+1:n/2^(i-1)) = cDi;
    end
    wav(1:n/2^i,1:n/2^i) = cAi;

function [ vectors] = w2vec(a,L)
% Break down an wavelet transform on "image" form into its sparse vectors %
according to paper: "A new wavelet based efficient image compression
algorithm using compressive sensing" by Qureshi and Deriche
% a -  wavelet transform image
% L  - wavelet level -> i.e. level depth of the decomposition
    [m,n] = size(a);
    numbOfVectors = m*n/2^(2*L); % Number of sparse vector. Equation descript
in 2.
   % Number of Blocks (Dimension of each vector)
        temp_i = 1:L;
        temp_s = 4.^(temp_i-1);
        B = 1+3*sum(temp_s); % Dimension of vector. Equation (8) in paper
        vectors = zeros(B,numbOfVectors); % Allocated space for sparse vector
values
    % Input values to each sparse vector
        ii=1; %counter
        g =1;%counter
        block = (n/2^L);
        for row = 0:block-1
            for col = 0:block-1
                for i=1+row:block:n
                    for j =1+col:block:m
                        vectors(ii,g) =a(i,j);
                        ii=ii+1;
                    end
                end
                ii =1;
        g =g+1;
            end
        end

end
```

```
function [ ak ] = vec2w(vec,m,n,L)
% opposite of w2vec - reconstruction of wavelet image form its sparse vectors
% vec - sparse vectors  to convert into wavelet image
% m,n  - dimension of image
    % Rocover sparse vectors
        block = (n/2^L);
        ii=1; % counter
        g =1;% counter
        ak = zeros(m,n);
        for r = 0:block-1
            for s = 0:block-1
                for i=1+r:block:256
                    for j =1+s:block:256
                        ak(i,j) = vec(ii,g); % Initial guess for vectors
                        ii=ii+1;
                    end
                end
                ii =1;
        g =g+1;
            end
        end

end
function [ im ] = wav2im( wav,filter,L)
% opposite of im2wav
    [m,n] = size(wav);
    cAi = wav(1:n/2^L,1:n/2^L);
    for i = L:-1:1
        cHi = wav(1:n/2^i,n/2^i+1:n/2^(i-1));
        cVi = wav(n/2^i+1:n/2^(i-1),1:n/2^i);
        cDi = wav(n/2^i+1:n/2^(i-1),n/2^i+1:n/2^(i-1));
        cAi = idwt2(cAi,cHi,cVi,cDi,filter,'mode','per');
   end
    im = cAi;


function [ y] = zeropad(x,m,n) (BY Dr. Fernando Cervantes)
% add zero entries to matrix x to be size m by n. Original size of x is
% less than mxn

y = zeros(m,n);
[k,h] = size(x);
y(1:k,1:h) = x;

end
```

Vita

Alonso Orea completed his undergraduate degree in Electrical Engineering at the University of Texas at El Paso in 2014. He graduated with honors among his generation. During his undergraduate studies, he did research under the supervision of Dr. Ricardo von Borries and Dr. Berenice Verdin in Compressive Sensing with Prior Information Applied to Lidar Systems. In addition, he did an internship with the National Center for Border Security and Immigration (NCBSI) as Summer Scholar in 2013.

He entered the Master Degree in Electrical engineering at the same university. During his Master Degree, he was awarded the Texas Instrument Endowed Scholarship three times. He collaborated with the Electrical and Computer Engineering Department as Peer Advisor and Teacher Assistant. In addition, he was Web Master and then President of the Theta Chapter of Tau Beta Pi - The Engineering Honor Society, 2014-2017.

Permanent address: 1501 N Oregon ST APT 2
El Paso, TX, 79902

This thesis was typed by Alonso Orea Amador.