8-26-2019 11:30 AM

Optimizing the usage of 2D and 3D transformations to improve the BM3D image denoising algorithm

Zaied Zaman
The University of Western Ontario

Supervisor
El-Sakka, Mahmoud
The University of Western Ontario

Graduate Program in Computer Science
A thesis submitted in partial fulfillment of the requirements for the degree in Master of Science

Abstract

Image denoising is one of the most important preprocessing steps before a wide range of applications such as image restoration, visual tracking, image segmentation, etc. Numerous studies have been conducted to improve the denoising performance. *Block Matching and 3D (BM3D) filtering* is the current state-of-the-art algorithm in image denoising and can provide better denoising performance than other existing methods. However, still, there is scope to improve the performance of BM3D. In this thesis, we have pointed out an aspect of the algorithm which can be improved and suggested an approach to improve it. We have proposed to perform a 2D and 3D transformation on certain patches rather than performing a 3D transformation on all the patches.

Keywords: Image Denoising, BM3D, 2D transform, 3D transform, Collaborative filtering

Lay Summary

Pixel values of natural images get corrupted by noise mostly in transmission and acquisition steps. It is needed to denoise or estimate the true pixel values from noisy pixel values for sophisticated imaging applications. Block-Matching and 3D filtering (BM3D) algorithm is one of the state-of-the-art algorithms to denoise natural images. In this thesis, we aim at improving the denoising performance of BM3D even further. BM3D uses a fixed approach (3D transformation) for the whole image. In this thesis, we have proposed an adaptive way to choose between two techniques (2D transformation and 3D transformation) for appropriate scenarios.

Acknowlegements

I would like to express my heartiest gratitude to the Almighty for the strength, patience, intelligence and endless kindness he provided me with to finalize this thesis. I am grateful to my honorable supervisor Dr. Mahmoud R. El-Sakka for his valuable direction, guidance, comments, and encouragement throughout this work. It was an absolute honor and privilege to work with such a modest and wise person like him. His wisdom and notable thoughts have helped this thesis become an ultimate success. He redirected my view of thinking to a progressive path every time I discussed my research problems with him. This dissertation under his supervision will always be a remarkable experience in my life. I would like to acknowledge my heartiest gratitude to all the professors of The University of Western Ontario who helped me through many courses to build my background for this dissertation. I acknowledge the support of my friends, family members and research group members throughout this long tiring period. Finally, I would like to thank The Department of Computer Science at The University of Western Ontario to fund my graduate study.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A digital image can be defined as two-dimensional discrete functions which can be represented by a two-dimensional matrix. The height of the image is the number of rows of the matrix and the width of the image is the number of columns of the matrix. Each of the entries in this matrix is defined as a pixel. If we represent this matrix as f, then $f(i, j)$ represents a specific pixel which is located at spatial coordinate (i, j). The value of $f(i, j)$ is called a pixel value, intensity value, brightness value interchangeably.

During the acquisition or transmission, digital images may be contaminated by noise. Faulty instruments, interfering natural phenomena, lossy compression can be accounted for as some of the many reasons for contaminations. True pixel values of the image get distorted because of noise. This distorted pixel values can produce erroneous results in sophisticated imaging applications such as satellite imaging, medical imaging, etc. So, the denoising is needed to estimate the true pixel value before the image goes to these applications. Thus, Image denoising performance is of high importance and considered as the most important pre-processing step before these sophisticated applications.

Image denoising is well studied over the past decades. Among other approaches, Block Matching and 3D(BM3D) filtering is an algorithm developed to improve the denoising performance. BM3D is currently the state-of-the-art algorithm for image denoising.

In this chapter, we will discuss our problem description and will point out the objectives of this thesis. We will also describe the major contributions and outline of this thesis.

1.1 Motivations

Digital noise can be of many types. Different techniques are available to denoise different types of noises in the literature. Noise is random. According to the central limit theorem [28], irrespective of the base distribution, if we sum samples taken from a random distribution, the

distribution approaches to normal or Gaussian distribution. As we are simulating practical scenarios, where the source of noise is seldom known, if we assume that the noise affecting our image is Gaussian, we can have a better scenario.

During the years, various approaches have been developed to denoise gaussian noisy images. BM3D is the current state-of-the-art algorithm to denoise gaussian noisy images. BM3D utilizes the redundancy of similar patches available in the image. BM3D works in two steps. It produces a basic estimate image in the first step. Then, it uses Wiener filtering to compute the final denoised image. The basic estimate image is to better facilitate the Wiener filtering.

It is worth mentioning that the performance of BM3D degrades for higher noise levels. Hence, the performance is still not sufficient for sensitive applications.

1.2 Problem Statement

To exploit the existing correlation between patches in the natural images, BM3D performs 3D transform regardless of whether enough similar patches are available or not. If enough similar patches are not available, exploiting the correlation between dissimilar patches might produce degraded denoised estimation. Hence, we suggest exploiting only the correlation between the patch itself to get a better-denoised estimate if enough similar patches are not available. We can exploit the correlation between the patch itself by performing a 2D transform when enough similar patches are not available.

1.3 Our Objective

Our objective is to improve the denoising performance of BM3D. BM3D uses 3D transformation irrespective of whether enough similar patches are available or not. We aim at finding an optimized way of utilizing both 2D and 3D transformation.

1.4 Thesis Contribution

The major contribution of this thesis is the improvement of the performance of the BM3D algorithm. We will show that if we attempt to combine dissimilar patches to perform 3D transform to exploit the correlation between patches, we might end up with a degraded denoising performance. We will also show that if enough number of similar patches are not available in the image, it is better to perform a 2D transform to produce a denoised version with the help of the information contained in the patch itself.

1.5 Thesis Outline

In **Chapter 2**, we will discuss the image denoising in details. We will introduce different types of noise, their effects on the images, our reason to consider a specific type, the different techniques available to denoise this specific type of noise, details of BM3D and improvements of BM3D in recent years.

In **Chapter 3**, we will discuss our approach to improve the performance of BM3D in details.

In **Chapter 4**, we will present the experimental results produced by our suggested approach and also compare these results with the results produced by BM3D.

Finally, In **Chapter 5**, we will summarize our works. Finally, we will present some possible future works.

Chapter 2

Background Studies

Random variation of image intensities can be defined as a noise in digital images. An image can be corrupted by noise in its acquisition and transmission. Mostly, the values acquired by image sensors are contaminated by noise because of imperfect instruments, problems with the data acquisition process and interfering natural phenomenas[30][2]. Figure 2.1 shows an example of a noisy image. The left image is the original image and the right image is the noisy image corrupted by Gaussian noise of standard deviation, σ=20.

Figure 2.1: Example of a noisy image

2.1 Image Noise

The noises which contaminate the image are random. Figure 2.2 will help us to understand the effect of noise on the true pixel value. We have plotted the pixel values of a specific row of both of the image from Figure 2.1. The left image is the plot of pixel values of row=10 from the original image and the right image is the plot of the same row from the noisy image. We

can observe that the pixel values were almost uniform when they were not corrupted by noise. When these pixel values are corrupted by Gaussian noise they took the shape of a Gaussian distribution.
In order to denoise, we need to approximate the noise mathematically. In order to approximate the noise, we choose different noise models for different noise cases.

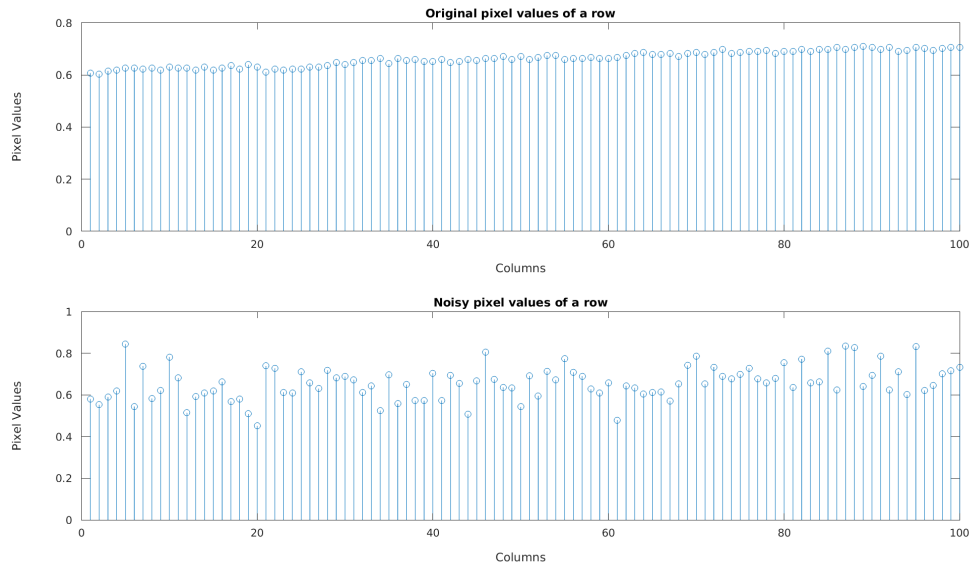


Figure 2.2: Effect of noise on pixel true value

2.2 Noise Types

We will divide the noise between two types based on the correlation of noise values with pixel values.

1. Spatially independent noise
2. Spatially dependent noise

In the following sections, we will discuss these two types of noise. We will briefly discuss another special type of noise named Impulse noise.

2.3 Spatially Independent Noises

If the noise is independent of spatial coordinates and has no correlation with pixel true value, then the noise is called spatially independent noise.

The following equation represents the spatially independent noise model,

$$z(i, j) = y(i, j) + \eta(i, j) \tag{2.1}$$

Here, i and j represents the pixel coordinates. $y(i, j)$ is the true pixel intensity. $\eta(i, j)$ is the spatially independent noise. $z(i, j)$ is the noisy image.

There are many types of spatially independent noises and we will discuss a few of them in the following sections.

2.4 Gaussian Noise

The Gaussian noise follows the Gaussian distribution. The "Amplifier Noise" is a major part of the "read noise" of an image sensor. Read noise is the amount of noise generated by electronics as the charge present in the pixels is transferred to the camera. This "Amplifier Noise" is modeled as Gaussian Noise.

Gaussian Noise is mostly used as a noise model for denoising because it is mathematically convenient.

The Probability Density Function (pdf) of Gaussian distibution [20] is as follows,

$$p(z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(z-\mu)^2}{2\sigma^2}} \tag{2.2}$$

Here, μ is mean and σ is the standard deviation of the distribution.

Figure 2.3 shows the Gaussian distribution pdf with μ=0 and σ=1.

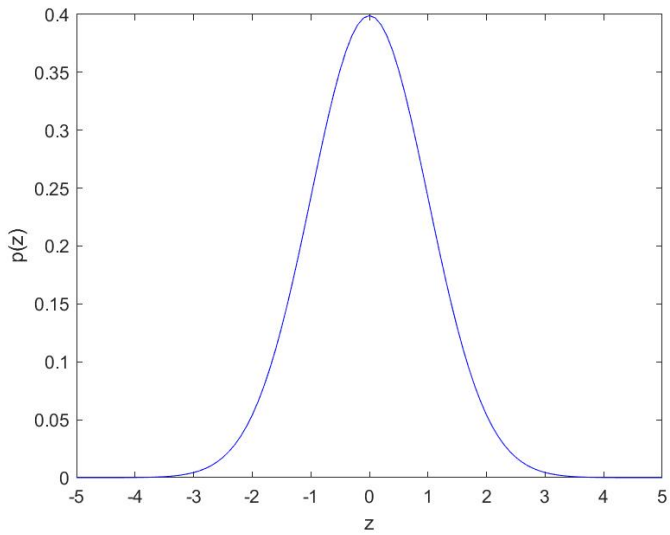


Figure 2.3: Gaussian Distribution pdf

The Normal distribution is a special case of Gaussian distribution where the mean is 0 and the standard deviation is 1. In Gaussian distribution, around 68% values will be within one standard deviation from the mean.

Figure 2.4 shows an example of a noisy image corrupted by Gaussian noise with mean 0 and standard deviation 10.

Figure 2.4: Example of Gaussian noisy image

2.5 Rayleigh Noise

Rayleigh Noise follows Rayleigh distribution. Signal values are almost zero in the background section of the MRI image. The noise generated at that portion can be modeled as Rayleigh

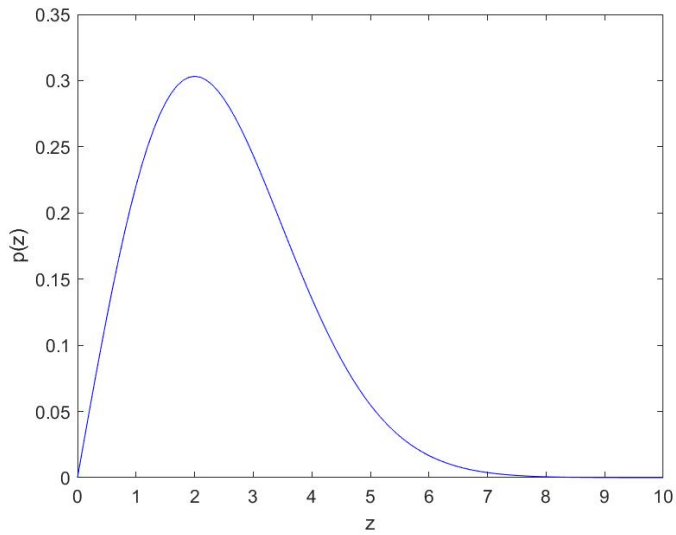


Figure 2.5: Rayleigh distribution pdf

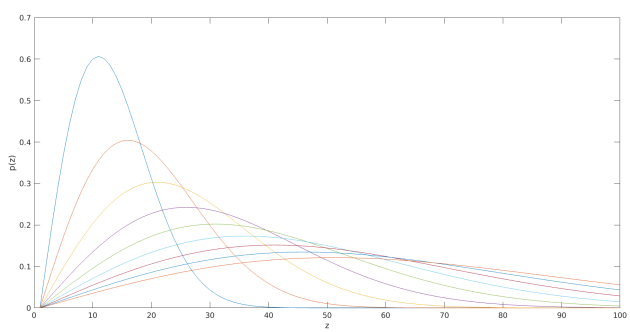


Figure 2.6: Effect of b on Rayleigh distribution

noise. The Pdf equation of the Rayleigh distribution is as follows [20],

$$p(z) = \begin{cases} \frac{2(z-a)}{b} e^{\frac{-(z-a)^2}{b}} & z \geq a \\ 0 & z < a \end{cases} \tag{2.3}$$

Here, b represents the spread of the distribution and a represents the shift from the origin.

Figure 2.5 shows the pdf of the Rayleigh distribution with b=2 and a=0.

The shape of the Rayleigh distribution is skewed to the right. It can be useful for approximating skewed histograms.

Figure 2.6 shows the effect of different values of b on the Rayleigh distribution.

In Figure 2.7, the left image is the actual cameraman image and the right image is the cameraman image corrupted by Rayleigh noise of a=0 and b=0.05.

Figure 2.7: Example of Rayleigh noisy image

2.6 Gamma Noise

Noises which occur in the laser-based images can be modeled as Gamma noise. The following is the equation of the pdf of Gamma noise,

$$p(z) = \begin{cases} \frac{a^b z^{(b-1)}}{(b-1)!} e^{(-az)} & z \geq 0 \\ 0 & z < 0 \end{cases} \tag{2.4}$$

Here, b is the shape parameter and a is the rate parameter.

Figure 2.8 shows the pdf of Gamma distribution with b=2 and a=0.

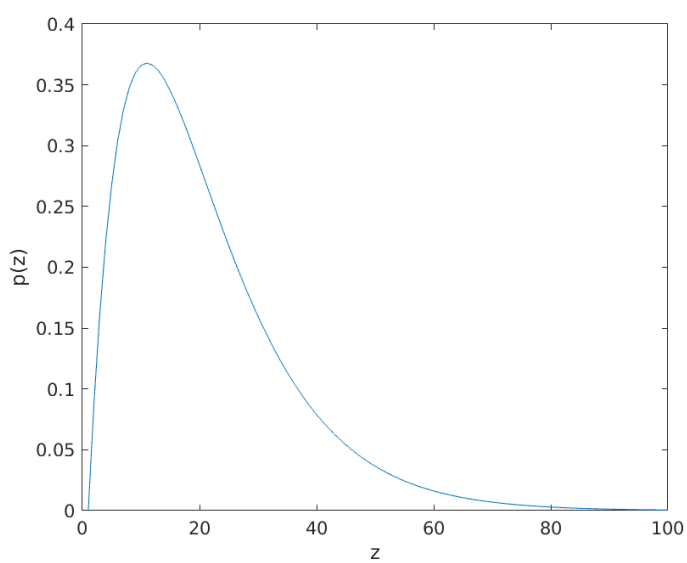


Figure 2.8: Gamma distribution pdf

Figure 2.9 shows the effect of the parameters on the distribution. The left image is the effect of various a values and the right image is the effect of various b values.

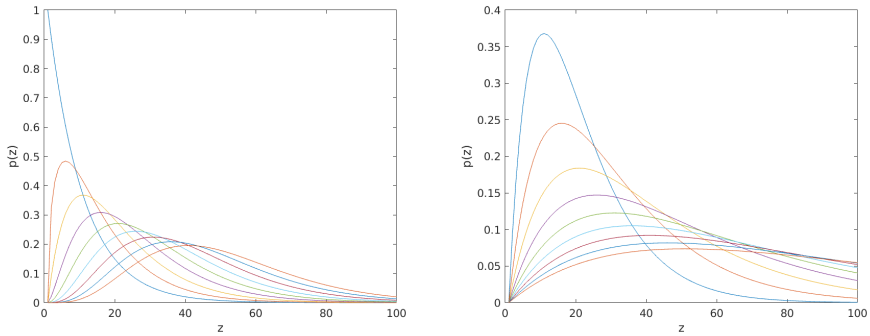


Figure 2.9: Effect of parameters on Gamma distribution

In Figure 2.10, the left image is the actual cameraman image and the right image is the cameraman image corrupted by Gamma noise with parameters $a = 5$ and $b = 5$.

Figure 2.10: Example of Gamma noisy image

2.7 Exponential Noise

In Channel-based communication, noises can be modeled as Exponential noise. Actually, the Exponential distribution is a variant of Gamma distribution where $b = 1$.

The following is the pdf equation of the Exponential noise[20],

$$p(z) = \begin{cases} ae^{-az} & z \geq 0 \\ 0 & z < 0 \end{cases} \tag{2.5}$$

Here, a is the rate parameter. It represents how quickly the Exponential pdf is decaying. Figure 2.11 shows the pdf of Exponential distribution with a=1.

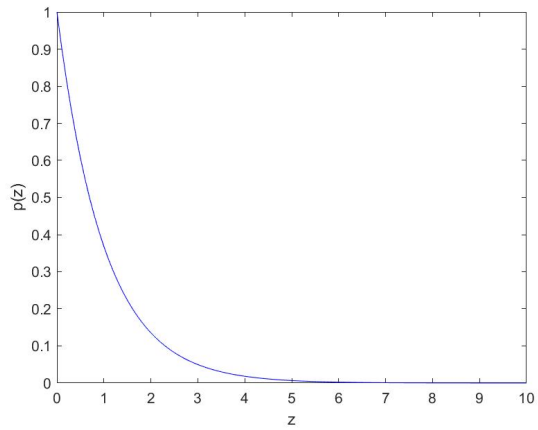


Figure 2.11: Exponential distribution pdf

In Figure 2.12, the left image is the actual cameraman image and the right image is the cameraman image corrupted by Exponential noise with parameter a = 0.1.

Figure 2.12: Example of Exponential noisy image

2.8 Uniform Noise

This noise model does not resemble any practical situation.

The pdf equation of the uniform noise is following[20],

$$p(z) = \begin{cases} \frac{1}{b-a} & b \leq z \geq a \\ 0 & otherwise \end{cases} \tag{2.6}$$

Here, all the values between a and b have an equal probability of occurring. The pdf of uniform noise is following,

Figure 2.13 shows the pdf of Exponential distribution with b=5 and a=2.

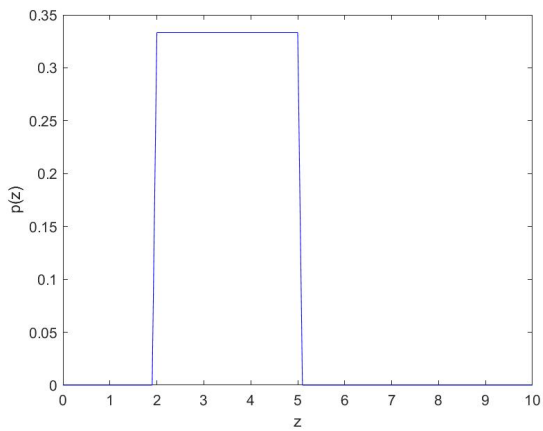


Figure 2.13: Uniform Distribution pdf

In Figure 2.14, the left image is the actual cameraman image and the right image is the cameraman image corrupted by uniform noise with parameter b = 40 and a=20.

Figure 2.14: Example of uniform noisy image

2.9 Additive White Gaussian Noise

When the noise has constant power spectral density, it is called as White noise[20]. Power spectral density shows how much power is contained in each of the spectral components. A White noise signal is constituted by a series of samples that are independent and generated from the same probability distribution. So, a White noise signal generated from a random

number generator in which all the samples follow a given Gaussian distribution is called White Gaussian noise.

2.10 Spatially Dependent Noise

This type of noise gets multiplied with the original signal. Spatially dependent noise has a correlation with a true pixel value. The spatially dependent noise model is following,

$$z(i, j) = y(i, j)\eta(i, j) \tag{2.7}$$

Here, y(i,j) is the pixel intensity and η(i,j) is the spatially dependent noise at (i, j) coordinate.$z(i, j)$ is the noisy pixel intensity. In the following section, we will discuss a common spatially dependent noise referred to speckle noise.

2.11 Speckle Noise

In almost all coherent systems, the noises can be modeled as speckle noise. Mainly, the source of this kind of noise is random interference between coherent returns. It follows Gamma distribution.

In Figure 2.15, the left image is the actual cameraman image and the right image is the cameraman image corrupted by speckle noise.

Figure 2.15: Example of speckle noisy image

2.12 Impulse Noise

The pdf equation of impulse noise is following[20],

$$p(z) = \begin{cases} P_a & z = a \\ P_b & z = b \\ 0 & otherwise \end{cases} \tag{2.8}$$

Figure 2.16 shows the pdf of impulse noise.

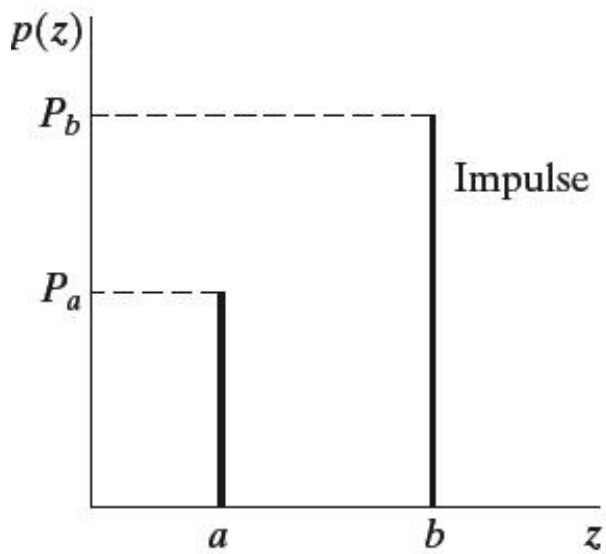


Figure 2.16: Impulse noise pdf

If $b > a$, gray-level b will appear as a light dot in the image. Conversely, level a will appear like a dark dot. If either P_a or P_b is zero, the impulse noise is called unipolar. Otherwise, it is called bipolar. If neither of them is zero and if they are approximately equal, impulse noise values will resemble salt-and-pepper granules randomly distributed over the image. For this reason, bipolar impulse noise is also called salt-and-pepper noise.

Noise impulses can be negative or positive. Impulse noises are very large compared to the image pixel intensity values, impulse noise is digitized as extreme(pure black or white) values in the image. Thus the assumption is that a and b is equal to the minimum and maximum values allowed in the digitized image. Negative impulses appear as a black dot and positive impulses appear as a white dot in the image. For, 8 bit image, $a = 0(black)$ and $b = 255(white)$.

In situations where quick transients, such as faulty switching, take place during imaging, noises can be modeled as impulse noise.

In Figure 2.17, the left image is the actual image and the right image is the image corrupted by impulse noise.

Figure 2.17: Example of impulse noisy image

2.13 Our Considered Noise

Noises of different types are different in nature. They also have different denoising techniques for each of them. For the rest of our discussion, we are going to consider the additive White Gaussian noise and some of the denoising techniques for the natural images which are corrupted by this noise. Our considered Noise has following properties,

- Additive: We are considering signal independent noise. Mathematically it will be added to the original signal or pixel true value.

- Independent and Identically distributed: The noise samples we are considering, are independent of each other. All of our considered noise samples will be drawn from the same distribution.

- Gaussian: According to central limit theorem, irrespective of underlying distribution of a population (with mean μ and standard deviation σ, if we take a number of samples of size N from the population, then the sample mean to follow a normal distribution with a mean of μ and a standard deviation of $\sigma/\sqrt{N}$ [28][16][15]. It signifies that irrespective of base distribution the probability distribution curve will approach Gaussian or normal distribution. In other words, the sum of independent and identically distributed random variables(with finite mean and variance) approaches normal distribution as sample size N tends to inf.

2.14 Image Denoising

In the previous section, we have discussed the random noises and how to model them. In this section, we will discuss various approaches to denoise noisy images.

There are two basic approaches to image denoising. They are following,

1. Spatial filtering methods
2. Transform domain filtering methods

We will discuss them in the following sections.

2.15 Spatial Filtering Methods

In the spatial filtering methods, we aim to denoise the noisy pixel values in the spatial domain. Spatial filters can be further classified into two following types,

1. Linear spatial filters
2. Non-linear spatial filters

We will discuss them in the following sections.

2.16 Linear Spatial Filters

In linear spatial filtering, the value of an output pixel is a linear combination of neighborhood values. There are various types of linear spatial filters. Mean filter is the most common one.

We will discuss the mean filter in the following section,

2.17 Arithmetic Mean Filter

The mean filter aims at finding an estimate which minimizes the mean square error between the noisy value and the respective estimate in the spatial domain. Let S_{xy} represent the set of coordinates in a rectangular sub-image window size of $m \times n$ centered at point (x, y). The Arithmetic Mean filtering process computes the average value of the corrupted image, $g(x, y)$ in the area defined by S_{xy}. The value of the restored image $\hat{f}$ at any point (x, y) is simply the arithmetic mean computed using the pixels in the region defined by S_{xy}. The equation is as follows[20],

$$\hat{f}(x,y) = \frac{1}{mn} \sum_{(s,t)\epsilon S_{xy}} g(s,t) \tag{2.9}$$

Mean filters actually smooth the local variations in the image. The Denoised image might be highly blurred.

Figure 2.18 shows an example of a denoised image by the Arithmetic Mean filtering.

(a) Actual Image (b) Noisy Image (c) Denoised Image

Figure 2.18: Example of denoised image by mean filter

2.18 Minimum Mean Square Error(Wiener) Filtering

The objective of this filter is to find an estimate,$\hat{f}$ of the uncorrupted image f such that the mean square error between them is minimized. This error measure is given by[20],

$$e^2 = E\{(f - \hat{f})^2\} \tag{2.10}$$

Here, $E\{.\}$ is the expected value of the argument. Wiener filter is used in the frequency domain. It is not a spatial filter but a linear filter. Let us assume, we have a noisy image, g(u,v) and we will denote the Discrete Fourier Transform(DFT) of the image as G(u,v). In the spatial domain, we usually convolute the filter kernel with the image to get the filtered output. In the frequency domain, the convolution operation becomes the multiplication operation. So, if we multiply the DFT of the Wiener filter with the noisy transformed image, we will get the filtered image. The following is the equation of DFT of the image estimated by Wiener filter[20],

$$\hat{F}(u,v) = \frac{H^*(u,v)S_f(u,v)}{S_f(u,v)|H(u,v)|^2 + S_\eta(u,v)} G(u,v) \tag{2.11}$$

Here, $H(u,v)$ is the DFT of the Degradation function or Point Spread Function,$H^*(u,v)$ is the complex conjugate of $H(u,v)$,$S_\eta(u,v)$ is the power spectrum of noise, $S_f(u,v)$ is the power

spectrum of the undegraded image. We will get the denoised image in the spatial domain by the inverse transform the DFT estimate.

Figure 2.19 shows an example of a Wiener filtered image.

(a) Actual Image (b) Noisy Image (c) Denoised Image

Figure 2.19: Example of Wiener filtered image

2.19 Non-linear Spatial Filter

One of the main problems associated with the linear spatial filters is that they blur the edges. Non-linear Spatial filters give better denoising result with less blurring for some of the noise models. The Median filter is the most common Non-linear spatial filter. We will discuss the Median filter in the following section.

2.20 Median Filter

The Median filter is one of the most popular order-statistics filters. Order-statistics filters are spatial filters whose response is based on ordering the pixels contained in the image area encompassed by the filter. The response of the filter at any point is determined by the ranking result. The Median filter replaces the value of a pixel by the median of the gray levels in the neighborhood of that pixel. We can understand it by the following equation[20],

$$\hat{f}(x, y) = \underset{(s,t)\epsilon S_{xy}}{median}\{g(s, t)\} \tag{2.12}$$

Median filters are particularly effective in the presence of both unipolar and bipolar impulse noise. Figure 2.20 shows an example of a Median filtered image.

(a) Actual Image (b) Noisy Image (c) Denoised Image

Figure 2.20: Example of Median filtered image

2.21 Adaptive Median Filter(ADM)

The performance of the Median filter discussed in the previous section degrades if the spatial density of impulse noise is high[20]. The Adaptive Median filter can perform well even if the density of impulse noise is high. Adaptive Median Filter (AMF) works in a rectangular window area, S_{xy}. Depending on the gray level values in the area, the algorithm changes the size of the window area. We will represent z_{min} as the minimum gray level value in S_{xy}, z_{max} as the maximum gray level value in S_{xy}, z_{med} as the median of gray levels in S_{xy}, z_{xy} as the gray level at coordinates (x, y), S_{max} as the maximum allowed size of S_{xy}.
We will briefly describe the algorithm here. AMF works in two levels, we will denote them as level A and level B. In level A, the algorithm searches for a region where z_{med} is greater than z_{min} and less than z_{max}. It keeps increasing search size as long as this condition does not meet. If the condition is satisfied in any region, the algorithm proceeds to level B. In level B, if z_{xy} is more than z_{min} and less than z_{max} in that region, then the denoised pixel value at (x, y) is z_{xy}, otherwise, the denoised value is z_{med}. If the algorithm could not proceed to level B and the search area becomes greater than S_{xy}, then the denoised pixel value at (x, y) is z_{xy}.

Figure 2.21 shows an example of ADM filtered image corrupted by Gaussian noise and Figure 2.22 shows an example of ADM filtered image corrupted by impulse noise.

(a) Actual Image (b) Noisy Image (c) Denoised Image

Figure 2.21: Example of ADM filtered image for Gaussian noisy image

(a) Actual Image (b) Noisy Image (c) Denoised Image

Figure 2.22: Example of ADM filtered image for impulse noisy image

2.22 Transform Domain Filtering Based Methods

In transform domain filtering methods, we will not filter in the spatial domain, rather we will transform the pixel intensity values into any other transform domain. We will choose transform domain such that in that domain, we can separate noise and signal values as better as we can. Then we will perform the filtering on the transformed coefficients so that the separated noise gets eliminated and we get a sparse representation. Then we will invert the transform to get the actual pixel values.

We will discuss two types of this kind of filtering. They are as following,

1. Spatial Frequency filtering

2. Wavelet domain filtering

2.23 Spatial Frequency Filtering

In this method, we will transform the image pixel values into the frequency domain using the Fast Fourier Transform(FFT). Then, we will use a low pass filtering to filter out the noise. We will choose a cut off frequency such that noises are decorrelated from the useful signal. Low pass filtering means that the filter will pass the low-frequency contents and block the high-frequency components. The low frequency and high frequency will be decided upon the choice of the cut-off frequency. The frequencies higher than the cut-off will be denoted as high frequency and lower than that will be denoted as low frequency. Generally, the low-frequency components of the image correspond to the uniform areas of the image and the high-frequency components correspond to the features(such as edges) and noises. Figure 2.23 shows us an example of the low-frequency component and high-frequency components. The right image is the FFT image of the left image. The bright values are the high valued coefficients and dark values are otherwise. The center portion of the image signifies the low-frequency component or uniform areas and the rest of the image contains high-frequency components.

Figure 2.23: Low and high frequency components

Then we will invert the Fourier transform on the transformed coefficients to get the spatial domain denoised image. The main disadvantage of this filtering is that the edge information spread across frequencies and it is also difficult to correctly separate out the edge and noise from the frequency spectrum because sometimes they share the same frequencies. So, in times of denoising, this filtering loses the edge information and as a result blurs the edges.

2.24 Wavelet Domain Filtering

In Wavelet domain filtering, We will use the Discrete Wavelet Transform(DWT). DWT decomposes the input image into different frequency subbands, labeled as LL_j, LH_k,HL_k, HH_k where, k=1,2,....,j and k indicates the kth resolution level of Wavelet transform and j is the largest resolution level in the decomposition. The Figure below shows the decomposition up to $3rd$ resolution level,

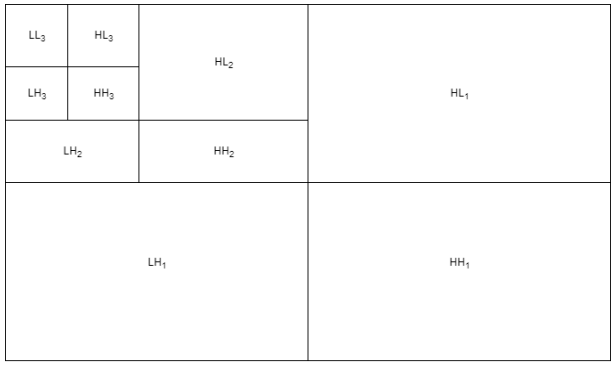


Figure 2.24: Decomposition upto 3rd level

The lowest frequency LL_j subband, obtained by low-pass filtering along with both directions, contains the approximation coefficients of the image signal, the LH_k, HL_k and HH_k contains the horizontal, vertical and diagonal coefficients at kth resolution level respectively. The LL_{k-1} subband will be further decomposed to get the kth level LL_k, HL_k, LH_k and HH_k.

The Figure below contains the actual cameraman image and noisy cameraman image corrupted by Gaussian noise of mean 0 and standard deviation 10.

(a) cameraman image (b) Noisy image

Figure 2.25: Example of an image corrupted by Gaussian noise of mean 0 and standard deviation 10

Figure 2.26 shows respectively approximation, vertical, horizontal, diagonal coefficients of the noisy cameraman image shown in the figure above.

(a) Approximation Coefficients

(b) Vertical Coefficients

(c) Horizontal Coefficients

(d) Diagonal Coefficients

Figure 2.26: Illustration of Wavelet coefficients

The advantage of DWT is that the signal energy will be concentrated in a small number of coefficients and the noise energy will be distributed in the whole domain. So, in the DWT of the noisy image, a small number of coefficients will have a high Signal to Noise Ratio(SNR) while a larger number of coefficients will have low SNR. We will remove the coefficients with low SNR as they will most likely be accounted to noise. Now, in order to eliminate noise we can apply three types of techniques[48], we will discuss only one of them denoted as Wavelet-based thresholding. Let us discuss this technique in details.

- Wavelet-based thresholding: It refers to modify the coefficients that are irrelevant relative to some threshold. However, this technique is highly dependent on the threshold estima-

tion. If the threshold is small then the noisy coefficients will still exist after thresholding and if it is large then important features might be removed. The thresholds available in literature can be divided into two types[24],

- Non-adaptive threshold estimation
- Adaptive threshold estimation

We will discuss only Non-adaptive threshold estimation here.

- Non-adaptive threshold estimation: Visushrink is a non-data adaptive threshold estimation criterion. We will discuss VisuShrink briefly. Visushrink is thresholding the Wavelet transform coefficients by applying universal threshold proposed by Donoho and Johnstone [14][35]. The VisuShrink threshold can be expressed by following equation,

$$t_d^F = \sigma\sqrt{2logM} \tag{2.13}$$

  and we will consider the noise variance, σ, as following,

$$\sigma = median(|W_c|)/0.6745 \tag{2.14}$$

  Where, t_d^F is the VisuShrink threshold, W_c is the Wavelet coefficients in HH_1 subband and M is the number of pixels in the image. However, in terms of denoising, if we threshold by VisuShrink, we get an overly smoothed image because this threshold tends to be high for large values of M. Because of being high, it removes many coefficients alongside the noise. This threshold does not perform well with the discontinuities of the image.

- Thresholding rule:

Now, we will discuss the ways to apply the thresholds. We will discuss two different ways to apply thresholds on DWT coefficients,

- Hard-thresholding: Hard-thresholding was proposed by Donoho[13]. In this rule, the coefficients that are less than or equal to the threshold will be zero and the remaining coefficient will be unchanged. This method creates artifact if the noise coefficients are moderately large. Hard-thresholding can be expressed by following expression,

$$D^H(d|\lambda) = \left\{ \begin{array}{ll} 0, & \text{for}|d| \leq \lambda \\ d, & \text{for}|d| > \lambda \end{array} \right\}$$

– Soft-thresholding: Soft-thresholding was also introduced by Donoho[12]. According to this rule, the coefficients larger than the threshold are reduced by the threshold value. We can represent the Soft-thresholding by the following expression,

$$D^S(d|\lambda) = \left\{ \begin{array}{ll} 0, & \text{for} |d| \leq \lambda \\ d - \lambda, & \text{for} d > \lambda \\ d + \lambda, & \text{for} d < -\lambda \end{array} \right\}$$

Soft-thresholding is also referred as Wavelet shrinkage, as coefficients which are larger than threshold are being reduced toward zero. On the other hand Hard-thresholding is either keep or remove the coefficients.

Figure 2.27 shows an example of a Wavelet denoised image. We have used Soft-thresholding and universal threshold.

(a) Actual Image (b) Noisy Image (c) Denoised Image

Figure 2.27: Example of a Wavelet denoised image

2.25 Edge Guided Image Denoising

Edges posses critical importance to the visual appearance of the image. At the same time as reducing noise, it is also important to preserve important features, such as edges, corners, and other sharp structures. We will discuss some methods which gave special importance to preserve these features. The Median filter is one of them which we discussed earlier.

2.26 Total Variation (TV) Based Filtering

If the signal possesses excessively high and spurious details, it is termed as the signal has high Total Variation. The integral of the absolute gradient of the signal is referred to as Total

Variation.

If $u_0(x, y)$ represents the noisy pixel value at (x, y), $u(x, y)$ is the desired image, $n(x, y)$ represents the additive noise then the following equation represents the considered model,

$$u_0(x, y) = u(x, y) + n(x, y) \tag{2.15}$$

If Ω represents the set which includes all the pixel of the image, then the following is the equation of the Median for the considered model,

$$\int_\Omega \sqrt{u_x^2 + u_y^2} dxdy \tag{2.16}$$

Excessively high and spurious details likely represent noise in the image. So, if we can minimize the Median, it is likely that noise will also be minimized.

In this method, the denoised image is produced by minimizing the Median norm of the estimated solution. A constrained minimization algorithm has been proposed [36][46] as a time-dependent nonlinear PDE, where the constraints are determined by the noise statistics.

The constraints are following,

- $$\int_\Omega dxdy = \int_\Omega u_0 dxdy \tag{2.17}$$

- $$\int_\Omega \frac{(u - u_0)^2}{2} dxdy = \sigma^2 \tag{2.18}$$

Here,σ represents the standard deviation of the noise $n(x, y)$.These constraints were imposed using Lagrange multiplier,λ.The constrained minimization problem is the following equation,

$$J(u) = \int_\Omega \sqrt{(u_x^2 + u_y^2)} dxdy + \frac{\lambda}{2} \int_\Omega (u - u_0)^2 dxdy \tag{2.19}$$

If we represent the number of iteration by n and the time step by Δt.The proposed iterative solution to this constrained optimization problem is following,

$$u^{n+1} = u^n + \Delta t(div(\frac{\Delta u}{|\Delta u}) - \lambda^n(u - u_0)) \tag{2.20}$$

2.27 Anisotropic Diffusion Filtering(ADF)

The idea [32][42][31] is to denoise the image and also preserve the edges. So, if we try to diffuse the image in the uniform region and limit the diffusion in the presence of an edge, highly likely, we can get better denoising. This is the idea of the anisotropic diffusion method. Mathematically, it is the diffusion equation with a variable term to limit smoothing at the edge. The term is a function of the gradient magnitude of the image at each pixel. The anisotropic diffusion equation is following,

$$I_t = div(c(|\Delta I|)\Delta I) \tag{2.21}$$

Here, $I_t = \delta I/\delta t$. Following function can be represented as the variable term or diffusivity parameter,

$$c(|\Delta I|) = \frac{1}{\sqrt{1 + \frac{|\Delta I|^2}{\lambda^2}}} \tag{2.22}$$

The function is monotonically decreasing. λ defines the threshold between the image gradients that are attributed as noise and that are attributed as an edge. if we denote n as each iteration step, the iterative formulation of the equation 1.21 is following,

$$I^{t+1} = I^t + div(c(\Delta I)\Delta) \tag{2.23}$$

The choice of λ and the number of iteration is of great importance. If the number of iteration is less than the denoising likely will not be good and if more then image will be over-smoothed.

In Figure 2.28 the left image is the actual image. We have added Gaussian noise of mean 0 and standard deviation 10 with this image.The right image is the ADF denoised image with number of iteration = 4, *lambda*= 0.1373, 0.1216,0.1098,0.1020 respectively for 4 successive iterations.

(a) cameraman image (b) ADF denoised image

Figure 2.28: Example of ADF denoised image

2.28 Non-local Means(NLM) Algorithm

Non-local means algorithm [5][4][6] attempts to utilize the redundancy of the natural image. Natural image likely has lots of instances of similar blocks. NLM denoised pixel value is the averaged value of all the pixels in the image when averaging, it imposes greater weight for those pixels who have similar neighborhood as the current pixel values neighborhood.

In order to explain mathematically, we will denote $y(i)$ as the observed noisy pixel value at index i, $x(i)$ is the original pixel value at index i, $\eta(i)$ is the independent and identically distributed (i.i.d) Gaussian noise with zero mean and variance σ_n^2. We will consider the following model,

$$y(i) = x(i) + \eta(i) \tag{2.24}$$

For the computational purpose, we will consider a search region rather than the whole image and will define it by S_t. If we want to denoise the pixel at index, i, then the denoised value will be calculated as the weighted average of all grey values within the search region. The denoised pixel value, $\hat{x}_{NLM}(i)$ can be found by the following equation,

$$\hat{x}_{NLM}(i) = \frac{\sum\limits_{j \epsilon S_t} w(i,j)y(j)}{\sum\limits_{j \epsilon S_t} w(i,j)} \tag{2.25}$$

The weight assigned for the pixel j is denoted by $w(i, j)$ in the above equation and can be found by the following equation,

$$w(i, j) = \frac{1}{Z(i)} e^{\frac{-\|N(i)-N(j)\|_{2,a}^2}{h^2}} \tag{2.26}$$

where $Z(i)$ is the normalizing constant and can be evaluated by the following equation,

$$Z(i) = \sum_j e^{\frac{-\|N(i)-N(j)\|_{2,a}^2}{h^2}} \tag{2.27}$$

Here, h is the smoothing parameter. $N(i)$ and $N(j)$ denotes a square neighborhood of size $P \times P$ centered on pixel i and j respectively. The vector norm used in the above equation is Gaussian weighted euclidean distance with standard deviation, a.

If we assign a small value for h, then the denoised pixel value will be almost as existing noisy value, otherwise, a large h value will produce an overly smoothed pixel value.

Figure 2.29 depicts the visual performance of the NLM algorithm. The left image is the original cameraman image. We have added Gaussian noise of mean 0 and standard deviation of value 10 with the image. The right image is the image denoised by NLM with $h = 0.0461$, $s = 21$ and $P = 5$.

(a) cameraman image (b) NLM denoised image

Figure 2.29: Example of NLM denoised image

2.29 Variants of Non-local Means Algorithm

During recent years, there were a lot of improvements on traditional Non-Local Means algorithm. As we have discussed in the previous section, in the original NLM, the search window size, S_t is fixed for all the pixels. If any pixel lies in a smooth region and small window size is applied, then the denoising will not likely be good enough and if the pixel lies in a non-

smooth region and we apply a larger search window then, we will likely lose features. So, if the search window size is adaptive with the region being considered, likely, we can achieve better denoising.

In this approach[41], this window size will be adaptive for each of the pixels. In the first step, we will utilize the traditional NLM and will get the denoised image. Let us consider it as the basic image. Then we will calculate the Gray Level Difference(GLD) image. In GLD image, the pixel value will be the absolute difference between the basic image pixel value and the mean value of the neighborhood centered on that pixel. In GLD image, the pixel value will be smaller if the region is smooth and larger otherwise. Then we will consider two thresholds and based on these thresholds we will divide the whole image in three regions namely as large, medium and small region. We will apply a larger window size for the pixels which fall into the large region, medium for medium region pixels and smaller for small region pixels. Finally, we will apply NLM with these adaptive search window sizes. This approach helps us preserving some image details and gives us slightly better results in terms of PSNR with the expense of using original NLM in the first step.

The original NLM algorithm uses a Gaussian weighted template with fixed weight coefficients for all the pixels in the neighborhood. This fixed weights can be susceptible to noise and when calculating the average, highly likely, that it will affect the denoising performance. So, if we can make this weight template adaptive with the noise level, we may get a better result.

In this approach[29], the Gaussian weight coefficient will be adjusted with the Laplace operator. In this approach at first, we will denoise the noisy image with traditional NLM. Let us call this denoised image as a basic image. In the basic image, if in any point still, noise exists then, the Laplace operator at that point will be of large intensity. Now, we will divide the neighborhood into some disjoint regions. We will calculate the weight of each region by utilizing the gradient and Laplace information. Now we will get a new weighting factor matrix by multiplying the original Gaussian weighting coefficient matrix with the weight matrix we calculated. Finally, we will apply NLM with this new weighting factor matrix. This method also shows better results in terms of both visually and PSNR and also preserves more textures and edges than original NLM with the expense of using original NLM in the first step.

2.30 Block Matching and 3D(BM3D) Filtering Based Denoising

At first, we will explain the algorithm stepwise in details and then we will summarise the algorithm in simple steps.

Now, let us consider the following image model,

$$z(x) = y(x) + \eta(x) \tag{2.28}$$

Here, x is the 2D spatial coordinate in the image. $z(x)$ is the noisy pixel value at x and $y(x)$ is the true pixel value at x and η is i.i.d zero mean Gaussian noise with variance σ^2.

Figure 2.30 shows the block diagram of the BM3D algorithm.

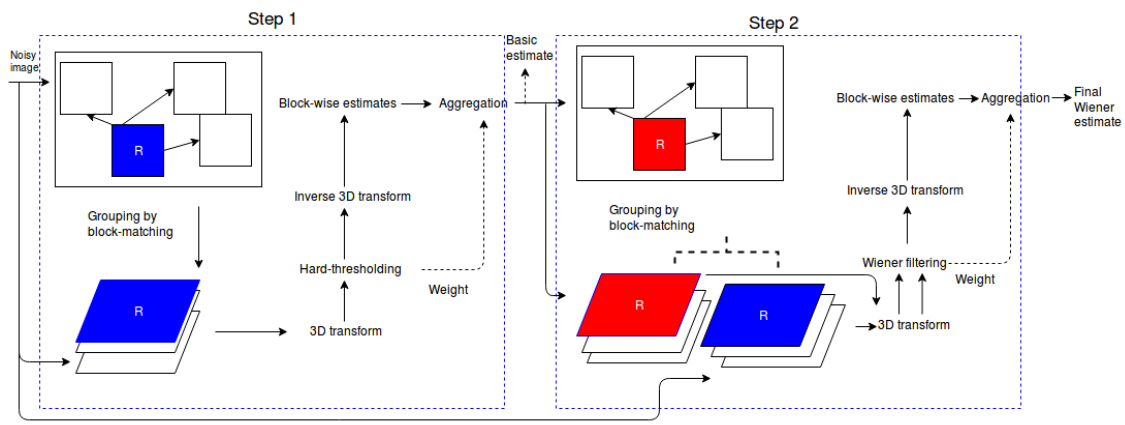


Figure 2.30: BM3D Block Diagram

We will use fragments, blocks, patches interchangeably. The algorithm works in two steps,

2.31 BM3D First Step

- *Group*ing and collaborative hard-thresholding:
  We want to utilize both of the intra-fragment correlation and inter-fragment correlation. We want to make a data structure which will be able to utilize both of these. We will denote this data structure as *group* and we will make this *group* for each of the fragments in the whole image. Now, in order to explain better, let us consider a current fragment of size $N_1^{ht} \times N_1^{ht}$ for which we want to make a *group*. We will consider each of the fragment of size $N_1^{ht} \times N_1^{ht}$ inside the search window, centered on the current patch and let us define each of them as a reference fragment. Now, we will measure the distance of each of the reference fragment with the current fragment. This distance measurement is called d-distance and can be expressed by the following equation,

$$d^{noisy}(Z_{xR}, Z_x) = \frac{\|Z_{xR} - Z_x\|_2^2}{(N_1^{ht})^2} \tag{2.29}$$

Here, Z_{xR} is the noisy reference block and Z_x is the current block. $\|.\|_2$ denotes as l^2-norm.

We will form a set of similar blocks for this current block by the following equation,

$$S_{xR}^{ht} = x \epsilon X : d(Z_{xR}, Z_x) \leq \tau_{match}^{ht} \tag{2.30}$$

Here, S_{xR}^{ht} is the set of all similar blocks for this current block and τ_{match}^{ht} is the maximum d-distance for which two blocks are considered similar. Now, we will stack all these similar blocks in this set alongside the current block and will get a 3D data structure which we will call as *group*.This step is Block Matching(BM). Figure 2.31 can well explain how we are *group*ing.

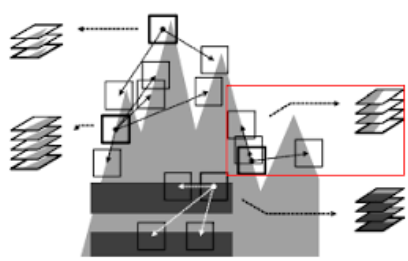


Figure 2.31: *group*ing and collaborative thresholding

Collaborative filtering is a very important step in this algorithm. We will choose a 3D transform or separable 2D+1D transform which is capable of utilizing the inter-fragment and intra-fragment correlation and provide us with sparse representation.
Sparse representation is another very important aspect of this algorithm. Sparse representation is to represent the data as coefficients such that most of the coefficient turns to almost zero or zero. Sparse representation helps to reduce redundancy and express correlation, we also want to detect the correlation between the pixels in the current fragment and also between the current and reference fragments. In this algorithm, after the 3D transform, we will hard-threshold the coefficients to get sparse representation. After the hard-thresholding, we will inverse the 3D transform to get the spatial values from the transformed coefficients. The process can be described by the following equation,

$$\hat{Y}_{S_{xR}^{ht}} = \tau_{3D}^{ht^{-1}}(\gamma(\tau_{3D}^{ht}(Z_{S_{xR}^{ht}})) \tag{2.31}$$

Here, γ is a hard-thresholding operator, τ_{3D}^{ht} is the 3D transform, $\tau_{3D}^{ht^{-1}}$ is the inverse 3D transform, $\hat{Y}_{S_{xR}^{ht}}$ is the 3D array of block-wise estimate we are getting after the collaborative filtering.

- Aggregation: Now as we are considering overlapping blocks, we will have multiple estimates for the same pixel. We will denote the final estimate of the first step as the basic estimate for each of the pixels. We will perform aggregation to get the basic estimate for each of the pixels. To find the basic estimate, $\hat{y}^{basic}$ for each of the pixel x, we will perform weighted average of the block-wise estimates $\hat{Y}_{S_{xR}^{ht}}$ using the weights w_{xR}^{ht}. If we can reduce the effect of the estimate which is coming from of noisier block at times of averaging, it is highly likely that we will get a better estimate. So, we will consider that a noisier block would have a larger sample variance. So, if we reward the estimate coming from the noisier block with low weight, likely, we will get a better estimate. So, we will consider our weight inverse proportionately to the total sample variance of the corresponding block-wise estimates. Our considered weight can be understood by the following equation,

$$w_{xR}^{ht} = \left\{ \begin{array}{ll} \frac{1}{\sigma^2 N_{har}^{xR}}, & \text{if } N_{har}^{xR} \geq 1 \\ 1, & \text{otherwise} \end{array} \right\}$$

Here, N_{har}^{xR} is the number of non-zero coefficients after hard-thresholding. We can calculate the basic estimate for each of the pixels by the following equation,

$$\hat{y}^{basic}(x) = \frac{\sum_{x_R \epsilon X} \sum_{x_m \epsilon S_{xR}^{ht}} w_{xR}^{ht} \hat{Y}_{x_m}^{ht,xR}}{\sum_{x_R \epsilon X} \sum_{x_m \epsilon S_{xR}^{ht}} w_{xR}^{ht} \chi_{x_m}(x)}, \forall x \epsilon X \tag{2.32}$$

Here, $\chi_{x_m} : X \rightarrow \{0, 1\}$ is the characteristic function of the square support of a block located at $x_m \epsilon X$, the blockwise estimates $\hat{Y}_{x_m}^{ht,xR}$ are zero outside their support.

In the figure below, the left image is the actual cameraman image, the middle image is the noisy cameraman image corrupted by Gaussian noise of mean 0 and standard deviation 10 and the right image is the denoised image after the first step.

(a) cameraman image (b) Noisy image (c) Denoised image

2.32 BM3D Second Step

After performing the first step, we have a basic estimate for all the pixels. We will denote this image as the basic image.

- *group*ing and collaborative filtering: We will form the *group* of similar blocks in the same as we have done in the previous step and can be expressed by following,

$$S_{xR}^{wie} = \frac{||\hat{Y}_{xR}^{basic} - \hat{Y}_{x}^{basic}||_2^2}{(N_1^{wie})^2} < \tau_{match}^{wie} \forall x \epsilon X \tag{2.33}$$

  Here, $\hat{Y}_{x}^{basic}$ is the considered current block of size $N_1^{wie} \times N_1^{wie}$ from basic image, $\hat{Y}_{xR}^{basic}$ is the reference block of same size ,S_{xR}^{wie} is the set of all blocks which has normalized squared l^2 distance between $\hat{Y}_{xR}^{basic}$ and $\hat{Y}_{x}^{basic}$ less than threshold, τ_{match}^{wie}. Now, according to the set, S_{xR}^{wie} we will form two *group*s respectively from the basic image and noisy image and denote them as following,

  - $\hat{Y}_{S_{xR}^{wie}}^{basic}$ by stacking together the basic estimate blocks $\hat{Y}_{x \epsilon S_{xR}^{wie}}^{basic}$.
  - $Z_{S_{xR}^{wie}}$ by stacking together the basic estimate blocks $Z_{x \epsilon S_{xR}^{wie}}$.

  Let us define the Wiener shrinkage coefficients as following,

$$W_{S_{xR}^{wien}} = \frac{|\tau_{3D}^{wie}(\hat{Y}_{S_{xR}}^{basic})|^2}{|\tau_{3D}^{wie}(\hat{Y}_{S_{xR}}^{basic})|^2 + \sigma^2} \tag{2.34}$$

  Here, $W_{S_{xR}^{wien}}$ is the Wiener shrinkage coefficients for the set S_{xR}^{wien}. We will perform the collaborative Wiener filtering of $Z_{S_{xR}^{wie}}$ by element-by-element multiplication of the 3D transform coefficients $\tau_{3D}^{wie}(Z_{S_{xR}^{wie}})$ of the noisy data with the Wiener shrinkage coefficients

$W_{S_{xR}^{wie}}$. We will inverse transform to get actual coefficients from the transformed coefficients by following equation, $\tau_{3D}^{wie}(\hat{Y}_{S_{xR}}^{basic})$ consists the 3D transform coefficients of $\hat{Y}_{S_{xR}}^{basic}$.

$$\hat{Y}_{S_{xR}^{wie}}^{wie} = \tau_{3D}^{wie^{-1}}(W_{S_{xR}^{wie}}\tau_{3D}^{wie}(Z_{S_{xR}^{wie}})) \tag{2.35}$$

Now, $\hat{Y}_{S_{xR}^{wie}}^{wie}$ is the *group* of block-wise estimates.

- Aggregation: As we have considered overlapping blocks, we might have multiple estimates for the same pixel. We can find the final estimate using the same procedure as we have used in the aggregation portion of the first step. For this step, the weight we will use is following,

$$w_{xR}^{wie} = \sigma^{-2}\|W_{S_{xR}^{wie}}\|_2^{-2} \tag{2.36}$$

Now, We will get the final estimate, $\hat{y}^{final}$ by a weighted average of the block-wise estimates, $\hat{Y}_{S_{xR}^{wie}}^{wie}$ using the weights described in the above equation for each of the pixels.

In the figure below, the left image is the actual cameraman image and the middle image is the noisy cameraman image corrupted by Gaussian noise of mean 0 and standard deviation of 10 and the right image is the final denoised image by BM3D algorithm.

(a) cameraman image (b) Noisy image (c) Denoised image

Now, we will summarise the whole algorithm in simple steps,

1. Step 1. Getting the basic image
   (a) Block-wise estimates:
      i. *group*ing:
         A. Find the blocks which are similar to the currently processing block
         B. stack them alongside the current processing one to form *group*.

ii. Collaborative hard-thresholding:

A. Apply a 3D transform on the *group*.

B. In order to attenuate the noise, apply hard-thresholding on the transform coefficients.

C. To produce estimates for all *group*ed blocks, invert the transform

D. Return the estimates of the blocks to their original positions.

(b) Aggregation: Compute the basic estimate for all the pixels by weighted averaging all of the obtained block-wise estimates that are overlapping.

2. Step 2. Final estimate: Use the basic image to perform *group*ing and collaborative Wiener filtering

(a) Block-wise estimates:

i. *group*ing:

A. Find the location of blocks that are similar to the current processing block.

B. Form two *group*s using these locations, one from the basic image and one from the noisy image.

ii. Collaborative Wiener filtering:

A. Apply a 3D transform on both *group*s.

B. Using the energy spectrum of the basic image as true energy spectrum for Wiener filter, perform Wiener filtering on the noisy *group*.

C. Apply inverse 3D transform on the filtered coefficients to produce final estimates.

D. Return the estimates of the blocks to their original positions.

(b) Aggregation: Compute the final estimate for all the pixels by weighted averaging all of the obtained block-wise estimates that are overlapping.

We will discuss the limitations of BM3D and some recent approaches to improve performance in the next section.

2.33 Improvements of BM3D

Although BM3D shows better denoising performance than most of the denoising methods in literature[2] in most of the image scenarios, some further improvement happened in recent

years. The authors Dabov et al.[7] who proposed BM3D extended their algorithm for color images [10].

Discrete cosine transform(DCT)[39][1] is representing the image in terms of sum of different cosine functions of different frequencies. Two-dimensional separable DCT possesses very good energy compaction properties and is a very efficient transform in order to achieve a sparse representation. However, if there are edges and singularities, then sparsity performance degrades[18]. So,the authors here [18] proposed to use the shape-adaptive dct(SA-DCT)[37][38] to process the arbitrarily shaped image segments where the underlying signal is uniform. SA-DCT has adaptive support and computed by cascaded application of varying-length DCT transforms on the columns and on the rows respectively for the region. The authors have utilized the Anisotropic Local Polynomial Approximation(LPA)-Intersection of Confidence Intervals(ICI) [26][25][19] technique to decide the adaptive support. In this technique, a varying-scale family of directional-LPA convolution kernels is used to produce a set of directional varying scale estimates for every specified direction. These estimates are then compared by ICI rule to get an adaptive scale for every direction. These estimates altogether in an adaptive convex combination provides us with the final anisotropic LPA-ICI estimate. Although this work itself is a separate denoising algorithm we are discussing it here because the authors later have used this approach with BM3D[8]. In this work, authors have decided the adaptive support for the transform with LPA-ICI technique and used the SA-DCT as the 3D transform of BM3D in both of the steps.

In order to improve the sparsity more, the authors proposed PCA on the adaptive shape neighborhood decided by LPA-ICI technique in this paper [9] as a part of the employed 3-D transform. From *group*s of similar adaptive-shape neighborhoods, an empirical second-moment matrix is formed and utilizing the eigenvalue decomposition of that matrix the PCA bases are formed. Depending on a threshold, in some cases, the SA-DCT is used and for some cases, the above mentioned PCA is used.

As the performance of BM3D degrades when the standard deviation of noise reaches 40, in this paper [11] the authors proposed to combine tetrolet prefiltering with BM3D when the noise is strong. They have proposed to use tetrolet transform in the first step of BM3D to remove part of the noise before Wiener filtering. Tetrolets are adaptive haar-type Wavelet filter bank whose supports are the shapes called tetrominoes[27].

In this paper [17], after decomposing the bm3d restored image into three orthogonal subbands, authors have observed that in the subband of the high-frequency component where the Wavelet coefficients are less than a given threshold, some textures of the original image are lost during denoising. Because of this loss, the Peak Signal to Noise Ratio(PSNR) in that band becomes less and as a result, the PSNR of the whole denoised image is becoming less.

, In order to increase the PSNR in this subband authors, have utilized directed diffusion. Directed diffusion equation is proposed by illner et.al [23]. Directed diffusion equation aims at finding an intermediate approximation between the noisy image and the initial approximation image of the original image[46]. The initial approximation of the original image is formed from all the known prior information. Directed Diffusion can be effective in protecting edges [45]. The final processed image maintains lots of known information from the approximation image but also deviates less from the noisy image [40].

Authors have used the anisotropic diffusion operator instead of the isotropic diffusion operator and used two different coefficients in this case. Utilizing the directed diffusion authors have diffused the estimated high-frequency subband image which is less than the threshold to the noisy version of it in order to get an intermediate estimate between them. This approach produced almost the same result as BM3D if the considering image has fewer edges and more uniform areas and if the noise level is smaller but better results if otherwise in terms of PSNR.

The performance of BM3D degrades significantly when the noise standard deviation reaches 40[22]. The authors of BM3D pointed the erroneous *group*ing as the reason and proposed to use coarse prefiltering before measuring the block-distance. Coarse prefiltering is applying a 2D linear transform on both blocks and then hard-thresholding the coefficients and then measuring the d-distance. The authors have also proposed to use the 2D DCT transform instead of 2D-Bior1.5 Wavelet transform and increasing the block size from 8 to 12. However, these approach results in producing more artifacts and increasing time complexity. In this paper [22] the authors proposed to increase the maximum number of blocks allowed in the *group*, increase the threshold. not using the DCT transform and decrease the block size. The authors have argued that the maximum allowed number of blocks in the *group* should be more when the noise is of the higher standard deviation to make sure that there are enough blocks to improve denoising.

The Wiener filter can be considered as the core of BM3D [21]. In this approach, [21] the authors have used the Structural Similarity Measure[43] as the objective function of the Wiener filter. In the original BM3D algorithm the objective function of the Wiener filter is Mean Square Error(MSE). In this paper, the authors have also shown that using an MSE optimized Wiener filter can provide us with misleading visual results. Furthermore, they have also proposed 3D Zigzag thresholding for the dc-only profile. If the 3D-DCT is used as 3D transform the authors have termed this as the dc-only profile of BM3D. 3D zigzag thresholding is using little or no thresholding for on the DC transform coefficients and first few AC coefficients and increasingly higher thresholding for the rest of AC coefficients.

In original BM3D hard-thresholding is used for thresholding which is fixed. So, if we use a threshold which will be adaptive we might get some improvements. In this paper[33] the

authors have proposed a different way of thresholding which will be adaptive to the sparsity level of Wavelet coefficients. In [33] the authors have shown that when the noise power is large, block matching becomes unreliable. Because of the incorrect choice of similar blocks, Wavelet transform coefficients are less sparse and loss of original information happens. So, they proposed to use hard-thresholding if the sparsity level is better and use Smooth Sigmoid-Based Shrinkage(SSBS) function[3] otherwise. When the sparsity level is better, the higher and lower coefficients are well separated, we can use hard-thresholding. But when the higher and lower components are not well separated, SSBS function provides us a way to control independently the amount of attenuation for small, medium and large Wavelet coefficients.

As we have described earlier the original BM3D algorithm utilizes hard-thresholding, in this paper [34], the authors have proposed to use a modified soft-thresholding and a Wavelet decomposition scale-dependent threshold. The Wavelet transform can concentrate the signal energy in a few large coefficients and noise energy is distributed throughout the Wavelet domain. So, the larger coefficients of the Wavelet transform likely consists of almost all of signal and smaller coefficients are likely consists of noise. So, hard-thresholding gets rid of all those smaller coefficients. But, the larger coefficients also have noises inside them. So, if we use the soft-thresholding then that noise inside those larger coefficients can likely be reduced. But, the soft-thresholding provides us with a fixed bias with respect to both Wavelet coefficient value and Wavelet decomposition level. Now, if we reduce the larger coefficients by this fixed amount, we might lose some signals as well. So, in order to improve the quality of thresholding, the authors have proposed a new thresholding rule which has an adaptive bias and also proposed a new scale-dependent threshold.

In order to better utilize the local sparsity of Wavelet coefficients and non-local similarity of *group*ed blocks, in this approach [47], the authors have removed the 1D transform used in the original BM3D algorithm. They have introduced the nonlocal centralization prior. Three nonlocal shrinkage functions have been proposed with different norms.

In this approach [44] the authors proposed a new convolutional neural network which follows the same computation pipeline as BM3D AND termed as BM3D-Net. The BM3D-Net consists of five layers: extraction layer, convolution layer, nonlinear transform layer, convolution layer, aggregation layer. The extraction layer corresponds to the block matching step, the next convolution layer, nonlinear transform layer, and convolution layer correspond to the 3D transform and thresholding of BM3D and aggregation layer implements the aggregation step of BM3D.

Chapter 3

Methodology

In this chapter, we will discuss our proposed method to improve the performance of BM3D in details and some experimental results based on our idea.

3.1 3D processing

3D processing plays a significant part in the denoising performance of BM3D. One of the peculiarities of natural images is the existence of mutually similar patches throughout the image. BM3D algorithm relies heavily on two types of correlation[7]:

- Intra-patch correlation: Intra-patch correlation means the correlation which can be found between the pixels of each patch.
- Inter-patch correlation: Inter-patch correlation means the correlation which appears between corresponding pixels of two patches.

To exploit these two types of correlation, BM3D algorithm calculates the d-distance between every two patches in the neighborhood to find out the mutually similar patches. To explain this procedure in details, we will consider a similar model as we described in Section 2.30.

The equation below represents our considered model,

$$Z(x) = Y(x) + \eta(x) \tag{3.1}$$

Here, x is the 2D spatial coordinate in the image, $Z(x)$ is the noisy pixel value at location x, $Y(x)$ is the true pixel value at location x and η is i.i.d. (independent and identically distributed) zero mean Gaussian noise with variance σ^2.

BM3D aims at utilizing both intra-patch and inter-patch correlations. BM3D makes a data structure which will be able to utilize both of these. This data structure is termed as *group*

and BM3D makes a *group* for each of the patches in the whole image. Now, to explain better, let us consider a current patch of size $N_1^{ht} \times N_1^{ht}$ for which we want to make a *group*. BM3D will consider each of the patches of size $N_1^{ht} \times N_1^{ht}$ inside a predefined search window, centered at the current processing patch and let us define each of them as a reference patch. Now, the distance between the current processing patch and each reference patch is calculated. This distance measurement is termed as d-distance and can be expressed by the following equation,

$$\text{d-distance}(Z_{xR}, Z_x) = \frac{\|Z_{xR} - Z_x\|_2^2}{(N_1^{ht})^2} \tag{3.2}$$

Here, Z_{xR} is the noisy reference patch, Z_x is the current processing patch, d-distance(Z_{xR}, Z_x) is the l^2-norm.

BM3D forms a set of similar patches for this current processing patch by the following equation,

$$S_{xR}^{ht} = x \epsilon X : \text{d-distance}(Z_{xR}, Z_x) \leq \tau_{match}^{ht} \tag{3.3}$$

Here, S_{xR}^{ht} is the set of all similar patches for this current patch and τ_{match}^{ht} is the maximum d-distance for which two patches will be considered similar. Now, the algorithm stacks all these similar patches in this set alongside the current processing patch and forms a 3D data structure which is termed as *group*.

Now, after forming the 3D data structure BM3D chooses a suitable separable 3D transform (2D+1D) to perform on this data structure. The chosen transform is capable of exploiting intra-patch correlation and inter-patch correlation.

$$\tilde{Y}_{S_{xR}^{ht}} = \tau_{3D}^{ht} Z_{S_{xR}^{ht}} \tag{3.4}$$

Here, τ_{3D}^{ht} is the 3D transform, $\tilde{Y}_{S_{xR}^{ht}}$ is the 3D array of patch-wise estimates we are getting after the transform

3.2 Our suggested method

As we mentioned in Section 3.1, BM3D algorithm focuses on exploiting the intra-patch and inter-patch correlations. The BM3D algorithm can completely exploit these correlations if it can find enough mutually similar patches. If BM3D can not find enough similar patches, it would not be able to perform to its full potential.

We consider that if the BM3D scheme can not find eight or more similar patches for a current processing patch, this current processing patch will not have enough mutually similar

patches to exploit the inter-patch correlation. Hence, if we combine these patches to form a 3D data structure and perform 3D transform, it is highly likely that it would not produce a good estimation.

Therefore, we suggest that if BM3D algorithm can not find eight or more similar patches for a current processing patch, we will perform a 2D transform rather than a 3D transform; we will perform a 2D Discrete Wavelet transform on the current processing patch rather than a 3D transform.

We will consider the same model we have described in Section 3.1. We will measure the d-distance of all the reference patches from the current processing patch for each of the current processing patch using Equation 3.2. We will form a set of similar patches for each of the current processing patch. We will consider a reference patch as a similar patch for a current processing patch if the d-distance between them is less than a predefined threshold,τ^{ht}_{match}.

The number of similar patches for each current processing patch is determined. If the number of similar patches is less than eight, we will not make a 3D data structure for the patch instead, we will perform a 2D transform on these patches. Otherwise, we will make a 3D data structure for the current processing patch by stacking the similar patches and the current processing patch altogether and perform 3D transform similar to the BM3D algorithm.

If we consider the patch as Z_x, τ^{ht}_{2D} as the 2D transform and $\tilde{Y}_x^{\,ht}$ as the estimate which we will get after the 2D transform then, the equation below represents the procedure,

$$\tilde{Y}_x^{\,ht} = \tau^{ht}_{2D}(Z_x) \tag{3.5}$$

Then, we will threshold the transformed coefficients for all the transformed patches (2D and 3D) and inverse the transform to get the estimates. Then we will aggregate these estimates to calculate the basic estimate for each pixel. The aggregation procedure can be represented by the equation 2.32. This will give us the denoised image after the first step.

After getting the final images from the first step, we will perform the second step of the BM3D scheme as described in Section 2.32. Lastly, we will get the final denoised image.

We will summarise our proposed method in simple steps below,

1. We will take the noisy image as input.
2. Then we will perform the d-distance measurement to find out the number of similar patches for each of the image patches.
    - If the number of similar patches is less than eight:
        - we will perform a 2D transform on the current processing patch.
        - we will threshold the transformed coefficients and inverse the 2D transform.

- If the number of similar patches is equal to or more than eight:
  - we will perform a 3D transform on the group.
  - we will threshold the coefficients and inverse the 3D transform.

3. We will aggregate all the estimates for each of the pixels and calculate the basic estimate.

4. Then we will perform the second step of BM3D and eventually, we will get our final denoised image.

Figure 3.2, Figure 3.3, Figure 3.4 and Figure 3.5 show four examples demonstrating the performance of 2D and 3D transformation when having less than eight similar patches. In these images, we marked the region by a blue square so that we can recognize those patches.

Figure 3.2, Figure 3.3, Figure 3.4, Figure 3.5 visually showed that the denoising of these patches becomes better if we perform 2D transform instead of 3D transform. In the next chapter, we will show detailed experimental results based on our suggested method.

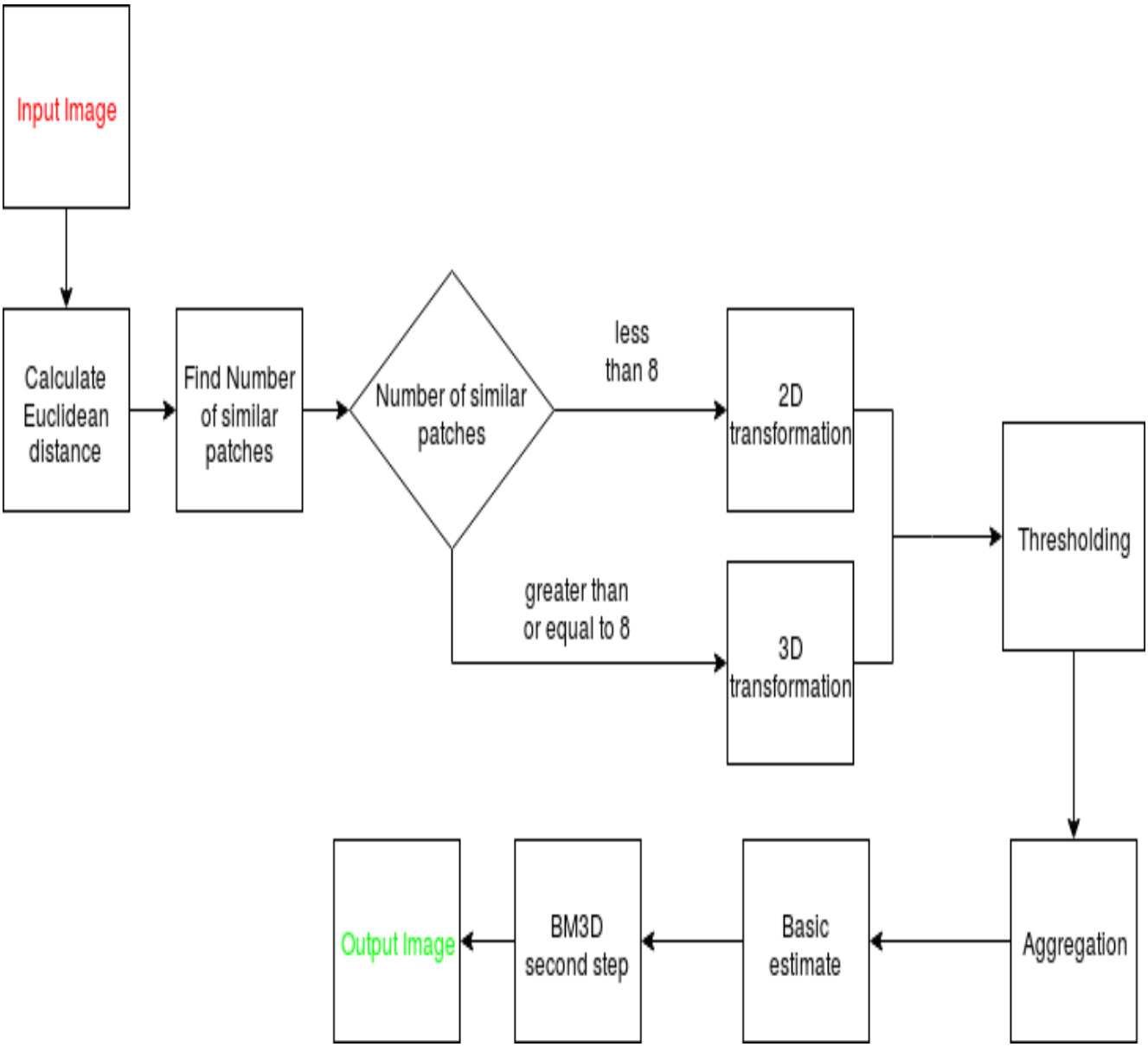


Figure 3.1: Block Diagram of our suggested method

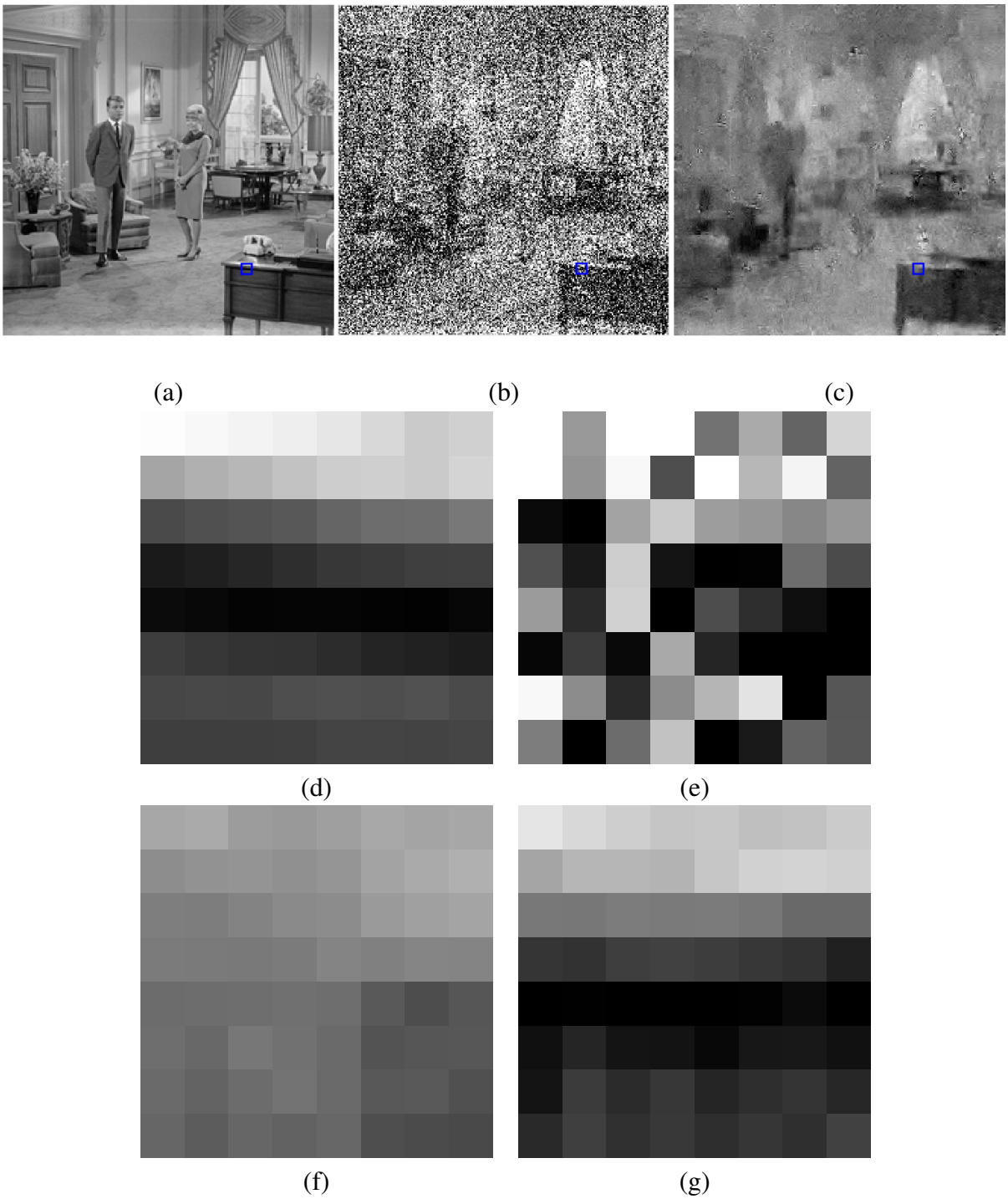


Figure 3.2: (a) Original image (b) Noisy image corrupted by gaussian noise of level 100 (c) Denoised image produced by BM3D (d) Original patch located at row 201 and column 185 (e) Noisy patch (f) the patch after 3D transform (g) the patch after 2D transform; we have marked the patch by blue square in the corresponding images

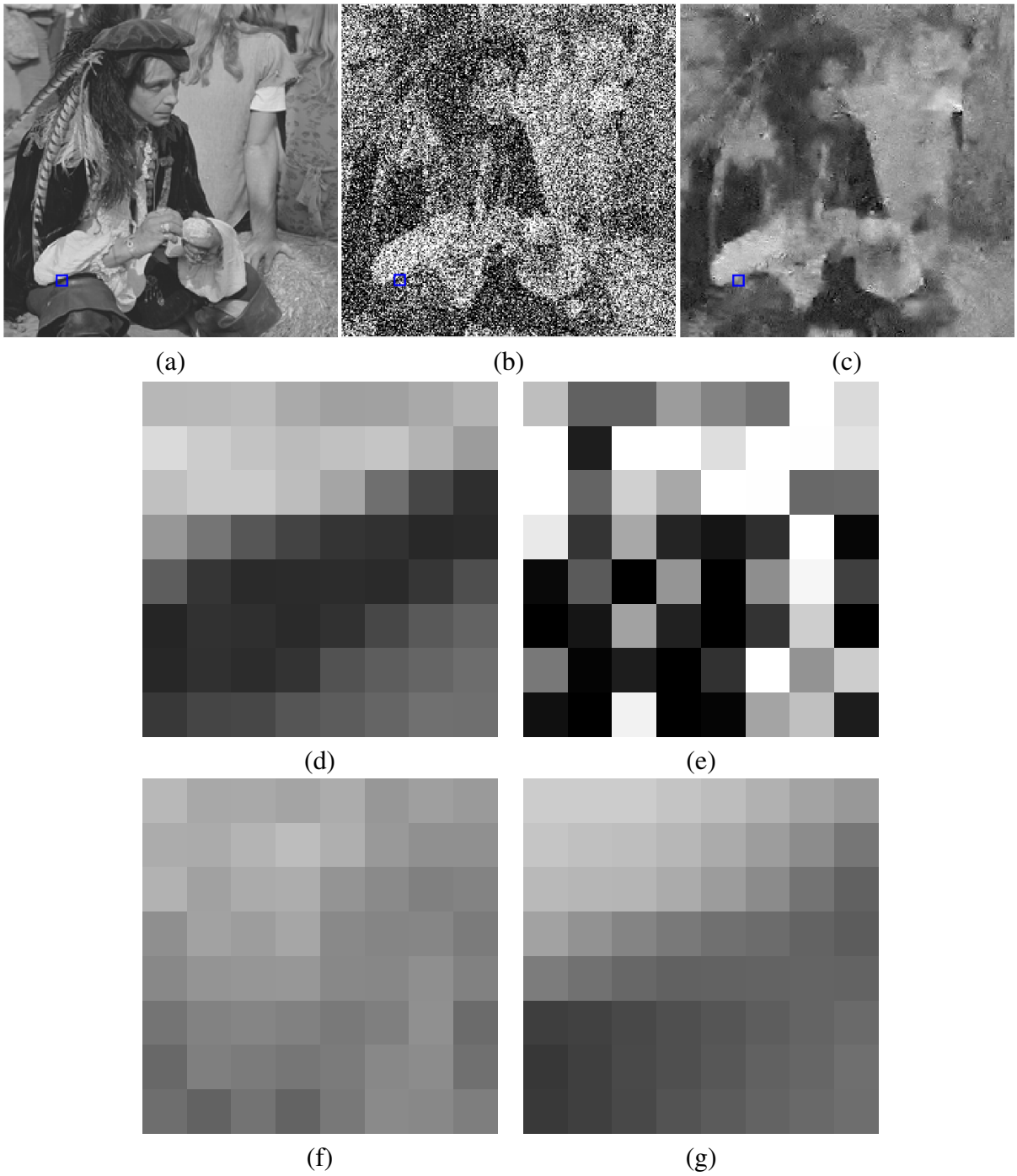


Figure 3.3: (a) Original image (b) Noisy image corrupted by gaussian noise of level 100 (c) Denoised image produced by BM3D (d) Original patch located at row 209 and column 41(e) Noisy patch (f) the patch after 3D transform (g) the patch after 2D transform; we have marked the patch by blue square in corresponding images

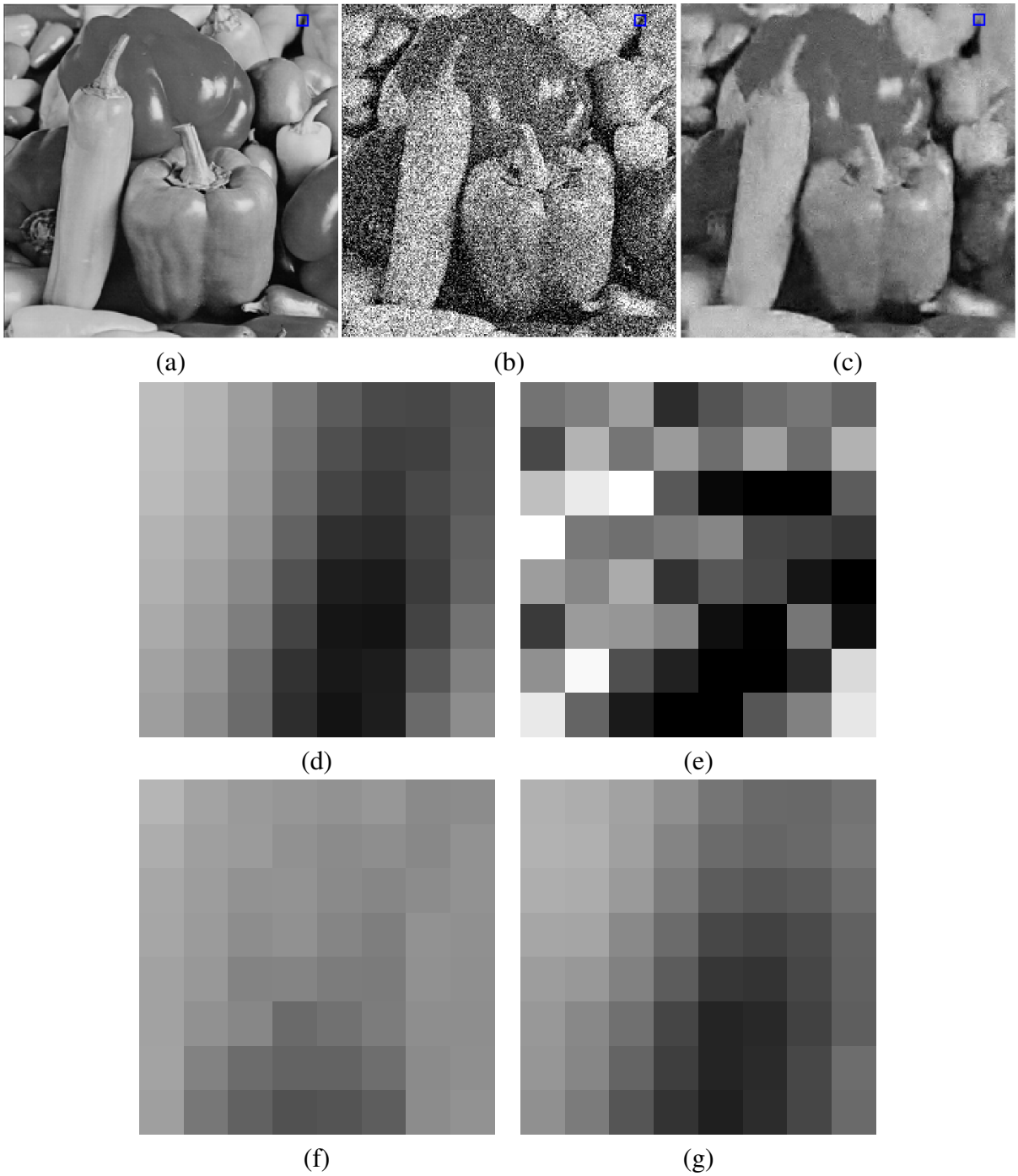


Figure 3.4: (a) Original image (b) Noisy image corrupted by gaussian noise of level 60 (c) Denoised image produced by BM3D (d) Original patch located at row 9 and column 225 (e) Noisy patch (f) the patch after 3D transform (g) the patch after 2D transform; we have marked the patch by blue square in corresponding images

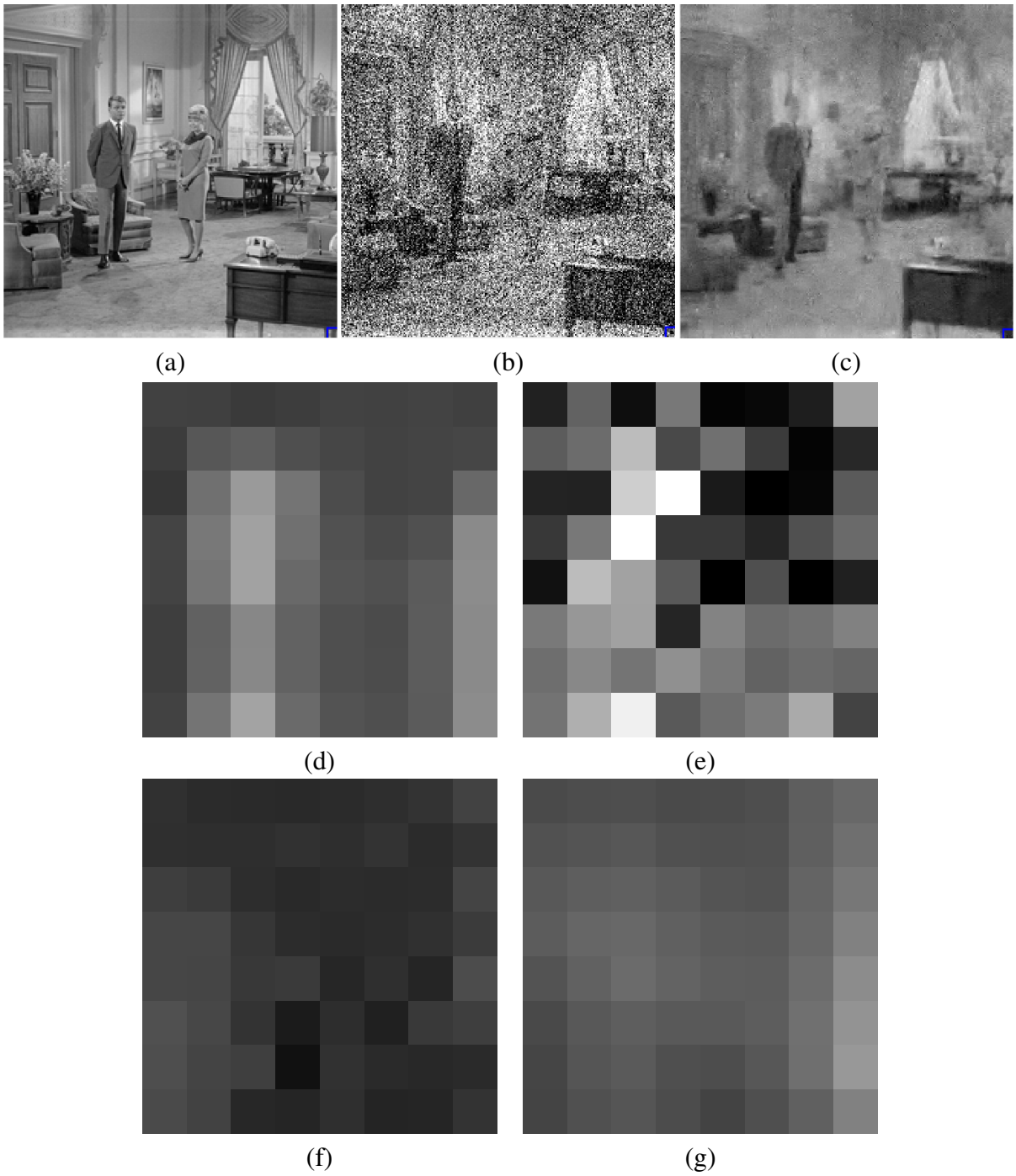


Figure 3.5: (a) Original image (b) Noisy image corrupted by gaussian noise of level 100 (c) Denoised image produced by BM3D (d) Original patch located at row 249 and column 249 (e) Noisy patch (f) the patch after 3D transform (g) the patch after 2D transform; we have marked the patch by blue square in corresponding images

Chapter 4

Experimental Results

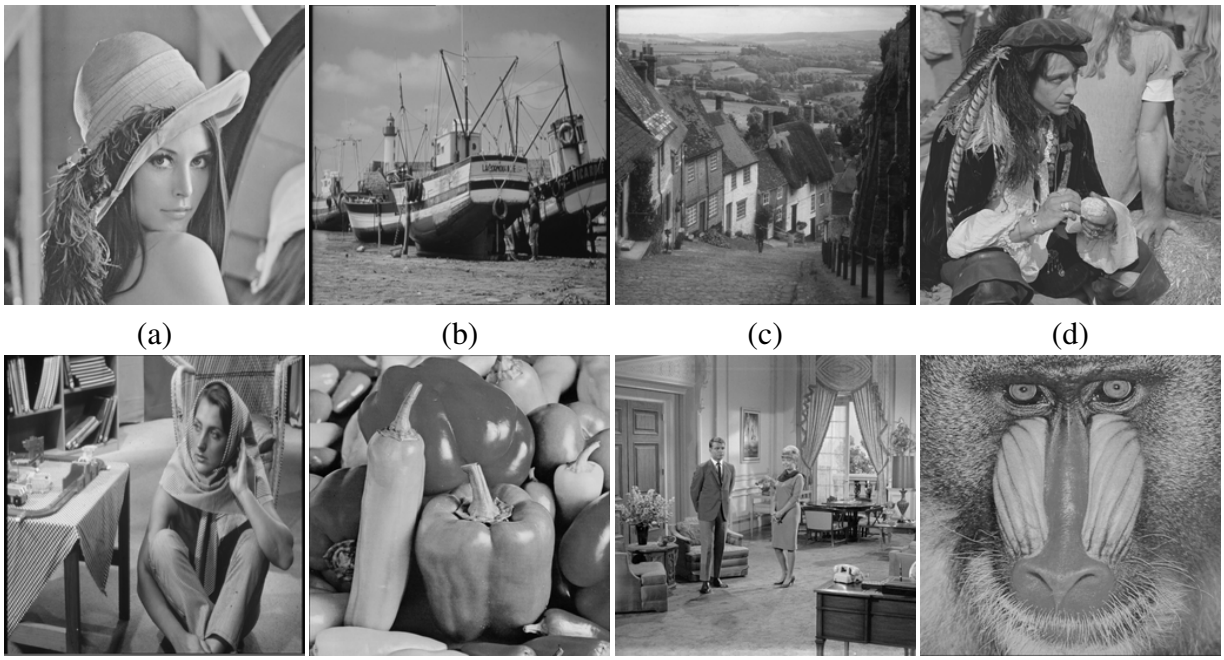
In this chapter, we will discuss some of the experiments and results of our suggested technique to improve the performance of BM3D. Initially, we will discuss the data set of images we have used for those experiments. Then we will discuss the performance metrics we have used to measure the performance of our suggested technique. Finally, we will discuss our conducted experiments in details and visually inspect the results produced from them.

4.1 Data Sets

For all the experiments performed in this chapter, we will use the eight standard images (Lena, Boats, Goldhill, Man, Barbara, Peppers, Couple, and Baboon) shown in Figure **??**. BM3D performs better when the input image is textured, e.g, Baboon and Barbara because the algorithm can find lots of similar patches which facilitates collaborative filtering [22]. On the other hand, if the image does not contain enough similar patterns, the performance of BM3D gets degraded. So, we have chosen these set of images which contain both textured and non-textured images.

4.2 Performance Measurement Metric

To measure the performance of our experimented method in comparison with the performance of BM3D, we have considered the Peak Signal to Noise Ratio (PSNR) measure as our performance metric. Besides, we will subjectively evaluate the results of our proposed scheme and compare them with the results of the BM3D scheme.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 4.1: Test Images: (a) Lena (b) Boats (c) Goldhill (d) Man (e) Barbara (f) Peppers (g) Couple (h) Baboon

4.2.1 Peak Signal to Noise Ratio (PSNR)

The PSNR can be represented by the following equation,

$$PSNR = 10 \times log_{10}(\frac{MAX_I^2}{MSE}) \tag{4.1}$$

Here, MAX_I is the maximum intensity of the image and MSE is the mean square error. We can represent MSE by the following equation,

$$MSE = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} (y(i, j) - \hat{y}(i, j))^2 \tag{4.2}$$

Here, $y(i, j)$ is the true pixel value at (i, j) coordinate, $\hat{y}(i, j)$ is the estimated pixel value, M is the number of rows and N is the number of columns in the image. A higher PSNR value represents a better-denoised image.

4.2.2 Subjective Fidelity Criteria

We will show the denoised images produced by both BM3D and our experimented method so that we can visually compare the performance of both of them.

4.3 Programming Language and Hardware details

We have used C++ as our programming language in all of our experiments. We have used HP pavilion g6 as our hardware for all of our experiments. We have used Intel(R) Core(TM) i5-2450M CPU @ 2.50GHz as the processor and the machine had 4GB memory.

4.4 Experimented Results

We have performed both BM3D and our experimented method on all the images in Figure **??** for noise levels 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100. Table 4.1 and Table 4.2 show the individual PSNR of all denoised images for all noise levels produced by both BM3D and our experimented method in two different cases namely, Case 1 and Case 2.

- Case 1:
  In this case, when we will get the number of similar patches less than or equal to two for a current processing patch we will perform a 2D transform on this patch instead on 3D processing. Otherwise, we will perform 3D processing on the patch.

- Case 2:
  In this case, when we will get the number of similar patches less than or equal to four for a current processing patch we will perform 2D transform on this patch instead on 3D processing. Otherwise, we will perform 3D processing on the patch.

We have also shown the average PSNR of all the denoised images produced by both BM3D and our experimented method in each of the noise levels in Table 4.1 and Table 4.2.

If we observe the results in Table 4.1 and Table 4.2 for all sigma level, we can notice that we achieve better PSNR improvement for all noise levels. BM3D has a sharp drop in denoising performance [22] for noise level equal and greater than 40. When the noise level is high, BM3D struggles to find similar patches [22]. When BM3D struggles to find similar patches, it can not exploit the intra-patch correlation [7]. To denoise these patches, we do not rely on similar patches as they might not be similar enough. So, we are denoising based on only inter-patch correlation which is giving us better denoising performance than the original BM3D. Although we can observe significant improvement for all noise levels, we can observe slightly better denoising performance for high noise levels.

Table 4.3 shows the average PSNR comparison between BM3D, experimented method case 1 and experimented method case 2. From Table 4.3, we can observe that both experimented cases produce better denoising performance than BM3D for all images and all noise levels.

Table 4.1: Comparison of PSNR of BM3D and Experimented Method Case 1: Applying 2D processing when having number of similar patches less than or equal to two. Exp: The PSNR of the denoised image produced by our experimented method, Imp: PSNR difference between the denoised images produced by our experimented method and the BM3D method in dB

σ		Lena	Boats	Goldhill	Man	Barbara	Peppers	Couple	Baboon	Average
10	BM3D	35.07	33.23	32.62	32.20	33.25	34.71	32.23	30.26	32.95
	Exp	35.37	33.91	33.13	32.70	33.79	35.09	33.00	30.95	33.49
	Imp	**0.30**	**0.68**	**0.51**	**0.50**	**0.54**	**0.38**	**0.77**	**0.69**	**0.54**
20	BM3D	31.08	29.52	29.35	28.72	29.35	30.75	28.43	26.51	29.21
	Exp	31.57	30.04	29.70	29.17	29.90	31.32	29.13	27.12	29.74
	Imp	**0.49**	**0.52**	**0.35**	**0.45**	**0.55**	**0.57**	**0.70**	**0.61**	**0.53**
30	BM3D	28.95	27.34	27.52	26.80	26.94	28.59	26.32	24.93	27.17
	Exp	29.61	28.02	27.87	27.29	27.88	29.07	27.02	25.23	27.75
	Imp	**0.66**	**0.68**	**0.35**	**0.49**	**0.94**	**0.48**	**0.70**	**0.30**	**0.58**
40	BM3D	27.45	25.84	26.32	25.55	25.37	26.83	24.97	23.92	25.78
	Exp	27.96	26.48	26.68	25.97	26.26	27.48	25.57	24.27	26.33
	Imp	**0.51**	**0.64**	**0.36**	**0.42**	**0.89**	**0.65**	**0.60**	**0.35**	**0.55**
50	BM3D	26.28	24.71	25.30	24.52	23.98	25.60	24.00	23.34	24.72
	Exp	26.84	25.46	25.83	24.91	25.00	26.04	24.49	23.59	25.27
	Imp	**0.56**	**0.75**	**0.53**	**0.39**	**1.02**	**0.44**	**0.49**	**0.25**	**0.55**
60	BM3D	25.09	23.81	24.53	23.70	23.00	24.50	23.27	22.87	23.85
	Exp	25.91	24.54	25.07	24.29	24.10	25.24	23.84	23.03	24.50
	Imp	**0.82**	**0.73**	**0.54**	**0.59**	**1.1**	**0.74**	**0.57**	**0.16**	**0.65**
70	BM3D	24.47	23.26	23.88	23.14	22.15	23.57	22.56	22.47	23.19
	Exp	25.08	23.83	24.47	23.82	23.18	24.50	23.13	22.65	23.83
	Imp	**0.61**	**0.57**	**0.59**	**0.68**	**1.03**	**0.93**	**0.57**	**0.18**	**0.64**
80	BM3D	24.01	22.62	23.42	22.54	21.65	23.11	22.02	22.06	22.68
	Exp	24.64	23.34	23.97	23.22	22.36	23.82	22.74	22.38	23.30
	Imp	**0.63**	**0.72**	**0.55**	**0.68**	**0.71**	**0.71**	**0.72**	**0.32**	**0.62**
90	BM3D	23.11	22.30	22.74	22.10	21.05	22.73	21.72	21.73	22.19
	Exp	23.93	22.92	23.42	22.51	21.88	23.42	22.31	22.11	22.81
	Imp	**0.82**	**0.62**	**0.68**	**0.41**	**0.83**	**0.69**	**0.59**	**0.38**	**0.62**
100	BM3D	22.60	21.60	22.20	21.52	20.60	21.75	21.21	21.46	21.62
	Exp	23.50	22.45	23.07	22.38	21.29	22.61	21.91	21.90	22.39
	Imp	**0.90**	**0.85**	**0.87**	**0.86**	**0.69**	**0.86**	**0.70**	**0.44**	**0.77**

Table 4.2: Comparison of PSNR of BM3D and Experimented Method Case 2 : Applying 2D processing when having number of similar patches less than or equal to four. Exp: The PSNR of the denoised image produced by our experimented method, Imp: The PSNR difference between the denoised images produced by our experimented method and the BM3D method in dB

σ		Lena	Boats	Goldhill	Man	Barbara	Peppers	Couple	Baboon	Average
10	BM3D	35.07	33.23	32.62	32.20	33.25	34.71	32.23	30.26	32.95
	Exp	35.38	33.94	33.20	32.72	33.75	35.00	33.03	30.94	33.50
	Imp	**0.31**	**0.71**	**0.58**	**0.52**	**0.50**	**0.29**	**0.80**	**0.68**	**0.55**
20	BM3D	31.08	29.52	29.35	28.72	29.35	30.75	28.43	26.51	29.21
	Exp	31.69	30.03	29.80	29.18	29.98	31.39	29.12	26.98	29.77
	Imp	**0.61**	**0.51**	**0.45**	**0.46**	**0.63**	**0.64**	**0.69**	**0.47**	**0.56**
30	BM3D	28.95	27.34	27.52	26.80	26.94	28.59	26.32	24.93	27.17
	Exp	29.44	27.95	27.91	27.35	27.96	29.12	27.12	25.24	27.76
	Imp	**0.49**	**0.61**	**0.39**	**0.55**	**1.02**	**0.53**	**0.80**	**0.31**	**0.59**
40	BM3D	27.45	25.84	26.32	25.55	25.37	26.83	24.97	23.92	25.78
	Exp	27.90	26.61	26.76	25.93	26.34	27.42	25.59	24.27	26.35
	Imp	**0.45**	**0.77**	**0.44**	**0.38**	**0.97**	**0.59**	**0.62**	**0.35**	**0.57**
50	BM3D	26.28	24.71	25.30	24.52	23.98	25.60	24.00	23.34	24.72
	Exp	26.94	25.58	25.75	25.12	24.84	26.12	24.50	23.50	25.29
	Imp	**0.66**	**0.87**	**0.45**	**0.60**	**0.86**	**0.52**	**0.50**	**0.16**	**0.58**
60	BM3D	25.09	23.81	24.53	23.70	23.00	24.50	23.27	22.87	23.85
	Exp	25.86	24.48	25.19	24.31	24.00	25.29	23.80	23.08	24.50
	Imp	**0.77**	**0.73**	**0.66**	**0.61**	**1.0**	**0.79**	**0.53**	**0.21**	**0.66**
70	BM3D	24.47	23.26	23.88	23.14	22.15	23.57	22.56	22.47	23.19
	Exp	25.17	23.82	24.39	23.70	23.16	24.44	23.19	22.70	23.82
	Imp	**0.70**	**0.56**	**0.51**	**0.56**	**1.01**	**0.87**	**0.57**	**0.23**	**0.63**
80	BM3D	24.01	22.62	23.42	22.54	21.65	23.11	22.02	22.06	22.68
	Exp	24.46	23.22	23.93	23.14	22.52	23.78	22.77	22.38	23.28
	Imp	**0.45**	**0.60**	**0.51**	**0.60**	**0.87**	**0.67**	**0.75**	**0.32**	**0.60**
90	BM3D	23.11	22.30	22.74	22.10	21.05	22.73	21.72	21.73	22.19
	Exp	24.09	22.90	23.49	22.77	21.82	23.15	22.28	22.04	22.82
	Imp	**0.98**	**0.60**	**0.75**	**0.67**	**0.77**	**0.42**	**0.56**	**0.31**	**0.63**
100	BM3D	22.60	21.60	22.20	21.52	20.60	21.75	21.21	21.46	21.62
	Exp	23.45	22.44	23.10	22.11	21.39	22.49	21.88	21.83	22.34
	Imp	**0.85**	**0.84**	**0.90**	**0.59**	**0.79**	**0.74**	**0.67**	**0.37**	**0.72**

Table 4.3: Comparison of average PSNR(dB) of BM3D, Experimented Method case 1: (Applying 2D processing when having number of similar patches less than or equal to two), Experimented Method Case 2: (Applying 2D processing when having number of similar patches less than or equal to four)

σ	BM3D Average PSNR	Experimented Average PSNR(Case : 1)	Improvement in Case: 1	Experimented Average PSNR(Case: 2)	Improvement in Case: 2
10	32.95	33.49	0.54	33.50	**0.55**
20	29.21	29.74	0.53	29.77	**0.56**
30	27.17	27.75	0.58	27.76	**0.59**
40	25.78	26.33	0.55	26.35	**0.57**
50	24.72	25.27	0.55	25.29	**0.57**
60	23.85	24.50	0.65	24.50	0.65
70	23.19	23.83	**0.64**	23.82	0.63
80	22.68	23.31	**0.63**	23.28	0.60
90	22.19	22.81	0.62	22.82	**0.63**
100	21.62	22.39	**0.77**	22.34	0.72

Although the amount is negligible, we can observe that our experimented method case 1 produces better denoising performance than experimented method case 2, especially at high noise levels. In our experimented method case 2, if we find any patch which has similar patches less than or equal to four, we perform 2D processing on those patches. This method should produce better denoising performance than experimented method case 1. But, when we get four similar patches, it might be better, in some cases, to process those patches with 3D processing rather than 2D processing, especially at low noise levels.

Table 4.4 and Table 4.5 shows the runtime difference between BM3D and our experimented method in seconds. Both of these tables show that our experimented methods produces less runtime than BM3D. On average, our experimented method case 1 produced 0.99% and case 2 produced 1.08% reduced runtime than original BM3D.

Table 4.6 shows the percentage of the number of patches having two similar patches and Table 4.7 shows the percentage of the number of patches having two or four similar patches. Both of these tables show that the percentage is more for higher noise levels. The Euclidean distance increases for higher noise levels which makes it difficult to find a similar patch.

4.5 Visual Comparison

Figure 4.10 shows a visual comparison between the denoising performance of our experimented method case 1 and the denoising performance of original BM3D. If we compare the

Table 4.4: Comparison of runtime (seconds) of BM3D and Experimented Method case 1: Applying 2D processing when having number of similar patches less than or equal to two, Exp: The runtime(seconds) of the denoised image produced by our experimented method case 1, Imp: runtime (seconds) difference between the denoised images produced by our experimented method and the BM3D method

σ		Lena	Boats	Goldhill	Man	Barbara	Peppers	Couple	Baboon	Average
10	BM3D	44.62	51.87	49.54	57.88	48.33	57.71	49.92	56.09	52.00
	Exp	44.32	51.07	49.15	57.25	47.89	57.11	49.29	55.50	51.45
	Imp	**0.30**	**0.80**	**0.39**	**0.63**	**0.44**	**0.60**	**0.63**	**0.59**	**0.55**
20	BM3D	44.55	59.32	52.21	46.92	48.11	56.24	47.78	51.54	50.83
	Exp	44.19	59.24	52.17	46.76	47.75	55.72	47.04	51.16	50.50
	Imp	**0.36**	**0.08**	**0.04**	**0.16**	**0.36**	**0.52**	**0.74**	**0.38**	**0.33**
30	BM3D	45.17	61.15	51.92	48.57	47.76	62.31	52.16	58.30	53.42
	Exp	44.52	60.77	51.73	48.10	46.60	62.45	51.40	58.12	52.96
	Imp	**0.65**	**0.38**	**0.19**	**0.47**	**1.16**	**0.14**	**0.76**	**0.18**	**0.46**
40	BM3D	56.26	61.18	56.86	55.48	62.19	61.92	61.11	58.79	59.22
	Exp	55.74	60.74	56.27	55.02	61.18	60.74	60.54	58.67	58.61
	Imp	**0.52**	**0.44**	**0.59**	**0.46**	**1.01**	**1.18**	**0.57**	**0.12**	**0.61**
50	BM3D	56.98	60.56	62.70	53.58	59.50	58.74	57.56	56.29	58.24
	Exp	57.75	59.46	62.26	53.49	58.31	58.05	57.33	56.07	57.84
	Imp	**0.77**	**1.1**	**0.44**	**0.09**	**1.19**	**0.69**	**0.23**	**0.22**	**0.40**
60	BM3D	53.70	56.86	64.03	55.73	54.51	52.13	59.65	58.50	56.89
	Exp	52.95	56.17	63.57	55.39	53.15	51.35	59.14	58.42	56.27
	Imp	**0.75**	**0.69**	**0.46**	**0.34**	**1.36**	**0.78**	**0.51**	**0.08**	**0.62**
70	BM3D	64.98	59.11	62.90	52.16	54.29	51.84	56.79	58.55	57.58
	Exp	64.08	58.66	62.19	51.69	53.25	50.78	56.32	58.56	56.94
	Imp	**0.9**	**0.45**	**0.71**	**0.47**	**1.04**	**1.06**	**0.47**	**0.01**	**0.64**
80	BM3D	65.11	56.79	54.96	50.72	57.16	53.29	59.89	54.32	56.53
	Exp	64.67	55.11	54.59	50.32	56.68	52.43	59.19	54.05	55.88
	Imp	**0.44**	**1.68**	**0.37**	**0.40**	**0.48**	**0.86**	**0.70**	**0.27**	**0.65**
90	BM3D	51.93	57.23	53.58	50.99	53.24	57.10	59.46	50.70	54.28
	Exp	51.08	56.78	52.76	50.47	52.56	56.49	58.99	50.42	53.69
	Imp	**0.85**	**0.45**	**0.82**	**0.52**	**0.68**	**0.61**	**0.47**	**0.28**	**0.59**
100	BM3D	50.17	57.00	64.25	48.72	62.42	51.37	58.22	58.18	56.29
	Exp	49.55	56.27	63.58	48.09	61.65	50.77	57.55	57.64	55.64
	Imp	**0.62**	**0.73**	**0.67**	**0.63**	**0.77**	**0.60**	**0.67**	**0.54**	**0.65**

Table 4.5: Comparison of runtime (seconds) of BM3D and Experimented Method case 2: Applying 2D processing when having number of similar patches less than or equal to four, Exp: The runtime(seconds) of the denoised image produced by our experimented method case 2, Imp: runtime (seconds) difference between the denoised images produced by our experimented method and the BM3D method

σ		Lena	Boats	Goldhill	Man	Barbara	Peppers	Couple	Baboon	Average
10	BM3D	44.62	51.87	49.54	57.88	48.33	57.71	49.92	56.09	52.00
	Exp	44.21	51.05	49.05	57.22	47.88	57.11	49.21	55.48	51.40
	Imp	**0.41**	**0.82**	**0.49**	**0.66**	**0.45**	**0.6**	**0.71**	**0.61**	**0.60**
20	BM3D	44.55	59.32	52.21	46.92	48.11	56.24	47.78	51.54	50.83
	Exp	44.10	59.19	52.14	46.70	47.75	55.68	46.93	51.00	50.44
	Imp	**0.45**	**0.13**	**0.7**	**0.22**	**0.36**	**0.56**	**0.85**	**0.54**	**0.39**
30	BM3D	45.17	61.15	51.92	48.57	47.76	62.31	52.16	58.30	53.42
	Exp	44.51	60.77	51.69	48.09	46.55	62.27	51.36	58.08	52.92
	Imp	**0.66**	**0.38**	**0.23**	**0.48**	**1.21**	**0.04**	**0.20**	**0.22**	**0.50**
40	BM3D	56.26	61.18	56.86	55.48	62.19	61.92	61.11	58.79	59.22
	Exp	55.55	60.74	56.25	55.01	61.17	60.69	60.44	58.57	58.55
	Imp	**0.71**	**0.44**	**0.61**	**0.47**	**1.02**	**1.23**	**0.67**	**0.22**	**0.67**
50	BM3D	56.98	60.56	62.70	53.58	59.50	58.74	57.56	56.29	58.24
	Exp	57.74	59.46	62.23	53.41	58.30	58.04	57.29	55.99	57.81
	Imp	**0.76**	**1.1**	**0.47**	**0.17**	**1.2**	**0.70**	**0.27**	**0.3**	**0.43**
60	BM3D	53.70	56.86	64.03	55.73	54.51	52.13	59.65	58.50	56.89
	Exp	52.91	56.14	63.47	55.29	53.10	51.34	59.14	58.43	56.23
	Imp	**0.79**	**0.72**	**0.56**	**0.44**	**1.41**	**0.79**	**0.51**	**0.07**	**0.66**
70	BM3D	64.98	59.11	62.90	52.16	54.29	51.84	56.79	58.55	57.58
	Exp	64.08	58.63	62.17	51.62	53.18	50.70	56.23	58.46	56.88
	Imp	**0.9**	**0.48**	**0.73**	**0.54**	**1.11**	**1.14**	**0.56**	**0.09**	**0.7**
80	BM3D	65.11	56.79	54.96	50.72	57.16	53.29	59.89	54.32	56.53
	Exp	64.64	55.04	54.57	50.22	56.66	52.42	59.14	53.98	55.83
	Imp	**0.47**	**1.75**	**0.39**	**0.50**	**0.50**	**0.87**	**0.75**	**0.34**	**0.70**
90	BM3D	51.93	57.23	53.58	50.99	53.24	57.10	59.46	50.70	54.28
	Exp	51.03	56.72	52.72	50.41	52.48	56.42	58.91	50.41	53.64
	Imp	**0.9**	**0.51**	**0.86**	**0.58**	**0.76**	**0.68**	**0.55**	**0.29**	**0.64**
100	BM3D	50.17	57.00	64.25	48.72	62.42	51.37	58.22	58.18	56.29
	Exp	49.50	56.25	63.57	48.08	61.59	50.71	57.51	57.60	55.60
	Imp	**0.67**	**0.75**	**0.68**	**0.64**	**0.83**	**0.66**	**0.71**	**0.58**	**0.69**

Table 4.6: Percentage of patches having two similar patches

σ	Lena	Boats	Goldhill	Man	Barbara	Peppers	Couple	Baboon	Average
10	19.95	31.16	26.24	37.33	36.95	25.12	37.00	55.74	33.69
20	17.41	26.17	19.13	31.17	36.30	23.93	30.87	41.06	28.26
30	17.72	25.83	17.31	28.91	37.31	25.68	30.92	28.70	26.55
40	16.09	26.52	12.21	22.53	33.85	23.99	24.89	15.11	21.90
50	19.30	27.70	18.18	25.47	44.07	29.96	29.43	19.05	26.65
60	23.67	34.78	24.58	36.24	53.25	37.94	37.63	29.44	34.69
70	35.84	45.05	41.86	47.28	63.47	45.96	45.54	47.34	46.54
80	52.11	54.42	59.68	63.17	73.49	56.07	67.11	60.87	60.87
90	66.75	73.48	64.52	63.85	78.04	73.17	69.01	68.92	69.72
100	74.97	79.62	70.58	79.26	87.34	76.95	81.05	78.16	78.49

Table 4.7: Percentage of patches having two or four similar patches

σ	Lena	Boats	Goldhill	Man	Barbara	Peppers	Couple	Baboon	Average
10	20.65	31.36	26.46	37.84	37.16	25.24	37.12	55.75	33.95
20	17.55	26.39	19.43	31.22	36.66	24.12	31.34	41.41	28.52
30	18.18	26.25	17.36	28.95	37.33	25.88	31.11	28.91	26.75
40	16.16	26.78	12.38	22.80	34.16	24.23	25.08	15.15	22.09
50	19.79	28.18	18.58	26.17	44.62	30.41	29.84	19.42	27.13
60	24.17	35.35	24.88	36.83	53.65	38.34	37.90	29.63	35.09
70	36.06	45.33	42.13	47.68	63.83	46.32	45.99	47.70	46.88
80	52.39	54.87	59.88	63.42	73.65	56.39	67.51	61.29	61.18
90	66.92	73.62	64.78	64.07	78.39	73.60	69.44	67.15	69.75
100	75.31	79.97	70.87	79.74	87.76	77.31	81.22	78.26	78.81

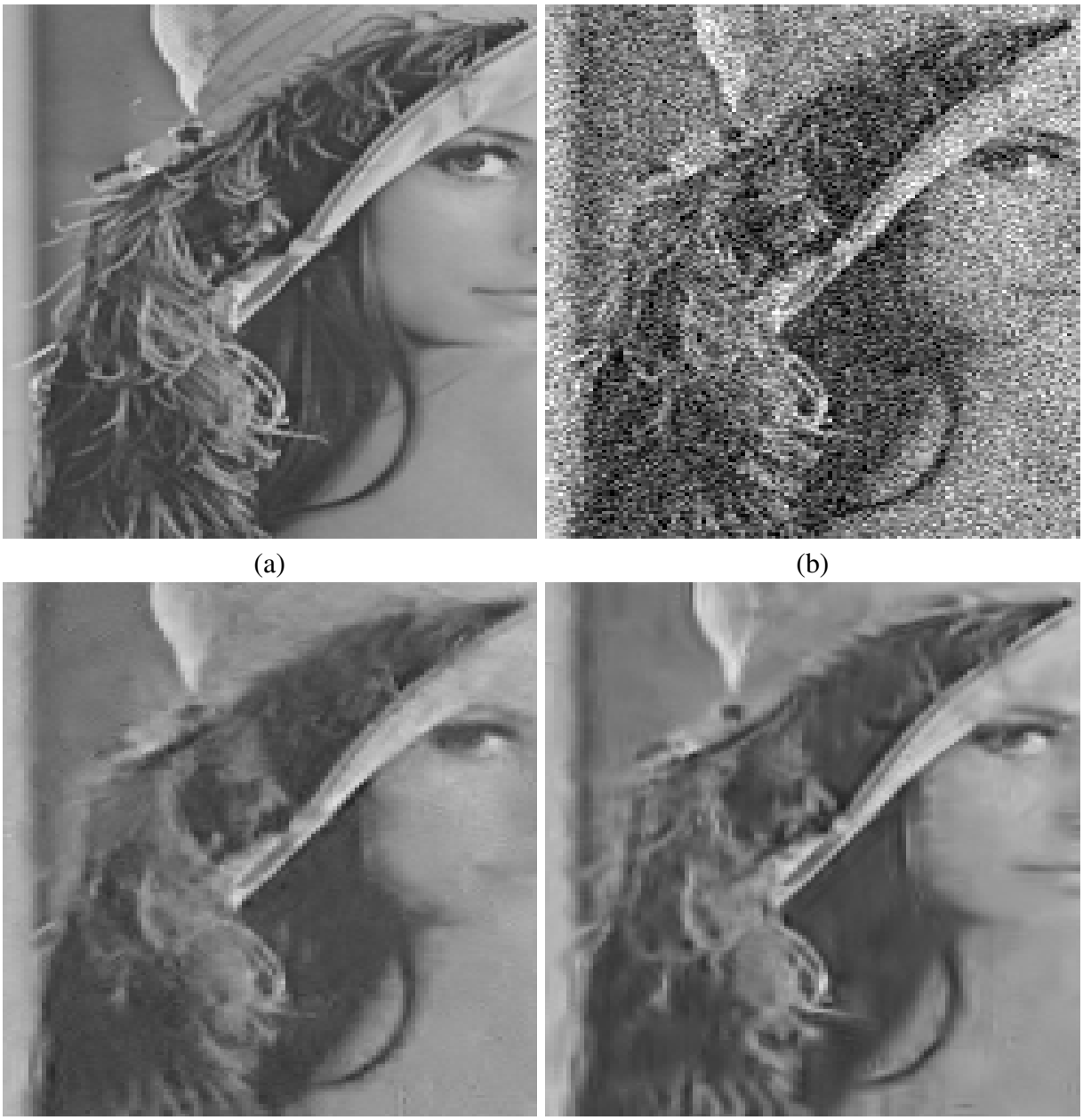
denoised image produced by BM3D and the denoised image produced by our experimented method case 1, we will find that our experimented method achieved visually better performance than the original BM3D.

If we compare both denoised images, we can find several regions where our experimented method performs visually better than the original BM3D scheme. Figure 4.11 shows a zoom-in for a part of the images in Figure 4.10, where the superiority of our method is signified.

The same observations can be applied to Figure 4.16 and Figure 4.17 when comparing the images produced by the original BM3D scheme and our proposed method case 2.

Figure 4.2: Visual result ((a) the original image, (b) the noisy image corrupted by gaussian noise of noise level 30, (c) the denoised image produced by BM3D and (d) the denoised image produced by our experimented method case 2: Applying 2D processing having the number of similar patches less than or equal to four)

.



(a) (b)

(c) (d)

Figure 4.3: Visual result with zoom-in ((a) the original image, (b) the noisy image corrupted by gaussian noise of noise level 30, (c) the denoised image produced by BM3D and (d) the denoised image produced by our experimented method case 2: Applying 2D processing having the number of similar patches less than or equal to four)

(a) (b)

(c) (d)

Figure 4.4: Visual result ((a) the original image, (b) the noisy image corrupted by gaussian noise of noise level 20, (c) the denoised image produced by BM3D and (d) the denoised image produced by our experimented method case 1: Applying 2D processing having the number of similar patches less than or equal to two)

.

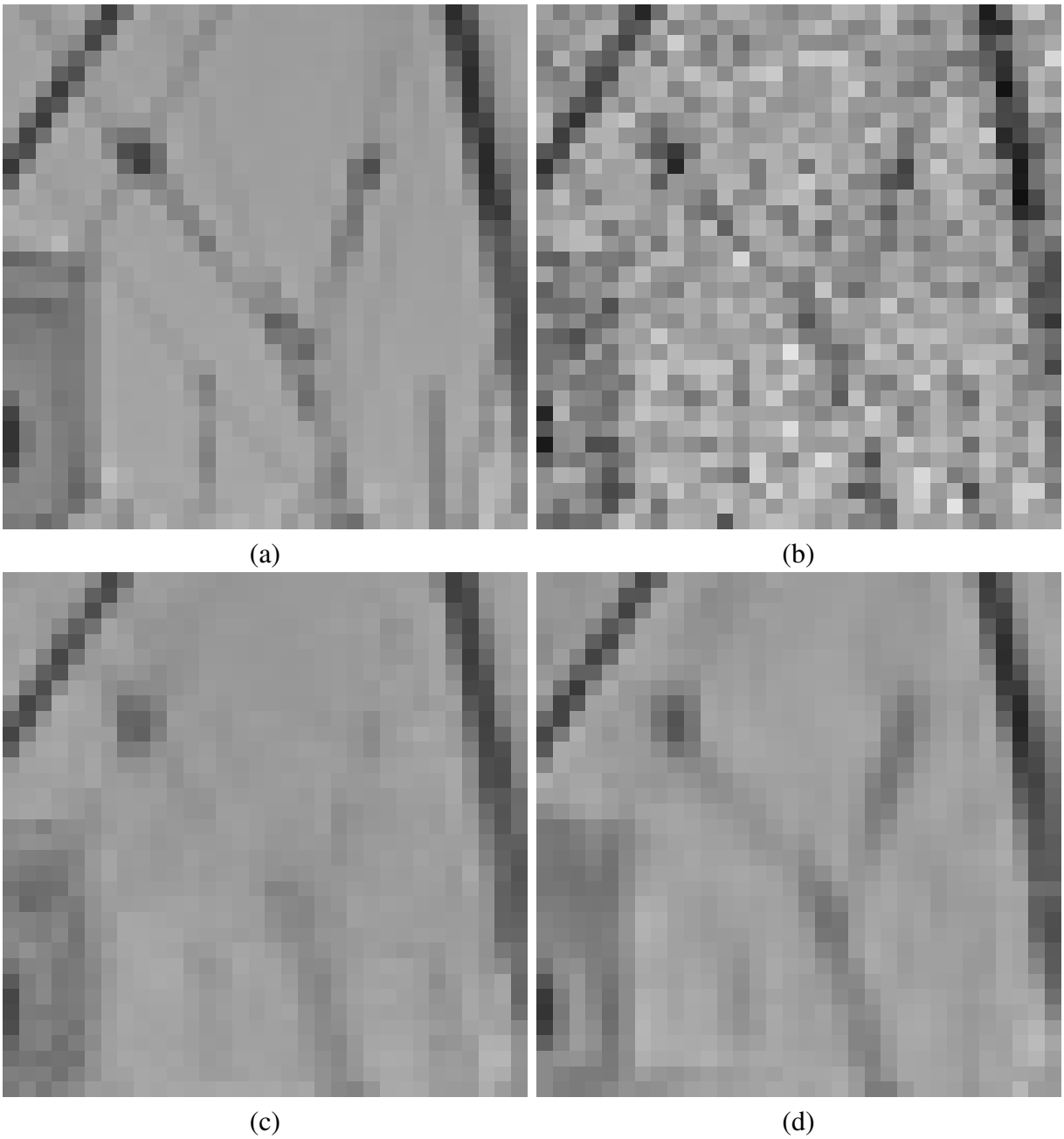


Figure 4.5: Visual result with zoom-in ((a) the original image, (b) the noisy image corrupted by gaussian noise of noise level 20, (c) the denoised image produced by BM3D and (d) the denoised image produced by our experimented method case 1: Applying 2D processing having the number of similar patches less than or equal to two)

(a) (b)

(c) (d)

Figure 4.6: Visual result ((a) the original image, (b) the noisy image corrupted by gaussian noise of noise level 10, (c) the denoised image produced by BM3D and (d) the denoised image produced by our experimented method case 2: Applying 2D processing having the number of similar patches less than or equal to four)

.

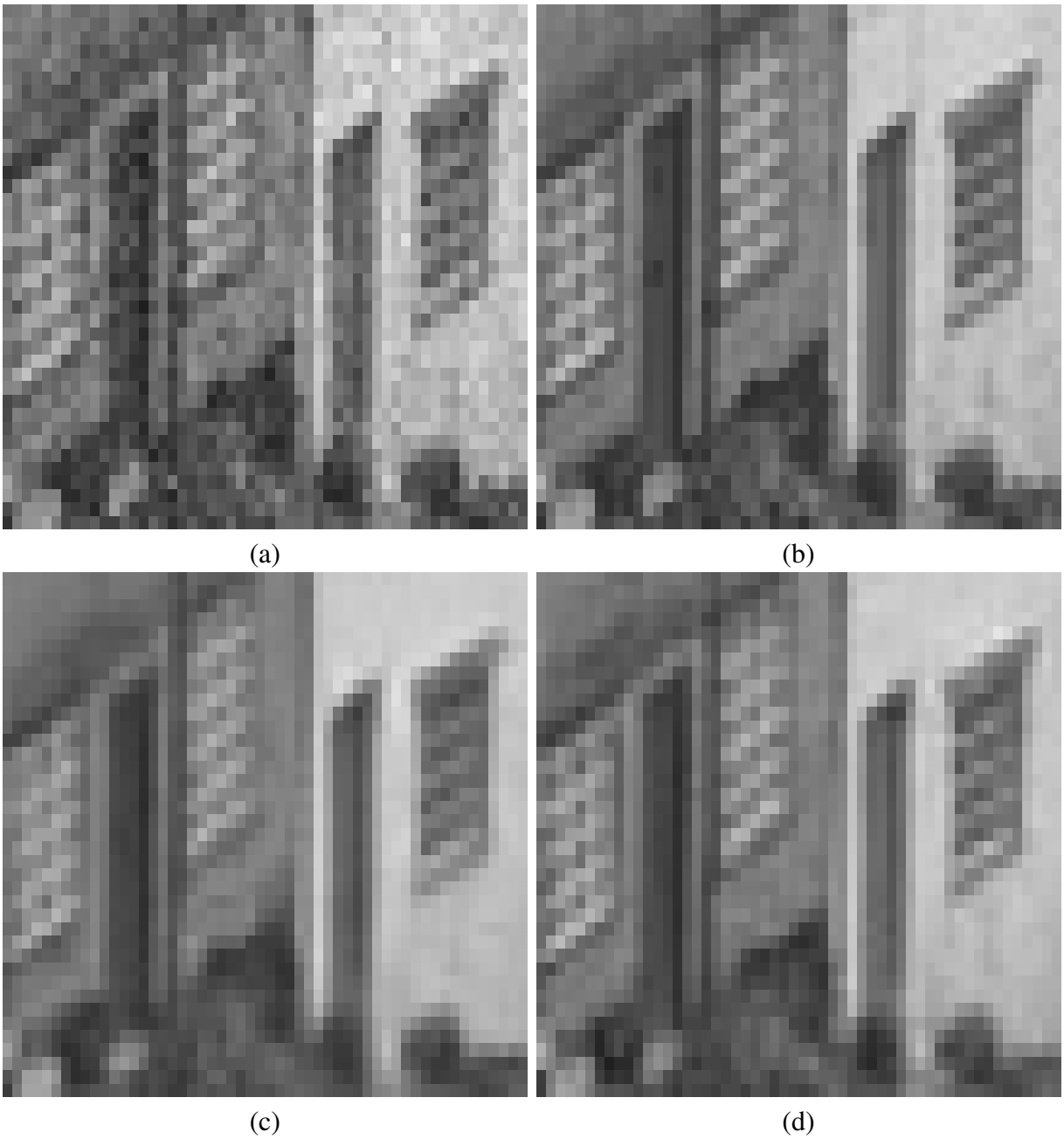


Figure 4.7: Visual result with zoom-in ((a) the original image, (b) the noisy image corrupted by gaussian noise of noise level 20, (c) the denoised image produced by BM3D and (d) the denoised image produced by our experimented method case 2: Applying 2D processing having the number of similar patches less than or equal to four)

Figure 4.8: Visual result ((a) the original image, (b) the noisy image corrupted by gaussian noise of noise level 10, (c) the denoised image produced by BM3D and (d) the denoised image produced by our experimented method case 1: Applying 2D processing having the number of similar patches less than or equal to two)

.

Figure 4.9: Visual result with zoom-in ((a) the original image, (b) the noisy image corrupted by gaussian noise of noise level 10, (c) the denoised image produced by BM3D and (d) the denoised image produced by our experimented method case 1: Applying 2D processing having the number of similar patches less than or equal to two)

(a) (b)

(c) (d)

Figure 4.10: Visual result ((a) the original image, (b) the noisy image corrupted by gaussian noise of noise level 60, (c) the denoised image produced by BM3D and (d) the denoised image produced by our experimented method case 1: Applying 2D processing having the number of similar patches less than or equal to two)

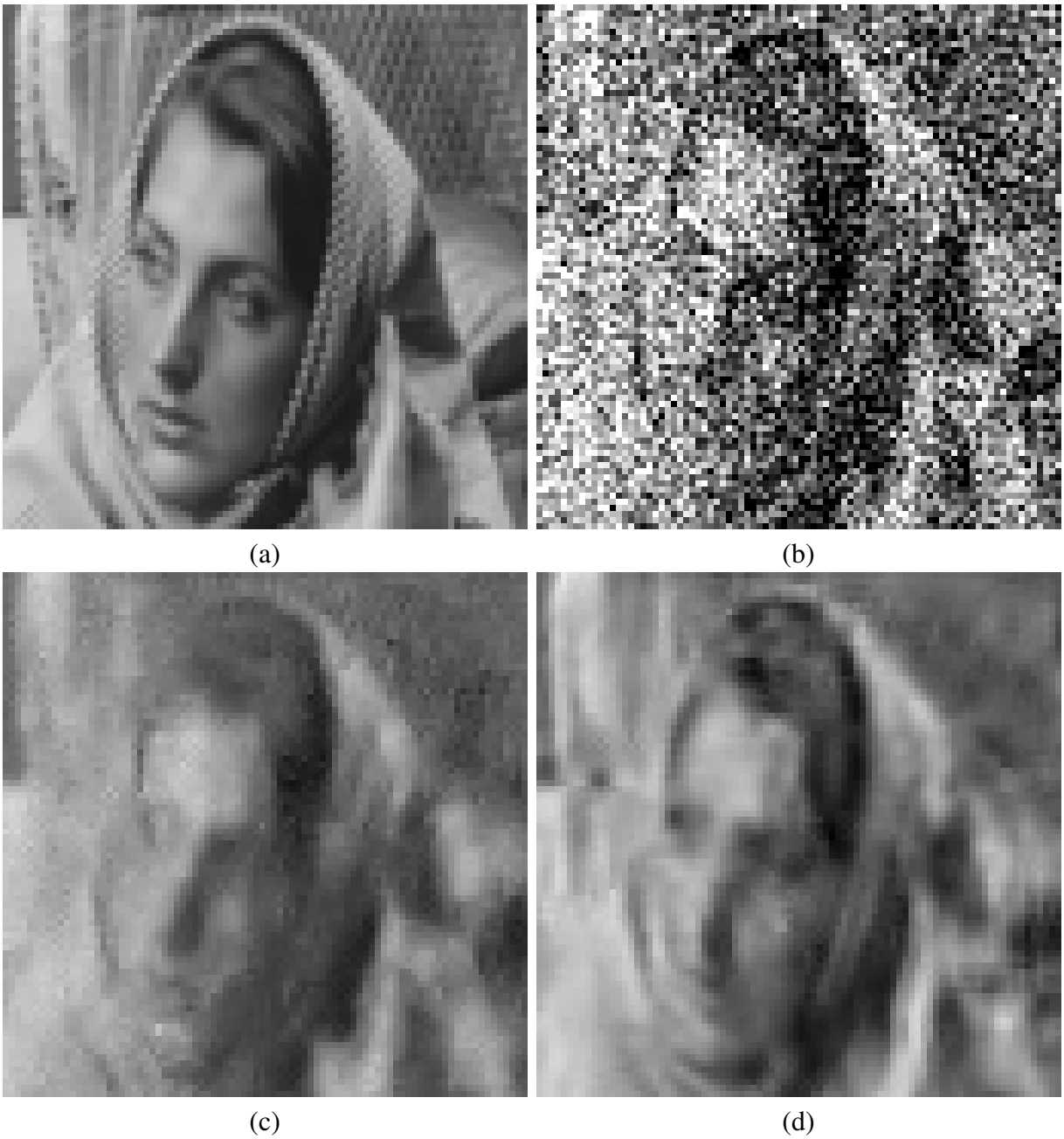


Figure 4.11: Visual result with zoom-in ((a) the original image, (b) the noisy image corrupted by gaussian noise of noise level 60, (c) the denoised image produced by BM3D and (d) the denoised image produced by our experimented method case 1: Applying 2D processing having the number of similar patches less than or equal to two)

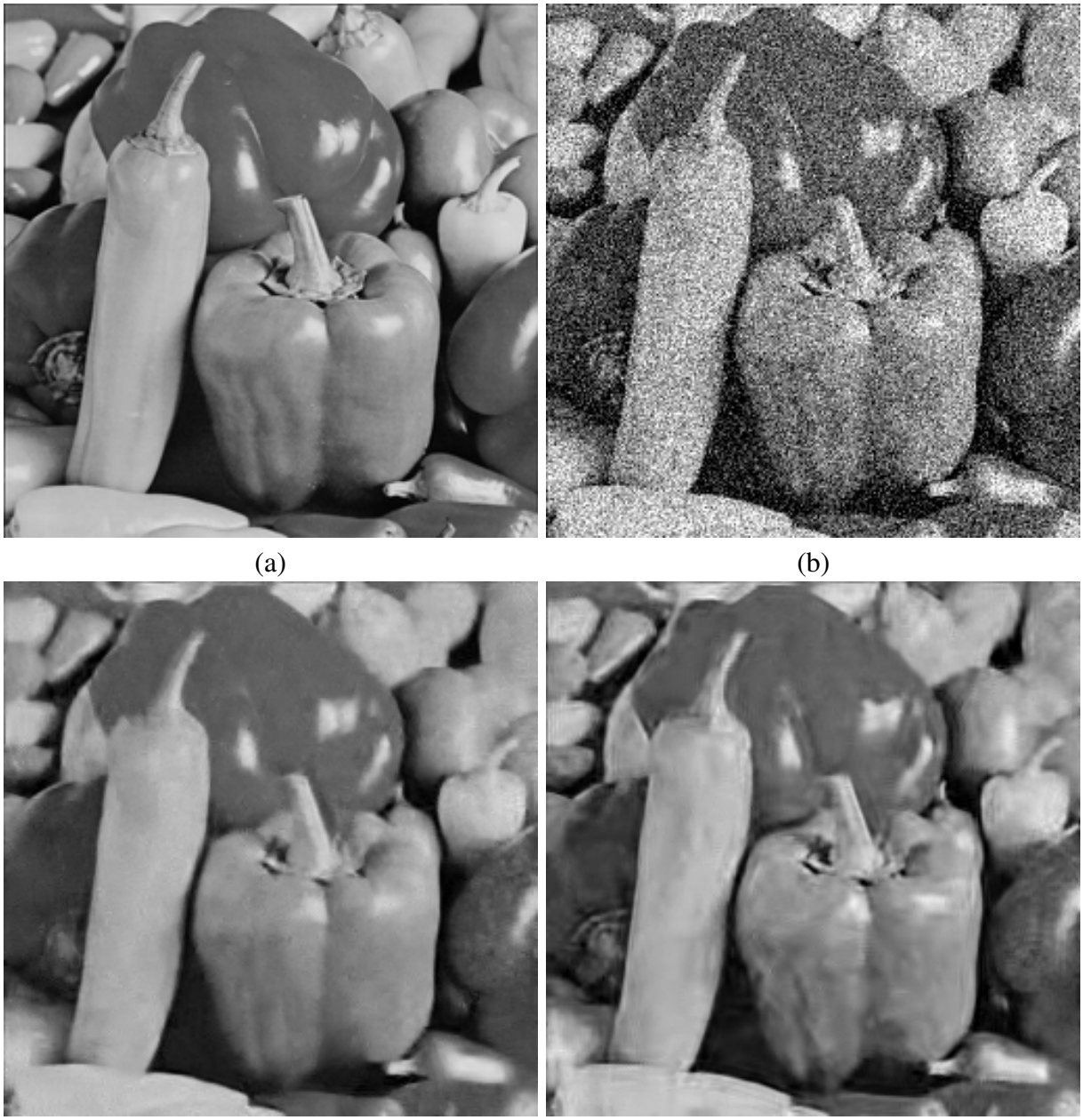


Figure 4.12: Visual result ((a) the original image, (b) the noisy image corrupted by gaussian noise of noise level 10, (c) the denoised image produced by BM3D and (d) the denoised image produced by our experimented method case 2: Applying 2D processing having the number of similar patches less than or equal to four)

.

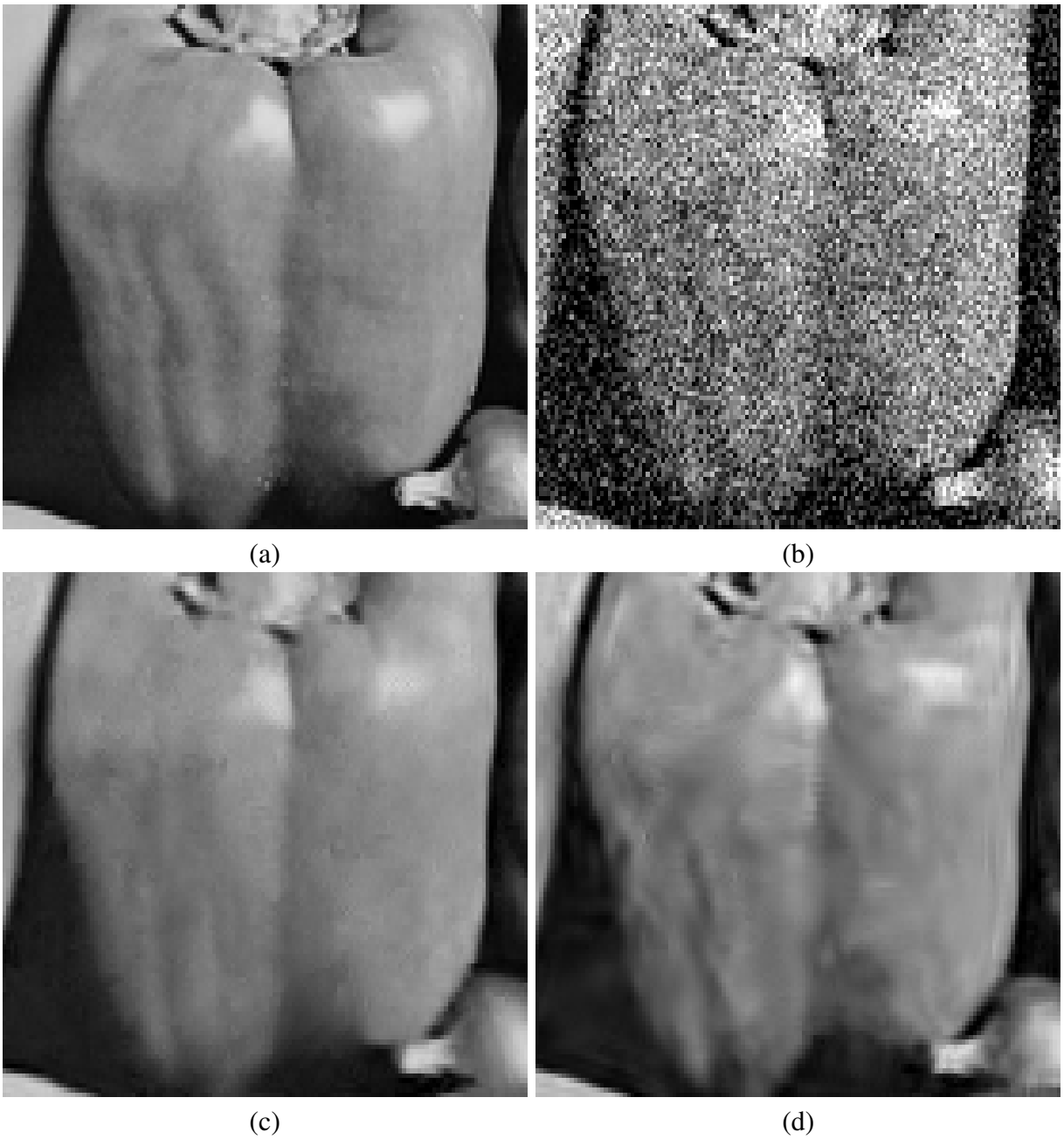


Figure 4.13: Visual result with zoom-in ((a) the original image, (b) the noisy image corrupted by gaussian noise of noise level 10, (c) the denoised image produced by BM3D and (d) the denoised image produced by our experimented method case 2: Applying 2D processing having the number of similar patches less than or equal to four)

Figure 4.14: Visual result ((a) the original image, (b) the noisy image corrupted by gaussian noise of noise level 20, (c) the denoised image produced by BM3D and (d) the denoised image produced by our experimented method case 1: Applying 2D processing having the number of similar patches less than or equal to two)

.

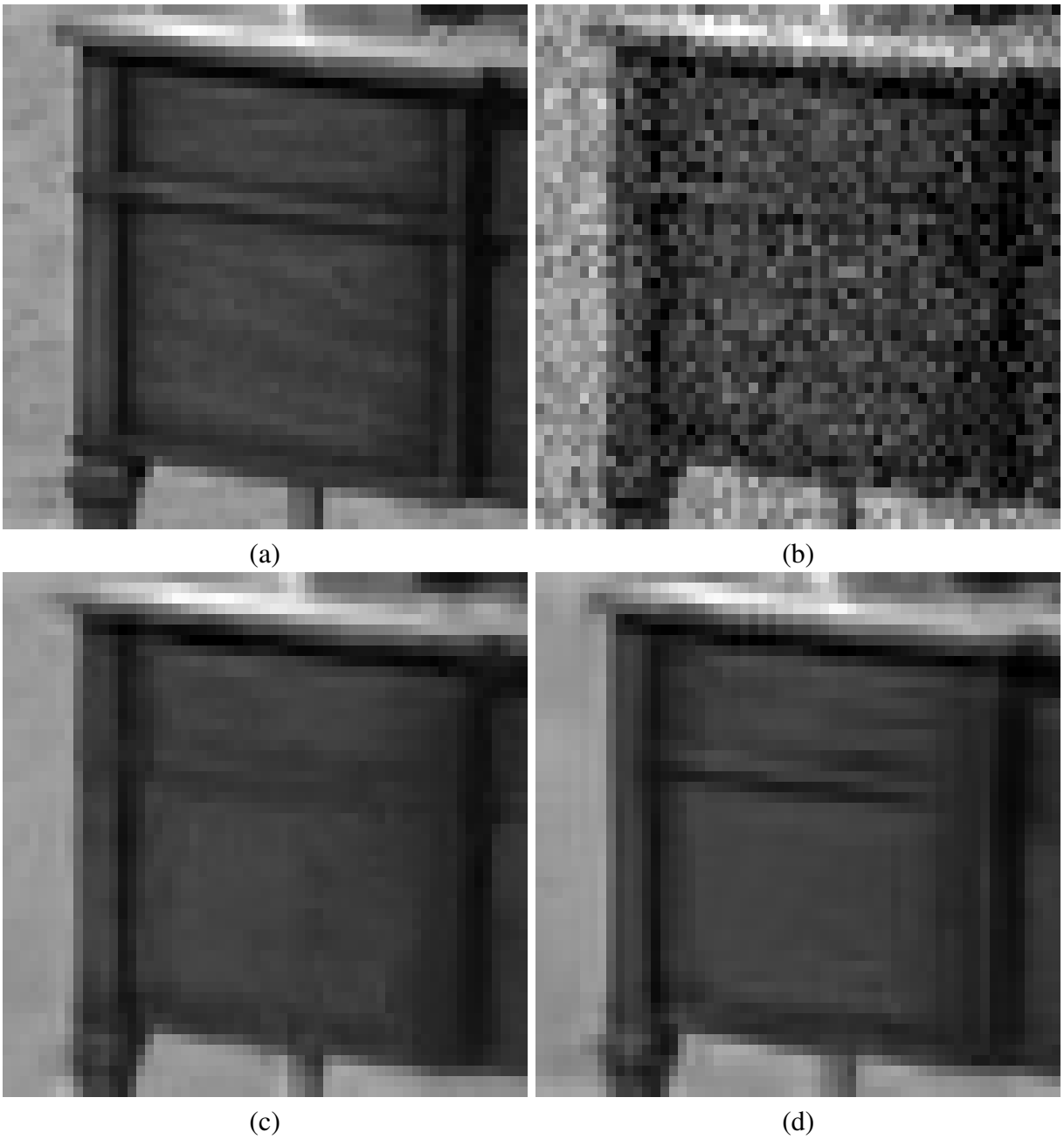


Figure 4.15: Visual result with zoom-in ((a) the original image, (b) the noisy image corrupted by gaussian noise of noise level 10, (c) the denoised image produced by BM3D and (d) the denoised image produced by our experimented method case 1: Applying 2D processing having the number of similar patches less than or equal to two)

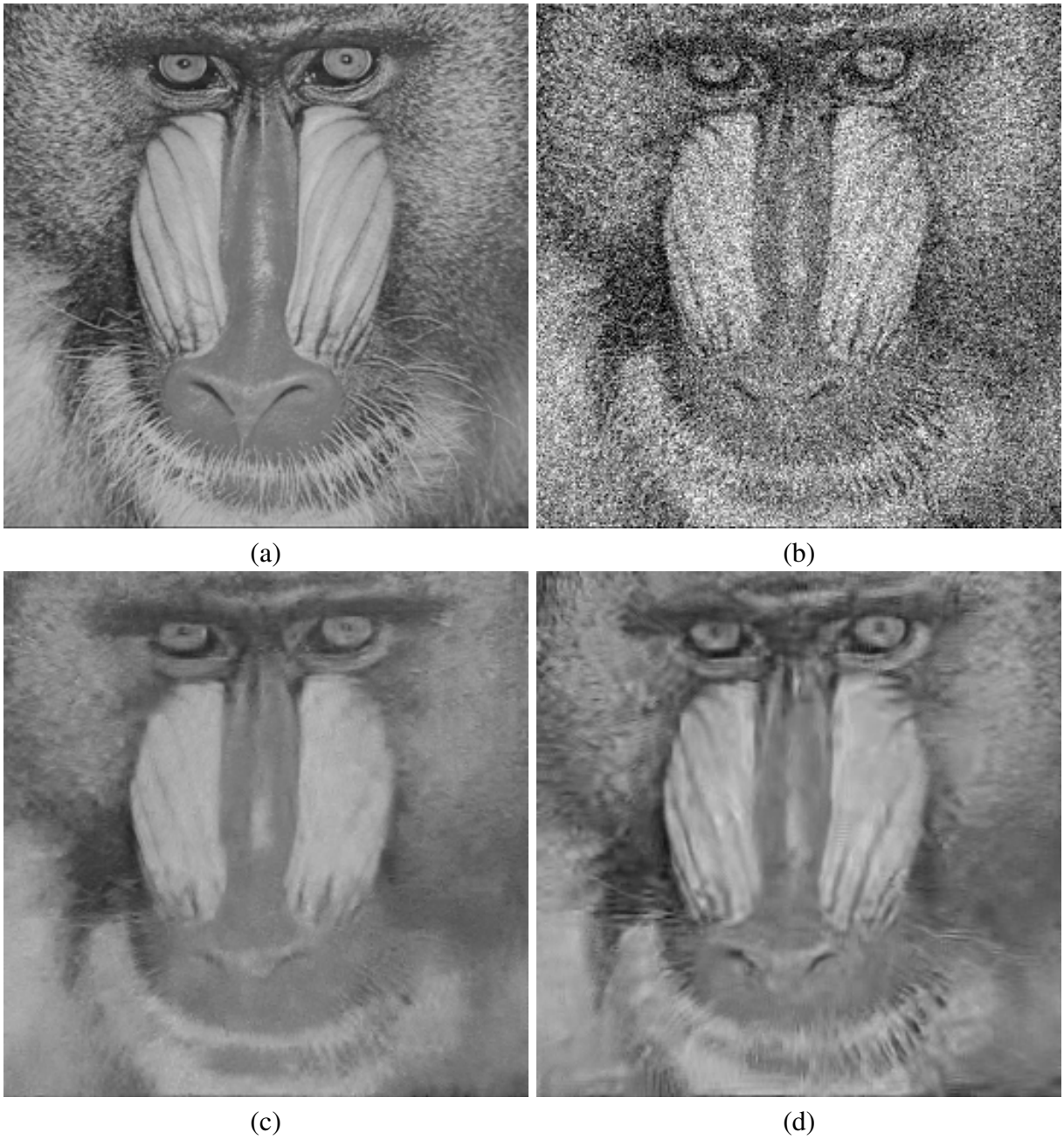


Figure 4.16: Visual result ((a) the original image, (b) the noisy image corrupted by gaussian noise of noise level 40, (c) the denoised image produced by BM3D and (d) the denoised image produced by our experimented method case 2: Applying 2D processing having the number of similar patches less than or equal to four)

.

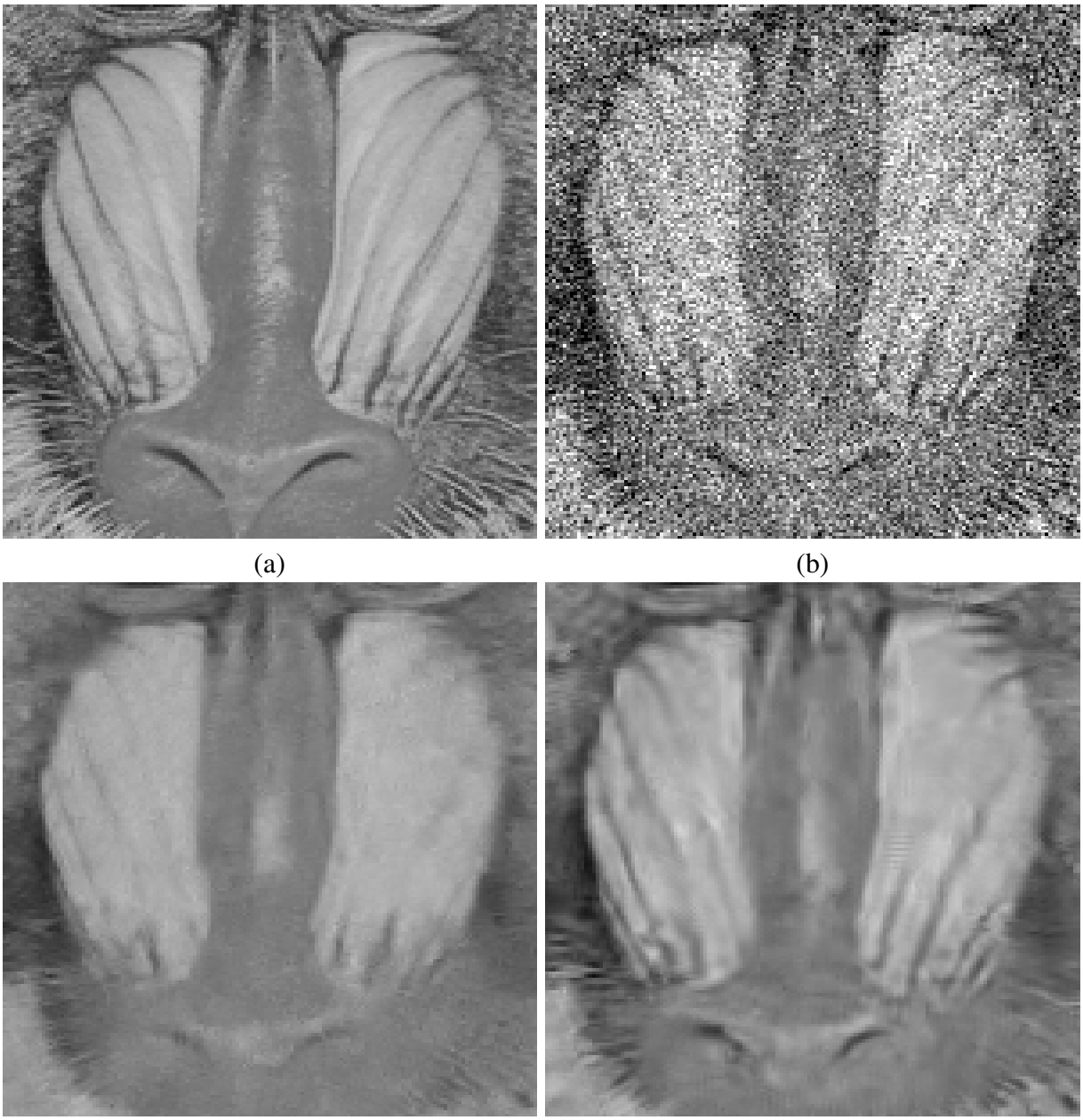


Figure 4.17: Visual result with zoom-in ((a) the original image, (b) the noisy image corrupted by gaussian noise of noise level 40, (c) the denoised image produced by BM3D and (d) the denoised image produced by our experimented method case 2: Applying 2D processing having the number of similar patches less than or equal to four)

Chapter 5

Conclusion and Future Work

In this chapter, we will discuss the summary of our work and possible future works to achieve further better denoising.

5.1 Summary

In this thesis, we have discussed different types of noises and techniques to denoise additive Gaussian noise. We have also discussed the current state-of-the-art denoising scheme, Block Matching and 3D (BM3D) filtering algorithm in details. We have discussed the prominent lackings of this algorithm and ways to improve it. We have suggested a technique to improve the performance of BM3D.

5.2 Conclusion

The original BM3D algorithm performs 3D transform for all the patches regardless of how many similar patches the algorithm could find. We have shown that if it is not possible to find enough similar patches, performing a 2D transform on those patches, instead of 3D transform, can produce better denoising performance.

Hence, we proposed that if enough number of similar patches can not be found in the image, we should perform the 2D transform, instead of 3D transform, on those patches to achieve better denoising performance as a whole. We have also shown that our experimented method produces reduced runtime than original BM3D.

5.3 Future Work

We would like to search for a better transformation for those patches for whom it is not possible to find similar patches in the image. We would also like to extend the algorithm to color images, as well as moving sequences.

Bibliography

[1] Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

[2] Monagi H Alkinani and Mahmoud R El-Sakka. Patch-based models and algorithms for image denoising: a comparative review between patch-based images denoising methods for additive noise reduction. *EURASIP Journal on Image and Video Processing*, 2017(1):58, 2017.

[3] Abdourrahmane M Atto, Dominique Pastor, and Gregoire Mercier. Smooth sigmoid wavelet shrinkage for non-parametric estimation. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3265–3268. IEEE, 2008.

[4] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005.

[5] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling & Simulation*, 4(2):490–530, 2005.

[6] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. Non-local means denoising. *Image Processing On Line*, 1:208–212, 2011.

[7] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8):2080–2095, 2007.

[8] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. A nonlocal and shape-adaptive transform-domain collaborative filtering. In *Proc. Int. Workshop on Local and Non-Local Approx. in Image Process., LNLA*, 2008.

[9] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Bm3d image denoising with shape-adaptive principal component analysis. In *SPARS'09-Signal Processing with Adaptive Sparse Structured Representations*, 2009.

[10] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen O Egiazarian. Color image denoising via sparse 3d collaborative filtering with grouping constraint in luminance-chrominance space. In *ICIP (1)*, pages 313–316, 2007.

[11] Li Dai, Yousai Zhang, and Yuanjiang Li. Image denoising using bm3d combining tetrolet prefiltering. *Information Technology Journal*, 12(10):1995–2001, 2013.

[12] David L Donoho. De-noising by soft-thresholding. *IEEE transactions on information theory*, 41(3):613–627, 1995.

[13] David L Donoho and Iain M Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the american statistical association*, 90(432):1200–1224, 1995.

[14] David L Donoho and Jain M Johnstone. Ideal spatial adaptation by wavelet shrinkage. *biometrika*, 81(3):425–455, 1994.

[15] Shlomo Engelberg. The central limit theorem and low-pass filters. In *Proceedings of the 2004 11th IEEE International Conference on Electronics, Circuits and Systems, 2004. ICECS 2004.*, pages 65–68. IEEE, 2004.

[16] RL Fante. Central limit theorem: use with caution. *IEEE Transactions on Aerospace and Electronic Systems*, 37(2):739–740, 2001.

[17] Xiang-Chu Feng, Xiao-Hui Li, Wei-Wei Wang, and Xi-Xi Jia. Improvement of bm3d algorithm based on wavelet and directed diffusion. In *2017 International Conference on Machine Vision and Information Technology (CMVIT)*, pages 28–33. IEEE, 2017.

[18] Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Pointwise shape-adaptive dct for high-quality denoising and deblocking of grayscale and color images. *IEEE transactions on image processing*, 16(5):1395–1411, 2007.

[19] A Goldenshluger and A Nemirovski. On spatially adaptive estimation of nonparametric regression. *Mathematical methods of Statistics*, 6(2):135–170, 1997.

[20] Rafael C Gonzalez and Richard E Woods. Image processing. *Digital image processing*, 2:1, 2007.

[21] Mahmud Hasan and Mahmoud R El-Sakka. Improved bm3d image denoising using ssim-optimized wiener filter. *EURASIP Journal on Image and Video Processing*, 2018(1):25, 2018.

[22] Yingkun Hou, Chunxia Zhao, Deyun Yang, and Yong Cheng. Comments on" image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 20(1):268–270, 2011.

[23] Reinhard Illner and Jinghzi Tie. On directed diffusion with measurable background. *Mathematical methods in the applied sciences*, 16(10):681–690, 1993.

[24] Paras Jain and Vipin Tyagi. A survey of edge-preserving image denoising methods. *Information Systems Frontiers*, 18(1):159–170, 2016.

[25] Vladimir Katkovnik. A new method for varying adaptive bandwidth selection. *IEEE Transactions on signal processing*, 47(9):2567–2571, 1999.

[26] Vladimir Katkovnik, Alessandro Foi, Karen Egiazarian, and Jaakko Astola. Directional varying scale approximations for anisotropic signal processing. In *2004 12th European Signal Processing Conference*, pages 101–104. IEEE, 2004.

[27] Jens Krommweh. Tetrolet transform: A new adaptive haar wavelet algorithm for sparse image representation. *Journal of Visual Communication and Image Representation*, 21(4):364–374, 2010.

[28] Sang Gyu Kwak and Jong Hae Kim. Central limit theorem: the cornerstone of modern statistics. *Korean journal of anesthesiology*, 70(2):144, 2017.

[29] Kaiqun Leng. An improved non-local means algorithm for image denoising. In *2017 IEEE 2nd International Conference on Signal and Image Processing (ICSIP)*, pages 149–153. IEEE, 2017.

[30] Mukesh C Motwani, Mukesh C Gadiya, Rakhi C Motwani, and Frederick C Harris. Survey of image denoising techniques. In *Proceedings of GSPX*, pages 27–30, 2004.

[31] Caio A Palma, Fábio AM Cappabianco, Jaime S Ide, and Paulo AV Miranda. Anisotropic diffusion filtering operation and limitations-magnetic resonance imaging evaluation. *IFAC Proceedings Volumes*, 47(3):3887–3892, 2014.

[32] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on pattern analysis and machine intelligence*, 12(7):629–639, 1990.

[33] Mariana Poderico, Sara Parrilli, Giovanni Poggi, and Luisa Verdoliva. Sigmoid shrinkage for bm3d denoising algorithm. In *2010 IEEE International Workshop on Multimedia Signal Processing*, pages 423–426. IEEE, 2010.

[34] Xiwen Qin, Yang Yue, Xiaogang Dong, Xinmin Wang, and ZhanSheng Tao. An improved method of image denoising based on wavelet transform. In *2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering*, volume 5, pages 167–170. IEEE, 2010.

[35] Raghuram Rangarajan, Ramji Venkataramanan, and Siddharth Shah. Image denoising using wavelets. *Wavelet and Time Frequencies*, 14, 2002.

[36] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[37] Thomas Sikora. Low complexity shape-adaptive dct for coding of arbitrarily shaped image segments. *Signal Processing: Image Communication*, 7(4-6):381–395, 1995.

[38] Thomas Sikora and Bela Makai. Shape-adaptive dct for generic coding of video. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(1):59–62, 1995.

[39] Gilbert Strang. The discrete cosine transform. *SIAM review*, 41(1):135–147, 1999.

[40] Xiaoli Sun, Min Li, and Weiqiang Zhang. An improved image denoising model based on the directed diffusion equation. *Computers & Mathematics with Applications*, 61(8):2177–2181, 2011.

[41] Rajiv Verma and Rajoo Pandey. Non local means algorithm with adaptive isotropic search window size for image denoising. In *2015 Annual IEEE India Conference (INDICON)*, pages 1–5. IEEE, 2015.

[42] Francesco Voci, Shigeru Eiho, Naozo Sugimoto, and H Sekibuchi. Estimating the gradient in the perona-malik equation. *IEEE Signal Processing Magazine*, 21(3):39–65, 2004.

[43] Zhou Wang, Alan C Bovik, Hamid R Sheikh, Eero P Simoncelli, et al. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[44] Dong Yang and Jian Sun. Bm3d-net: A convolutional neural network for transform-domain collaborative filtering. *IEEE Signal Processing Letters*, 25(1):55–59, 2018.

[45] Jianjun Yuan. Improved anisotropic diffusion equation based on new non-local information scheme for image denoising. *IET Computer Vision*, 9(6):864–870, 2015.

[46] Tieyong Zeng and Michael K Ng. On the total variation dictionary model. *IEEE Transactions on Image Processing*, 19(3):821–825, 2010.

[47] Hua Zhong, Ke Ma, and Yang Zhou. Modified bm3d algorithm for image denoising using nonlocal centralization prior. *Signal Processing*, 106:342–347, 2015.

[48] Binyi Zou, Hui Liu, Zhenhong Shang, and Ruixin Li. Image denoising based on wavelet transform. In *2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pages 342–344. IEEE, 2015.

Curriculum Vitae

Name:	Zaied Zaman
Post-Secondary Education and Degrees:	Bachelor of Science in Electrical & Electronic Engineering Islamic University of Technology 2011-2015
Related Work Experience:	Teaching Assistant The University of Western Ontario 2017 - 2019 Research Assistant Image Processing Dr. Mahmoud El-Sakka The University of Western Ontario 2017 - 2019
Honors and Awards:	Western Graduate REsearch Scholarship The University of Western Ontario 2017 - 2019