12-10-2015 12:00 AM

BM3D Image Denoising using Learning-Based Adaptive Hard Thresholding

Farhan Bashar
The University of Western Ontario

Supervisor
Dr. Mahmoud R. El-Sakka
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

BM3D IMAGE DENOISING USING LEARNING-BASED ADAPTIVE HARD THRESHOLDING

(Thesis format: Monograph)

by

Farhan Bashar

Graduate Program in Computer Science

A thesis submitted in partial fulfillment

of the requirements for the degree of

Masters of Science

The School of Graduate and Postdoctoral Studies

The University of Western Ontario

London, Ontario, Canada

Abstract

Image denoising is an important pre-processing step in most imaging applications. *Block Matching and 3D Filtering (BM3D)* is considered to be the current state-of-art algorithm for additive image denoising. But this algorithm uses a fixed hard thresholding scheme to attenuate noise from a 3D block. Experiments show that this fixed hard thresholding deteriorates the performance of *BM3D* because it does not consider the context of corresponding blocks. In this thesis, we propose a learning based adaptive hard thresholding method to solve this issue. Also, *BM3D* algorithm requires as an input the value of the noise level in the input image. But in real life it is not practical to pass as an input such noise level. In this thesis, we also attempt to automatically estimate the level of the noise in the input image. Experimental results demonstrate that our proposed algorithm outperforms *BM3D* in both objective and subjective fidelity criteria.

Keywords: Image denoising, additive white Gaussian noise, Block Matching and 3D Filtering (BM3D), adaptive threshold, classification, random forest classifier, Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM) Index

Acknowledgments

At this happy moment, 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 remember and broaden my delicate respect to all the professors of The University of Western Ontario who helped me through many courses to build my background for this dissertation. Without their sincere care, the understanding of Image Processing fundamentals would have been impossible for me.

Finally, I acknowledge the support of my friends, family members, my loving mother and research group members throughout this long tiring period. Their inspiration and encouragement strengthened me in my tough time to keep focused.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A digital image is defined as a two dimensional discrete function. The value of this function at any particular point is called the gray level or intensity of the image at that location. For gray-scale images, this intensity value is limited between 0-255.

During the image acquisition or the transmission processes, sometimes the image sensor produces mechanical and electronic interference which generate some unexpected or random brightness information, known as noise. For general purpose images, the noise may be negligible and often ignored. However, there are sophisticated imaging applications such as face recognition, object tracking, medical imaging, satellite imaging and segmentation where even the small amount of noise degrades the performance of these applications. Thus the need of proper image denoising algorithm has grown with much interest. Image denoising is often considered as an important pre-processing step for various imaging applications.

Currently there is a lot of image denoising algorithms present to reduce various types of noise. Block Matching and 3D Filtering (BM3D) [1] is one such popular algorithm that reduces Additive White Gaussian Noise (AWGN) [2] from digital images and is considered to be the current state-of-art image denoising algorithm.

In terms of denoising performance, BM3D is considered to be the best denoising filter so far. It exhibits remarkable results as compared to other existing methods. BM3D works in

two identical steps. In first step, it generates a basic estimate of the noisy image using hard thresholding. Then in the second step, it generates Wiener coefficients from the basic estimated image and denoises the noisy image based on these coefficients.

Although BM3D achieves excellent performance in reducing additive white gaussian noise, it posses some limitations as well. Our main study will focus on finding these limitation and provide possible solutions for them. By solving these limitations we can achieve higher denoising performance than that of the original BM3D.

1.1 Thesis Contributions

The main contribution of our thesis is to improve the performance of BM3D and provide solutions over the limitations of BM3D. We will focus on the following points:

- BM3D algorithm relies on a user provided noise level for each noisy image, which is not practical for real time systems. This noise level is very important for estimating the denoised image. We will incorporate a noise level estimation mechanism without hampering the performance to make this as an automated system.

- In BM3D algorithm, a hard thresholding is used for any block of the noisy image. We will illustrate that this thresholding scheme depends on image block's texture and noise level. Tuning this threshold can improve the performance of the BM3D scheme. Thus we will propose a learning-based adaptive hard thresholding mechanism where each block utilizes different threshold based on it's context.

1.2 Thesis Outline

We have formalized our thesis into five chapters including this introductory discussion as **Chapter 1**. In **Chapter 2**, we discuss various image denoising schemes, focusing primarily on BM3D and its variants. In **Chapter 3**, different classification techniques are outlined. In

Chapter 4, we describe our proposed method. In **Chapter 5**, the detailed experimental results are analyzed. Finally in **Chapter 6**, we conclude our thesis by providing possible extensions and future works.

Chapter 2

Image Denoising Background

In this chapter, we will review different types of noise that affect digital images and existing image denoising techniques. We will explain thoroughly the current state-of-art denoising algorithm and its limitations.

2.1 Image Noise

Noise is a random variation of brightness information in images. Usually noise is produced by the sensor or circuitry of imaging devices, i.e., scanner or digital camera. There are many variants of image noise. A brief introduction of some noise variants is given below.

2.1.1 Additive White Gaussian Noise

Additive noise refers to the noise signal which is independently added to the image signal. If $y(x)$ is a original signal where $x \in X$ is a 2D spatial coordinate that belongs to the image domain and $\eta(x)$ is the noise signal, then the received signal can be represented as:

$$z(x) = y(x) + \eta(x). \tag{2.1}$$

White noise is a random signal with a constant power spectral density, that means it has

uniform power across the frequency band [2].

Gaussian noise is a statistical noise having a probability density function equal to that of the normal distribution. The probability density function of a Gaussian noise is [3]:

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

Here i represents the gray level, μ the mean value and σ the standard deviation. Figure 2.1 shows the probability density function for Gaussian noise.

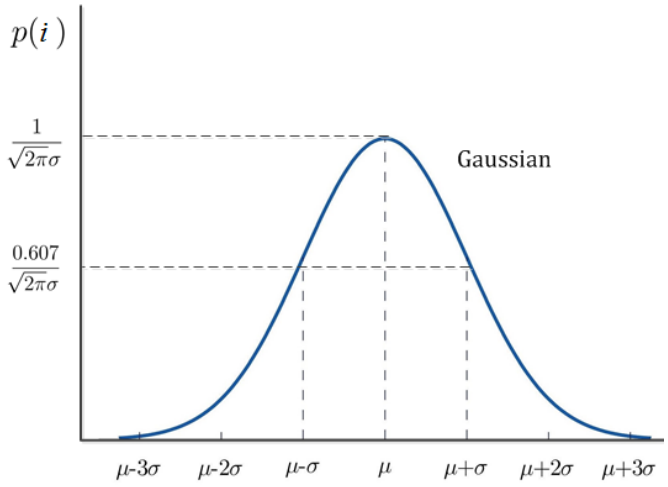


Figure 2.1: Distribution of Gaussian Noise

Thus, Additive White Gaussian Noise (AWGN) refers to the signal which is independent, has constant power spectral density and follows Gaussian (normal) distribution. Figure 2.2 shows an example of applying additive white gaussian noise (using matlab imnoise function).

2.1.2 Salt-and-Pepper Noise

Salt-and-pepper noise or spike noise is kind of impulsive noise spreaded across the entire image where the intensity of the affected pixel changes to minimum possible gray value (0) or maximum possible gray value (255). This type of noise can be caused by analog-to-digital conversion errors, bit errors in transmission [4]. Figure 2.3 shows an example of applying salt-

(a) (b)

Figure 2.2: Example of Additive White Gaussian Noise. (a) Original Lena Image. (b) AWGN added with $\mu = 0$ and $\sigma = 50$

and-pepper noise over the Lena image. Popular salt-and-pepper noise reduction techniques are Median Filter and Adaptive Median Filter [2].

(a) (b)

Figure 2.3: Example of Salt-and-pepper Noise. (a) Original Lena Image. (b) Salt-and-Pepper added with density = 0.25

2.1.3 Speckle Noise

Speckle noise is an example of multiplicative noise which means, the unwanted random signal (noise) gets multiplied with the original signal. Speckle noise usually found in Synthetic Aperture Radar (SAR) or medical ultrasound and optical coherence tomography images. A generic formula of speckle noise can be represented as:

$$z(x) = y(x) \times \eta(x). \tag{2.3}$$

Here $y(x)$ is the original signal where $x \in X$ is a 2D spatial coordinate that belongs to the image domain and $\eta(x)$ is the noise signal. Figure 2.4 shows an example of applying speckle noise over the Lena image. Some of the popular speckle noise reduction techniques are Lee Filter [5], Kuan Filter [6], Frost Filter [7] and SRAD [8].

(a) (b)

Figure 2.4: Example of Speckle Noise. (a) Original Lena Image. (b) Speckle noise added with mean = 0 and variance = 0.08

Since this thesis is focused on reducing additive white Gaussian noise, we will only focus on such denoising techniques.

2.2 Image Denoising Techniques

Image denoising can be performed either in the spatial domain or in the frequency domain. In spacial domain, denoising is done by applying filter directly on the intensity values of the image. On the other hand, in frequency domain technique, an image is transformed into the frequency domain and then the filtering operations are performed. The resulting denoised signal is transformed back into the spatial domain.

2.2.1 Spatial Domain Filter

In spatial domain filtering techniques, an image is denoised based on the statistics of its spatial information, i.e., the intensity values of the image. This class of image denoising is simple because there is no cost for domain transformation. Some of the popular spatial domain filtering algorithms are discussed below.

Mean Filter

Mean filter is one of the simplest method for spatial image denoising. It reduces the amount of intensity variation between one pixel and the next. In this method, a $M \times N$ window is selected around a particular pixel and the arithmetic average of all the intensity values of the neighboring pixel is calculated. This value is then replaced with the center pixel of that window [2]. Thus that particular pixel gets the mean value of its neighborhood. This filter smooths the image and reduces noise. Typically, the value of $M \times N$ is 3×3, 5×5, 7×7 or 11×11. The formula of mean filter is given below:

$$\hat{I}(x, y) = \frac{1}{MN} \sum_{x,y \in \eta} I(x, y) \tag{2.4}$$

Here $I(x, y)$ is the original pixel, $\hat{I}(x, y)$ is the denoised pixel and η is the $M \times N$ local neighborhood of each pixel.

Median Filter

Median filter is another simple image denoising method similar to mean filter. Instead of taking the arithmetic mean around a center pixel, median filter takes the median among the neighborhood pixels and replace the center pixel value with this value. Median filter is very effective for images having salt & pepper noise. As any gray image contaminated with salt-and-pepper noise only have noisy pixels valued 0 and 255, taking median around the neighborhood of a noisy pixel and replacing noisy pixel with the median value reduces the noise. But it also smooths the edges. Median filter is further modified to adaptive median filter by changing the window size based on image noise level [2].

Gaussian Smoothing

Gaussian smoothing operator is a 2-D convolution operator that is used to reduce noise. It is similar to the mean filter, but it uses a different kernel that represents the shape of a Gaussian (bell-shaped). Equation (2.4) can be changed to Gaussian smoothing by including the Gaussian kernel, as shown in the following equation.

$$\hat{I}(x,y) = \frac{1}{MN} \sum_{x,y \in \eta} G(x,y) \times I(x,y), \tag{2.5}$$

where $G(x,y)$ is the Gaussian kernel, as shown below:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{2.6}$$

where σ^2 is the variance.

Anisotropic Diffusion

In typical spatial domain image denoising techniques i.e, mean filter or Gaussian smoothing, every pixel is averaged by its surrounding pixels. The calculation does not take into account

whether the pixel is from a smooth region or an edge. So, edges become blurred in these methods. To avoid this problem, Perona and Malik [9] proposed an edge preserving image denoising technique called Anisotropic Diffusion. The main idea of this technique is to first identify whether a pixel is from a smooth region or an edge and then denoise it. The authors suggested that a smoothing algorithm should have the following criteria:

1. **Causality:** No spurious detail should be generated while passing from finer to coarser scales.

2. **Immediate Localization:** At each resolution, the region boundaries should be sharp and coincide with the "semantically meaningful" boundaries at that resolution.

3. **Piece-wise Smoothing:** At all scales, intra-region smoothing should occur preferentially over inter-region smoothing.

Based on the smoothing criteria, they proposed an algorithm for image denoising where iterative Gaussian smoothing is applied, while preserving the edges. The main features of this algorithm can be divided into two parts:

1. Using non-linear smoothing function instead of linear smoothing.

2. Using anisotropic diffusion instead of isotropic diffusion

Linear smoothing treats every pixel with the exact same convolution whereas non-linear smoothing treats a pixel with varying intensity, depending on its neighborhood qualities. In general, if a pixel is part of an edge, then little smoothing is applied, otherwise full smoothing is applied. To detect whether a pixel is a part of an edge, we need an estimation E, which is calculated using the following equation:

$$E(x, y, t) = \nabla I(x, y, t) \tag{2.7}$$

where ∇I is the gradient of the image. Based on the estimation E, a coefficient $c(x, y, t)$ is generated which will control how much smoothing is needed to apply.

$$c(x, y, t) = g(||E||) \tag{2.8}$$

where $g(||E||)$ is a monotonous decreasing function. Perona and Malik proposed two variations of this function:

$$g(||E||) = e^{-(\frac{||E||}{k})^2} \tag{2.9}$$

and

$$g(||E||) = \frac{1}{1 + (\frac{||E||}{k})^2} \tag{2.10}$$

Anisotropic Diffusion algorithm for image denoising iteratively blurs an image by calculating the divergence of its gradient (Laplacian of the image). But instead of calculating the Laplacian directly from the image, the coefficient $c(x, y, t)$ is multiplied with the gradient and then divergence is calculated. At time t, the next blurring function is:

$$I^{t+1} = I^t + \lambda[c_N \times \nabla_N I + c_S \times \nabla_S I + c_E \times \nabla_E I + c_W \times \nabla_W I] \tag{2.11}$$

where $0 \leq \lambda \leq 1/4$ for the numerical scheme to be stable. N, S, E, W are mnemonic subscripts for North, South, East and West. ∇I indicates the nearest neighbor difference between two pixels in a particular direction. Figure 2.5 shows the performance of Anisotropic diffusion.

Non-Local Means

Non-Local Meas (NLM) is one of the most successful spatial domain image denoising scheme. It follows patch based denoising method. Instead of filtering a single pixel based on its neighboring pixels, it works with patches within a defined window. In this algorithm, a block/patch is defined around a particular pixel, also referred to as the reference patch. A search window is also defined around the pixel where similar patches to the reference patch is searched. The

(a)

(b)

(c)

Figure 2.5: Performance of Anistropic diffusion (a) Original Lena image. (b) AWGN added with $\sigma = 60$ (PSNR=12.57) (c) Denoised image using Anisotropic diffusion, iteration = 12 (PSNR=24.75)

similar patches are given a weight based on its similarity with the reference patch. The center pixel of the reference patch is then denoised by a weighted averaging using center pixels of the similar patches [10, 11].

Let a discrete noisy image $v = \{v(i)|i \in I\}$, where I is the input image. The estimated

(a)

(b)

(c)

Figure 2.6: Performance of Non-Local Means (a) Original Lena image. (b) AWGN added with $\sigma = 60$ (PSNR=12.57) (c) Non-Local Means denoised image (PSNR=27.25)

denoised value is computed using the following equation.

$$NL[v](i) = \sum_{j \in I} w(i, j)v(j) \tag{2.12}$$

where $w(i, j)$ is the weight which depends on the similarity between patches center i and j

which are $v(N_i)$ and $v(N_j)$, respectively. The weight is computed as,

$$w(i,j) = \frac{1}{Z(i)} e^{-\frac{\|v(N_i)-v(N_j)\|_2^2}{h^2}} \tag{2.13}$$

Here $Z(i)$ is a normalization constant and h is a smoothing kernel width which controls decay of the exponential function. Figure 2.6 shows the performance of Non-Local Means algorithm.

Many improvements have been suggested on the Non-Local Means algorithm in past few years. Recently, Rehman and Wang proposed SSIM-Based Non-Local Means Algorithm [12] which uses SSIM [13] between two patches to calculate their similarity. This algorithm achieved much improvement over the original Non-Local Means algorithm. Also there are many variants of Non-Local Means algorithm. Adaptive Non-Local Means [14], Non-Local Medians [15] are popular ones. Maruf and El-Sakka introduced t-test in Non-Local Means for reducing feature space [16]. Alkinani and El-Sakka [17] applied Non-Local Means in stereo image denoising.

2D Adaptive Wiener Filter

2D Adaptive Wiener filter or Wiener2 filter took the idea of Wiener filter and applied it in the spatial domain. Basically it estimates the image noise and filters the image based on this noise [18, 19].

At first 2D Adaptive Wiener filter estimates the local mean and variance around each pixel respectively:

$$\mu(a,b) = \frac{1}{MN} \sum_{x,y \in \eta} I(x,y) \tag{2.14}$$

and

$$\sigma^2(a,b) = \frac{1}{MN} \sum_{x,y \in \eta} I^2(x,y) - \mu^2(a,b), \tag{2.15}$$

where η is the $N \times M$ local neighborhood of each pixel in the image I. This filter then creates a

(a)

(b)

(c)

Figure 2.7: Performance of 2D Adaptive Wiener Filter (a) Original Lena image. (b) AWGN added with $\sigma = 60$ (PSNR=12.57) (c) Wiener2 denoised image (PSNR=24.35)

pixel-wise Wiener filter using these estimates, as shown in Equation (2.16).

$$\hat{I}(x,y) = \mu + \frac{\sigma^2(x,y) - v^2}{\sigma^2(x,y)}(I(x,y) - \mu(x,y)) \tag{2.16}$$

Here v^2 is the noise variance. If the noise variance is not given, it is calculated by averaging all

the local estimated variance. The noise variance calculation is shown below:

$$v^2 = \frac{1}{MN} \sum_{x,y \in \eta} \sigma(x, y), \tag{2.17}$$

where $\sigma(x, y)$ is the local estimated variance calculated from Equation (2.15). Figure 2.7 shows the performance of 2D Adaptive wiener filter.

2.2.2 Frequency Domain Filter

Spatial domain filter work with the intensity of the image. In frequency domain filtering techniques, the input image $f(x, y)$ is transformed to frequency domain $F(x, y)$ and then a filter $H(x, y)$ is used. The denoised signal $G(k, l) = F(k, l) \times H(k, l)$ is then inverse transformed to get back the output image.

Low Pass Filter

Low pass filter is one of the basic frequency domain filter. This filter allows to pass the signal with frequencies lower than the cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency. As a result, the output image is smoothed because edge information remains in high frequencies. Thus this filter is not an edge preserving filter. The low pass filter can be represented, as shown below:

$$H(k, l) = \begin{cases} 1, & \text{if } \sqrt{k^2 + l^2} < f_{cut-off} \\ 0, & \text{if } \sqrt{k^2 + l^2} > f_{cut-off} \end{cases} \tag{2.18}$$

Wiener Filter

Wiener filter is one of the most popular frequency domain filter. It tries to estimate the noise from a degraded image and denoise it based on the estimation. The main objective of Wiener filter is to reduce Mean-Square Error between two signals [20]; also known as *Minimum Mean*

Square Error or *Least Square Error* filter. If f is the uncorrupted image and $\hat{f}$ is the noisy image then the expectation of Wiener filter is to have minimum MSE between them.

$$e^2 = E(f - \hat{f})^2 \tag{2.19}$$

As Wiener filter works in frequency domain, the estimated signal after Wiener filtering is $\hat{F}(u, v)$.

$$\hat{F}(u, v) = \Big(\frac{H^*(u, v)}{|H(u, v)|^2 + \frac{S_n}{S_f}}\Big)G(u, v) \tag{2.20}$$

Here $H(u, v)$ is a degradation function e.g., Gaussian blur. $H^*(u, v)$ is the complex conjugate of the degradation function. S_n and S_f are power spectrum of noise and power spectrum of uncorrupted image respectively. $G(u, v)$ is the observation. Usually the notation $K = S_n/S_f$ is used as S_n and S_f are uniform.

2.3 Block Matching and 3D Filtering (BM3D) Algorithm and its Extensions

In recent years Block Matching and 3D Filtering (BM3D) becomes the most popular image denoising technique. Dabov et al. [1] first proposed the idea in 2006 [21] and explained it thoroughly in 2007. The algorithm was then analyzed by Lebrun [22] and implemented an open-source version of this algorithm. BM3D achieves excellent performance in reducing *Additive White Gaussian Noise* (AWGN). It has achieved state-of-the-art denoising performance in terms of both objective and subjective assessment. In this section, we study the algorithm of BM3D and its extensions and improvements. We also discuss the limitations of this algorithm.

2.3.1 Algorithm

BM3D follows the concept of patch-based denoising mechanism, first introduced in Non-Local Means (NLM) algorithm [10]. BM3D also extended it by denoising an image using two identical steps. In first step, a basic estimate of the noisy image is generated. This basic estimate is then passed to the second step to generate the final denoised image. The block diagram of this algorithm is shown in Figure 2.8.

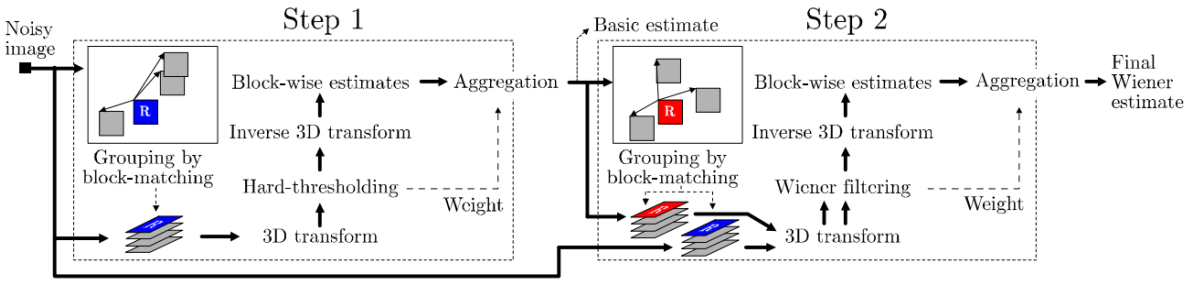


Figure 2.8: BM3D Block Diagram

Let us consider the noisy observation of a true signal $y(x)$ is $z(x) = y(x) + \eta(x)$ where $x \in X$ is a 2D spatial coordinate that belongs to the image domain and $\eta(x) \sim N(0, \sigma^2)$ is white Gaussian noise of variance σ^2. After the first step, a basic estimation is $\hat{y}^{basic}$ and after second step the final estimation is $\hat{y}^{final}$. In the following portions of this section, we study detailed description of these two steps.

BM3D First Step

The first step of BM3D is known as *hard thresholding* step because here a hard thresholding is used to eliminate noise from the image. First, a block/patch is defined around a particular pixel, also referred to as the reference patch. A search window is also defined around the pixel where similar patches to the reference patch are searched. As the initial image is noisy, calculating similarity between noisy blocks may degrade the performance. So, first the blocks are filtered by using 2D transformation and then the obtained coefficients are hard thresholded. Next the

Euclidean-distance between the reference block and each of the other blocks are calculated. This similarity measurement is called d-distance. Equation (2.21) shows how the d-distance between two blocks is calculated.

$$d(Z_{x_R}, Z_x) = \frac{\Upsilon(\tau^{ht}_{2D}(Z_{x_R})) - \Upsilon(\tau^{ht}_{2D}(Z_x))}{(N^{ht}_1)^2} \tag{2.21}$$

where Υ is the hard-thresholding operator and τ^{ht}_{2D} denotes the normalized 2D linear transform. Using d-distance values, similar noisy patches are grouped together into a set, $S^{ht}_{x_R}$, as shown below:

$$S^{ht}_{x_R} = d(Z_{x_R}, Z_x) \leq \tau^{ht}_{match}, \tag{2.22}$$

where the fixed τ^{ht}_{match} is the maximum d-distance which is used for calculating similarity between blocks. From this set, the noisy blocks are grouped together into a 3D block which we denote $Z_{S^{ht}_{x_R}}$. A 3D linear transform is applied on this 3D block and hard thresholding is applied on the obtained coefficients, called collaborative filtering. This thresholding attenuates the noise. An inverse 3D transform is applied to get back to the spatial domain. Equation (2.23) shows the block-wise estimation of a 3D block.

$$\hat{Y}^{ht}_{S^{ht}_{x_R}} = {\tau^{ht}_{3D}}^{-1}(\Upsilon(\tau^{ht}_{3D}(Z_{S^{ht}_{x_R}}))), \tag{2.23}$$

where Υ is a hard-threshold operator and $\hat{Y}^{ht}_{S^{ht}_{x_R}}$ is stacked block-wise estimation of noisy blocks in set $S^{ht}_{x_R}$. The final step in the first step of BM3D is to aggregate all the estimated blocks together. Each of the estimated sets contains a number of blocks and these blocks contains one or more same pixel locations. That means, a single pixel can have more that one estimation. So to get the final estimation a weighted averaging is applied on the estimations. The global basic estimate $\hat{y}^{basic}$ is computed by a weighted average of the block-wise estimates $\hat{Y}^{ht}_{S^{ht}_{x_R}}$ shown

below:

$$\hat{y}^{basic} = \frac{\sum_{x_R \in X} \sum_{x_m \in S^{ht}_{x \in R}} w^{ht}_{x_R} \hat{Y}^{ht,x_R}_{x_m}(x)}{\sum_{x_R \in X} \sum_{x_m \in S^{ht}_{x \in R}} w^{ht}_{x_R} X_{x_m}(x)}, \tag{2.24}$$

where $w^{ht}_{x_R}$ is the weight of a group of block which is calculated by the Equation (2.25).

$$w^{ht}_{x_R} = \begin{cases} \frac{1}{\sigma^2 N^{x_R}_{ht}}, & \text{if } N^{x_R}_{ht} \geq 1 \\ 1 & \text{otherwise.} \end{cases} \tag{2.25}$$

The basic denoised estimate of the noisy image is then passed to the second step of BM3D to generate the final estimation.

BM3D Second Step

The basic estimate $\hat{y}^{basic}$ from the input image is assumed to be significantly attenuated. This image is used in the second step as reference true noise free image of the original image. Second step of BM3D is identical to the first step. At first the noisy blocks are grouped together. But here instead of using the thresholding-based d-distance, normalized squared l^2-distance is used to find similarity between the basic estimated blocks. For any reference block, the basic estimated blocks are stacked together in a set using:

$$S^{wie}_{x_R} = \frac{\|\hat{y}^{basic}_{x_R} - \hat{y}^{basic}_{x_R}\|}{(N^{wie}_1)^2} \leq \tau^{wie}_{match} \tag{2.26}$$

Lets denote $\hat{y}^{basic}_{S^{wie}{}_{x_R}}$ as the stack of basic estimated blocks and $Z_{S^{wie}{}_{x_R}}$ as the stack of noisy blocks from original image. Wiener shrinkage coefficients are generated by applying 3D transform on the basic estimated group using:

$$W_{S^{wie}_{x_R}} = \frac{|\tau^{wie}_{3D}(\hat{y}^{basic}_{S^{wie}{}_{x_R}})|^2}{|\tau^{wie}_{3D}(\hat{y}^{basic}_{S^{wie}{}_{x_R}})|^2 + \sigma^2} \tag{2.27}$$

where σ^2 is the variance of the noisy image. The 3D transform coefficients of the noisy blocks are multiplied, element-by-element, with the Wiener shrinkage coefficients. This is called Wiener collaborative filtering, as shown in Equation (2.28). Inverse 3D transform is applied to get the estimated pixel values.

$$\hat{Y}^{wie}_{S^{wie}_{x_R}} = \tau^{wie^{-1}}_{3D}(W_{S^{wie}_{x_R}}\tau^{wie}_{3D}(Z_{S^{ht}_{x_R}})) \tag{2.28}$$

Here $\hat{Y}^{wie}_{S^{wie}_{x_R}}$ is the final block-wise estimation. These estimated blocks are then aggregated together to generate the final global estimated image $\hat{y}^{final}$. Aggregation is done using:

$$\hat{y}^{final} = \frac{\sum_{x_R \in X}\sum_{x_m \in S^{wie}_{x \in R}} w^{wie}_{x_R}\hat{Y}^{wie,x_R}_{x_m}(x)}{\sum_{x_R \in X}\sum_{x_m \in S^{wie}_{x \in R}} w^{wie}_{x_R}X_{x_m}(x)} \tag{2.29}$$

where $w^{wie}_{x_R}$ is weight of a reference block's group which is calculated as:

$$w^{wie}_{x_R} = \sigma^{-2}\|W_{S^{wie}_{x_R}}\|_2^{-2} \tag{2.30}$$

The performance of BM3D is better, compared to the previous denoising algorithms. Figure 2.9 shows the performance of BM3D.

2.3.2 Extensions and Improvements

BM3D achieves better Peak Signal-to-Noise Ratio (PSNR) as compared to any existing image denoising methods. It produces significant better result than the previous state-of-art image denoising algorithm, Non-Local Means (NLM). A number of improvements and extensions of BM3D has been introduced and still work is going on to improve the performance of BM3D.

After proposing the original BM3D based on gray scale image denoising, Dabov et al. extended their work to color image denoising [23] and video denoising [24]. They also introduced PCA into BM3D and proposed a new method called BM3D-SAPCA [25], where collaborative

Figure 2.9: Performance of BM3D (a) Original Lena image. (b) AWGN added with $\sigma = 60$ (PSNR=12.57) (c) Basic denoised image (PSNR=27.00) (d) Final denoised image (PSNR=28.27)

filtering is done on the PCA coefficients of 3D blocks. They also proposed a new method named BM4D which uses BM3D to filter volumetric data. Thus instead of having 3D blocks, here each time 4D blocks of data are filtered [26].

Zhang et al. [27] proposed a method for calculating adaptive hard threshold used in *d*-

distance of BM3D's first step . Here instead of using a single hard threshold, they adaptive generate the threshold value by using the gradient of the block, the SSIM between two blocks and the noise level. But they have achieved a little improvement over original BM3D.

Harold et al. [28] presented the idea of learning based image denoising. They used neural network to train large datasets and used multi-layer perceptron (MLP) method to denoise an image patch by patch. But for high textured noise, this method achieves less accuracy than BM3D.

Mittal et al. [29] also adapted a machine learning technique to predict the parameters for BM3D. In the second step of BM3D, the noise level of the image is used. They have generated the value of this parameter using a natural science statistics (NSS) method. A training is applied on huge datasets and using SSIM, the sigma value is predicted for test noisy image. This method needs a lot of time for training but the improvement is still not up to the bound.

Hasan [30] proposed an adaptive edge-guided BM3D to improve the performance of BM3D for higher noise levels. It detects a noisy pixel suing its neighborhood pixels statistics and uses a pre-filter in the fist step of BM3D. It achieves a little improvement over BM3D for higher noise levels but for lower noise levels the performance is same as the original BM3D.

Hasan and El-Sakka [31, 32] proposed a method for improving the result of BM3D by optimizing the wiener filter used in the algorithm. Instead of using MSE in the objective function of the Wiener filtering step of BM3D, they have used SSIM. Thus the objective function of Wiener function changed to maximizing the SSIM between the denoised and true image. This method achieves comparatively better performance than other improvements mentioned above.

2.3.3 Limitations

Though BM3D is currently the state-of-art image denoising algorithm, it still has limitations and the performance can be improved. We have studied the BM3D algorithm in detail and identified the persisting limitations of this algorithm. Some of the limitations are reported below:

1. **Noise Level:** In the implementation of BM3D algorithm, we have found that the actual noise variance of the image is provided to the algorithm which is used in both first and second step. In real time systems, the actual noise variance is not practical to be provided as an input.

2. **Fixed Hard Thresholding:** In the first step of BM3D algorithm, a hard threshold is used to attenuate noise from the 3D blocks. This threshold is fixed for all the blocks. Using fixed threshold value deteriorates the performance of this algorithm because for smooth region and textured region thresholding value should not be the same.

3. **Poor Performance for Higher Noise Levels:** BM3D has poor performance in higher noise levels as compared to lower noise levels. It happens because of using fixed parameters. If the parameters can be made adaptive then for higher noise levels, it can achieve better performance.

4. **Parameterized Setup:** Regardless the type of transform used in BM3D, it used a fixed set of parameter. But each transform has its own best parameter. Therefore using same parameters for all kind of transform need to be reviewed to evaluate whether these parameters are the best or it is needed to be adaptively adjusted.

Chapter 3

Classification Background

In our proposed method, we have used a classifier. Therefore this chapter discusses some of the existing classification techniques.

3.1 Classification

Classification is the problem of identifying the category of a new observation on the basis of a training set of data containing observations whose category is known. It is considered an instance of supervised learning, where a training set of correctly identified observations is available. An algorithm that implements classification is known as a classifier. This classifier is a mathematical function which maps input data to a category. Figure 3.1 shows how a classifier works.

3.1.1 Classification Schemes

Different algorithms of classification has been proposed over the past years [33]. These algorithms were proposed, based on solving particular classification problems. The popular classifiers are Linear classifiers (i.e., Fisher's linear discriminant, Logistic regression, Naive Bayes classifier), Support Vector Machines, Quadratic classifiers, K-Nearest Neighbor, Boost-

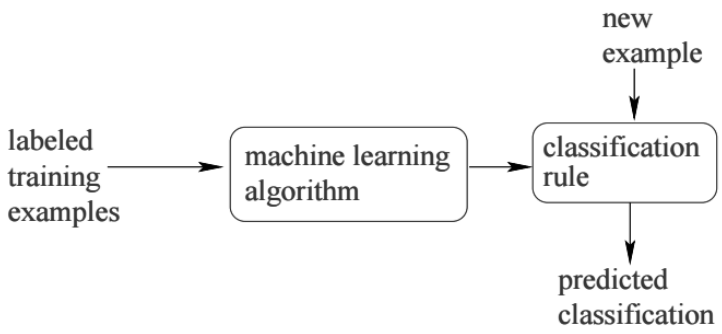


Figure 3.1: Block diagram of a Classifier

ing, Decision Trees (i.e., Random Forest) and Neural Networks. In the following section, we will provide brief description of some of these classifiers.

Linear Discriminant Analysis (LDA)

Fisher's linear discriminant or generally known as Linear discriminant analysis (LDA) is a method to find a linear combination of features that characterizes or separates two or more classes. LDA is used for dimensionality reduction also.

LDA seeks to reduce dimensionality while preserving as much of the class discriminatory information as possible. Fisher suggested maximizing the difference between the means, normalized by a measure of the within-class scatter. LDA considers maximizing the following objective:

$$J(w) = \frac{w^T S_B w}{w^T S_w w} \tag{3.1}$$

where S_B is the between classes scatter matrix and S_w is the within classes scatter matrix. The definition of these matrices are:

$$S_B = \sum_c (\mu_c - \bar{x})(\mu_c - \bar{x})^T \tag{3.2}$$

$$S_W = \sum_c \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c)^T) \tag{3.3}$$

where $\bar{x}$ is the overall mean of the data-cases. LDA produces $c - 1$ feature projections which

are used for classification. LDA performs very well if the features can be separated by a linear hyperplane. Performance of LDA is not satisfactory if the discriminatory information of features resides not in the mean but in the variance of the data.

Native Bayes

A Naive Bayes classifier is a simple probabilistic classifier based on applying Bayes' theorem (from Bayesian statistics) with strong (naive) independence assumptions. In simple terms, a naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is related to the presence (or absence) of any other feature [33].

Let T be a training set of samples, each with its own class labels. There are k classes, $C_1, C_2, ..., C_k$. Each sample is represented by an n-dimensional vector, $X = \{x_1, x_2, ..., x_n\}$, depicting n measured values of the n attributes, $A_1, A_2, ..., A_n$, respectively. Given a sample X, the classifier will predict that X belongs to the class having the highest posteriori probability, conditioned on X. That is, X is predicted to belong to the class C_i if and only if $P(C_i|X) > P(C_j|X)$ for $1 \leqslant j \leqslant k, j \neq i$. Thus the class that maximizes $P(C_i|X)$ is predicted. The class C_i for which $P(C_i|X)$ is maximized is called the maximum posteriori hypothesis. By Bayes theorem:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)} \tag{3.4}$$

Given data sets with many attributes, the naive assumption of class conditional independence is made. This presumes that the values of the attributes are conditionally independent of one another, given the class label of the sample. Mathematically it is:

$$P(X|C_i) = \prod_{k=1}^{n} P(x_K|C_i) \tag{3.5}$$

The probabilities $P(x_1|C_i), P(x_2|C_i), ..., P(x_n|C_i)$ is estimated from the training set.

A class prior may be calculated by assuming equiprobable classes, or by calculating an estimate for the class probability from the training set. To estimate the parameters for a feature's

distribution, one must assume a distribution or generate non-parametric models for the features from the training set. The assumptions on distributions of features are called the event model of the Naive Bayes classifier. For discrete features multinomial and Bernoulli distributions are popular. If the training data contain a continuous attribute, Gaussian distribution works very well [34].

Support Vector Machine (SVM)

Support Vector Machine (SVM) are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification analysis. SVM performs an implicit mapping of data into a higher dimensional feature space, and finds a linear separating hyper-plane with maximal margin to separate the data [33]. Given a training set of labeled examples $T = (X_i, l_i), i = 1, 2, ..., L$ where $X_i = \{x_1, x_2, ..., x_n\}$ is a feaure vector and $l_i \in \{-1, 1\}$, a new test data x is classified by:

$$f(x) = sign(\sum_{i=1}^{L} \alpha_i l_i K(x_i, x) + b) \tag{3.6}$$

where α_i are Lagrange multipliers of dual optimization problem, b is a bias or threshold parameter, and K is a kernel function [34]. The training samples x_i with $\alpha_i > 0$ are called the support vectors, and the separating hyperplanes maximizes the margin with respect to these support vectors. Given a non-linear mapping function ϕ that transforms the input data to the higher dimensional feature space, kernels have the form $K(x_i, x_j) = \phi(x_i, \phi(x_j))$. Of the various kernels found in the literature, linear, polynomial and radial basis function (RBF) kernels are the most frequently used.

SVM makes binary decisions and multi-class classification can be achieved by adopting the one-against-rest technique, which trains a binary classifier for each class to discriminate one class from all others, and outputs the class with the largest count.

K-Nearest Neighbors

K-Nearest Neighbors (KNN) algorithm is a non-parametric method used for classification. In this algorithm, the training dataset is used only to populate a sample of the search space with instances whose class is known [33]. No actual model or learning is performed during this phase. When an instance whose class is unknown is presented for evaluation, the algorithm computes its k closest neighbors, and the class is assigned by voting among those neighbors. Different distance metrics can be used, depending on the nature of the data. The Euclidean distance is typically used for continuous variables.

The training phase for *KNN* consists of simply storing all known instances and their class labels. A tabular representation can be used, or a specialized structure such as a kd-tree. If we want to tune the value of k and/or perform feature selection, n-fold cross-validation can be used on the training dataset. The testing phase for a new instance t, given a known set I is as follows:

1. Compute the distance between t and each instance in I.

2. Sort the distances in increasing numerical order and pick the first k elements.

3. Compute and return the most frequent class in the k nearest neighbors, optionally weighting each instance's class by the inverse of its distance to t.

KNN algorithm is very simple in implementation and robust with regard to the search space; for instance, classes don't have to be linearly separable. The main disadvantage is that testing is very expensive as the distance to all known instances need to be calculated. For large number of dataset, it requires a lot time for classifying the data [33].

AdaBoost

AdaBoost, short for "Adaptive Boosting", is a classification algorithm based the idea of creating a highly accurate prediction rule by combining many relatively weak and inaccurate rules.

It is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers, however, it can be less susceptible to the overfitting problem than other learning algorithms.

AdaBoost refers to a particular method of training a boosted classifier. It is an iterative method, where initially equal weights are assigned to each training example. At successive iterations, the weight of missclassified examples is increased. This forces the algorithm to concentrate on examples that have not been classified correctly so far. AdaBoost use the following discriminant function:

$$g(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \tag{3.7}$$

where each h_t is a weak learner that takes an object x as input and returns a real valued result indicating the class of the object. The final classifier is the sign of the discriminant function.

$$F_{final} = sign[g(x)] \tag{3.8}$$

AdaBoost can achieve similar classification results with much less tweaking of parameters or settings over SVM. The user only needs to choose which weak classifier might work best to solve their given classification problem and the number of boosting rounds that should be used during the training phase.

Random Forests

Random forests are an ensemble of learning method for classification operated by constructing a multitude of decision trees at training time and outputting the class that is the maximum votes of the classes of the individual trees [33]. Random Forests grow many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree votes for that class. The forest chooses the classification having the most votes (over all the trees in the forest). Each tree is grown as follows [35]:

1. If the number of cases in the training set is N, sample N cases at random - but with replacement, from the original data. This sample will be the training set for growing the tree.

2. If there are M input variables, a number $m<<M$ is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during forest growing.

3. Each tree is grown to the largest extent possible. There is no pruning.

The forest error rate depends on two things:

1. The correlation between any two trees in the forest. Increasing the correlation increases the forest error rate.

2. The strength of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the forest error rate.

Random forest is one of the most efficient classifier where a large number of samples are used. Some features of this algorithm is [35]:

- It is excellent in accuracy among current algorithms.

- It runs efficiently on large data bases.

- It can handle thousands of input variables without variable deletion.

- It gives estimates of what variables are important in the classification.

- It generates an internal unbiased estimate of the generalization error as the forest building progresses.

Neural Networks

Artificial neural networks (ANNs) are a family of models inspired by biological neural networks and are used to estimate or approximate functions that can depend on a large number of inputs and are generally unknown. Neural networks have been trained to perform complex functions in various fields, including pattern recognition, identification, classification, speech, vision abnd control systems. It can also be trained to solve problems that are difficult for human beings [33].

Neural networks are typically organized in layers. Layers are made up of a number of interconnected nodes which contain an activation function. Patterns are presented to the network via the input layer, which communicates to one or more hidden layers where the actual processing is done via a system of weighted connections. The hidden layers then link to an output layer where the answer is output as shown in Figure 3.2.

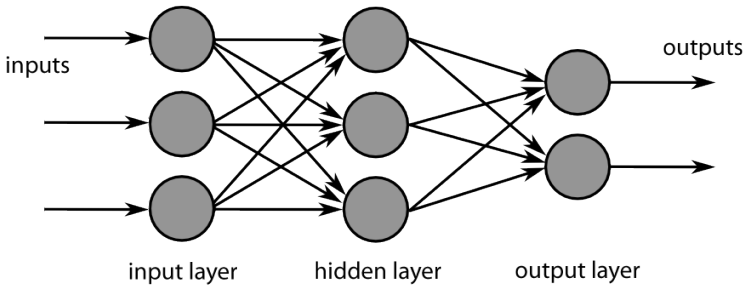


Figure 3.2: Layers of Neural Network

Most ANNs contain some form of learning rule which modifies the weights of the connections according to the input patterns that it is presented with. In a sense, ANNs learn by example as do their biological counterparts. Although there are many different kinds of learning rules used by neural networks, this demonstration is concerned only with one; the delta rule. The delta rule is often utilized by the most common class of ANNs called backpropagational neural networks (BPNNs). Backpropagation is an abbreviation for the backwards propagation of error. Backpropagation performs a gradient descent within the solution's vector space towards a global minimum along the steepest vector of the error surface. The global

minimum is that theoretical solution with the lowest possible error.

Once a neural network is trained to a satisfactory level it may be used as an analytical tool on other data. To do this, the user no longer specifies any training runs and instead allows the network to work in forward propagation mode only. New inputs are presented to the input pattern where they filter into and are processed by the middle layers as though training were taking place, however, at this point the output is retained and no backpropagation occurs. The output of a forward propagation run is the predicted model for the data which can then be used for further analysis and interpretation.

Chapter 4

Methodology

In Chapter 2, we have given background for different image denoising techniques. Also we have rigorously studied the BM3D Image denoising algorithm and pointed out some persisting limitations of this state-of-the-art algorithm. In this chapter, we will first try to find solutions of these limitations and then propose a new algorithm. Our proposed algorithm need a classifier. We have reviewed some of the classifers in Chapter 3.

4.1 Main Idea of Proposed Algorithm

The BM3D has several limitations. In our thesis, we focus on two limitations: The *Noise Level* and *Fixed Hard Thresholding* (See Section 2.3.3). We have performed several experiments to investigate these two limitations and developed our algorithm on those results. Below is a detailed description of how we found our solution to these limitations.

4.1.1 Automated Noise Estimation

While using the authors' provided Matlab software of BM3D, we have found that the true noise level (standard deviation) of input noisy image is provided as input. Also, we have observed that if the noise level is changed a little, the performance of the algorithm varies a lot. So,

we can say that the performance of the BM3D algorithm provided by the authors of BM3D depends heavily on the value of the input parameter.

In our proposed algorithm, we created an automated noise estimation system where the noise level of an input image is calculated first. We found that our noise level estimation is very close to the actual noise level (see Section 4.3.1), hence eliminating such input without degrading the performance.

For the noise level calculation, we have taken an idea from 2D adaptive Wiener filtering (wiener2) algorithm [18, 19]. In this algorithm, image noise level is estimated based on the local variance of the image. The local mean and variance around each pixel are calculated using:

$$\mu(a,b) = \frac{1}{MN} \sum_{x,y \in \eta} I(x,y) \tag{4.1}$$

and

$$\sigma^2(a,b) = \frac{1}{MN} \sum_{x,y \in \eta} I^2(x,y) - \mu^2(a,b), \tag{4.2}$$

where η is the $N \times M$ local neighborhood of each pixel in the image I. The noise variance is then calculated by averaging all the local estimated variance using Equation (4.3)

$$v^2 = \frac{1}{MN} \sum_{x,y \in \eta} \sigma^2(x,y), \tag{4.3}$$

where $\sigma^2(x,y)$ is the local estimated variance calculated from Equation (4.2). From this noise variance given in Equation (4.3), we can calculate the standard deviation (σ) of the noisy image. We have experimented using various neighborhood and found that 2×2 neighborhood produces the closest estimation to the actual noise level.

4.1.2 Context-based Hard Thresholding

In the first step of BM3D algorithm, a hard thresholding is applied on the 3D noisy blocks during the collaborative filtering. This hard thresholding attenuates the noise of corresponding

blocks (see Section 2.3.1).

Note that, the hard threshold operator Υ is fixed for every block of the image.The BM3D used a fixed value, $\Upsilon = 2.7$. For images having noise level greater than 40 they have changed this value to $\Upsilon = 2.8$. This threshold is used to attenuate the noise of corresponding block. But using a fixed level of threshold value is not quite a good choice, as blocks with different properties should have different threshold values.

We started our experiment to figure out whether different blocks should use different threshold values or not. First, we took two different images, one with high texture and the other with smooth region and then applied same level of noise to both of them. Next we applied the BM3D algorithm to both of them using various threshold values and observed the best denoised image. We observed that different threshold values are used to generate the best denoised image in each case. Figure 4.1 shows the result of this experiment. For a highly textured image, we obtained the best denoised image for $\Upsilon = 2.4$ and for a smooth image, we obtained the best denoised image with $\Upsilon = 3.1$. Both of these denoise image showed significant improvement over the original BM3D where fixed threshold value is used. Similar experiments were also performed using an image with various noise levels and it was observed that the best threshold value also changes with the noise level.

4.2 Detailed Proposed Algorithm

Our proposed algorithm is divided into two main parts: *training* and *testing*. A generic block diagram of the algorithm is shown in Figure 4.2. A detailed description of the algorithm is given below.

4.2.1 Training

In the training phase, we developed 10 different classifiers for 10 different noise levels , σ = 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100. Here, image blocks are used as feature vectors and

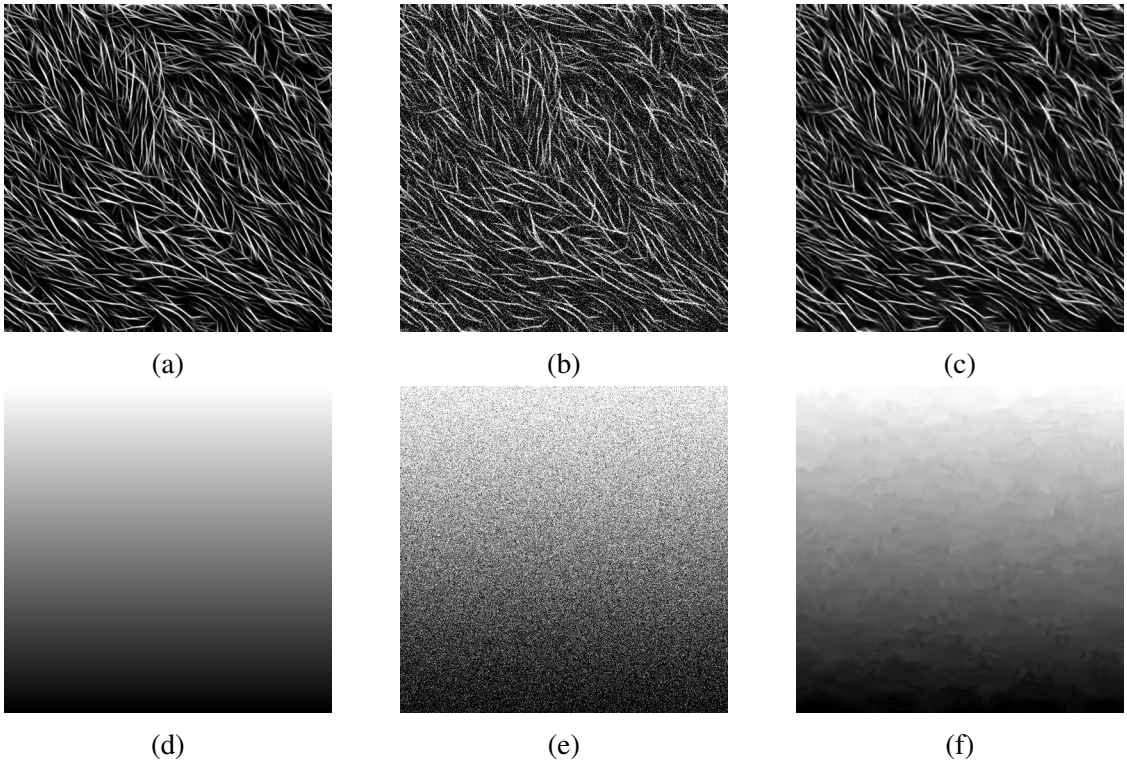


Figure 4.1: Performance of BM3D with different Threshold value (a) High Textured Image. (b) AWGN added with $\sigma = 60$ (c) Best Denoised with threshold = 2.4 (PSNR=21.44, Original BM3D PSNR=20.35) (d) Smooth Image. (e) AWGN added with $\sigma = 60$ (f) Best Denoised with threshold = 3.1 (PSNR=36.95, Original BM3D PSNR=35.29)

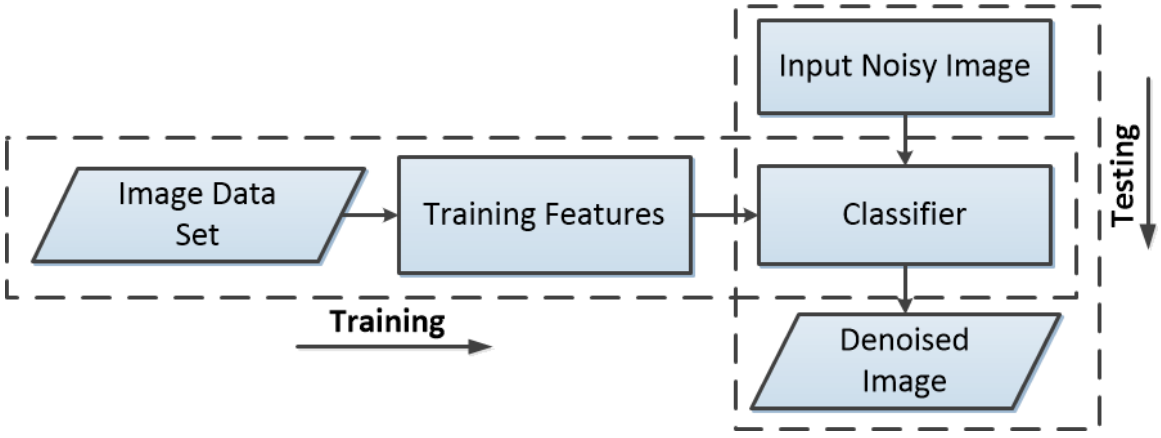


Figure 4.2: Block Diagram of Proposed Algorithm

their corresponding best threshold value as a label of that vector. The flowchart of this training part is shown in Figure 4.3. Below is a detailed description for training a single classifier.

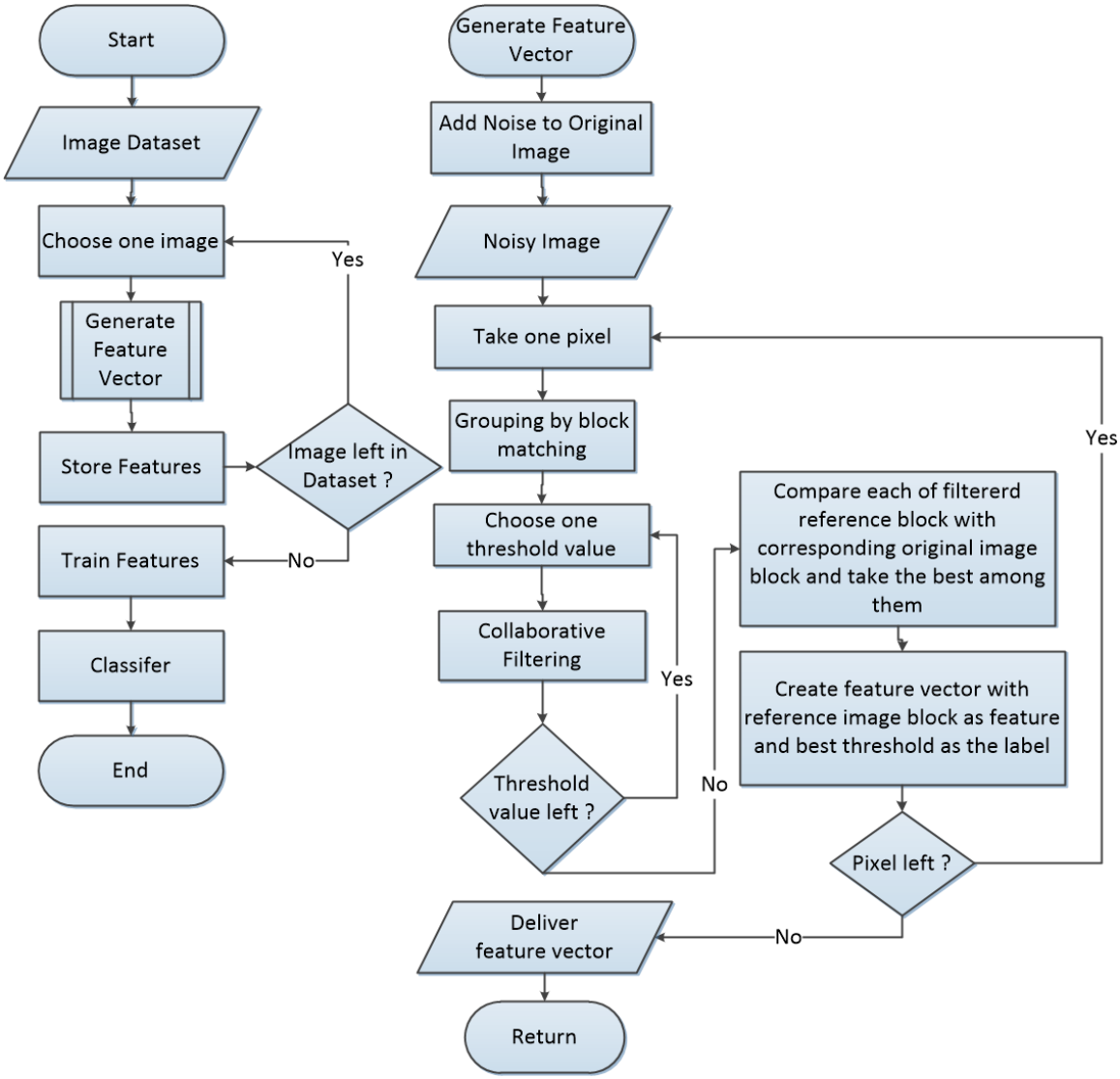


Figure 4.3: Flowchart of Training

Best Threshold Calculation

From the idea of context based hard thresholding, see Section 4.1.2, we know that different types of image blocks should have different threshold values. So, we conducted experiments on finding the best threshold value for any reference block.

1. To build a classifier for noise level n, we have applied AWGN with $\sigma = n$ to all input images from the database.

2. The BM3D image denoising algorithm is applied to these noisy images generated from

the previous step. During collaborative filtering in the first step, we have used various threshold values for each block. In our experiments, we have used 22 different threshold values, ranging from 1.7 to 3.8 with step size of 0.1.

3. The 22 denoised blocks (based on applying various threshold values) are compared with the corresponding block in the original true image and the best among them is selected. The comparison is done based on squared euclidean distance between these two blocks.

4. The threshold corresponding to the best denoised block is considered to be the best threshold value of that particular block.

5. Every block and their best threshold value is stored in a matrix.

After taking all of the best denoised block during the collaborative filtering operation, we found the generated output to be significantly better than the output of original BM3D algorithm in terms of PSNR and visual quality. To assess the performance, we figured from the results of experiment that the PSNR is improved by about 3 dB (on average) when the best threshold is used, for a given noise level. Figure 4.4 shows the comparison among Best-BM3D denoised image and original BM3D denoised image. From the subjective viewpoint, it is also clear that using different threshold values for every block generates a better denoised image than the original BM3D algorithm. But in real life, the original image is not available, so we need an intelligent algorithm to predict the best threshold values for any particular block. Thus we will use various blocks and their best threshold values to train a classifier.

Feature Generation

1. For any input image, each of the image block is considered as a feature vector and their corresponding best threshold value is considered as label of that feature.

2. Since we are considering blocks of size 7×7, thus each feature consists of 49 noisy pixel values.

(a) (b)

(c) (d)

Figure 4.4: Performance of Denoised Image (a) Original Lena image (b) Noise image with σ = 100 (c) Original BM3D Algorithm (PSNR = 25.95) (d) Best BM3D Image (PSNR = 29.21)

3. In an image of size $M \times N$, a total $M \times N$ feature vectors are generated, each of length 49 features.

4. To reduce memory and time complexities, we have only taken 10% of the total features, by taking one feature and leaving the following nine features.

Training Features

1. The generated features and their corresponding labels are used to train a classifier that will be used in the next part of the algorithm.

In our experiment, we have used different classification techniques namely, Naive Bayes, SVM, K-Nearest Neighborhood and Random Forest to find out the best classifier for our algorithm. Based on their performance we have decided to use Random Forest (RF) classification algorithm.

4.2.2 Testing

In the testing phase, the classifier developed in the training phase is used to generate the appropriate threshold to be used to denoise the input image. Figure 4.5 shows the flowchart for testing. A detailed description of the testing phase is given step by step below.

Noise Calculation and Classifier Selection

1. From the test image, initial noise level is calculated using the algorithm described in Section 4.1.1.

2. The noise is then rounded to the nearest multiple of 10. If the value is more than 100, it is clipped at 100.

3. From the 10 trained classifiers, we choose our classifier based on the noise level.

Feature Vector Generation

1. The noisy input image is divided into several blocks of size 7×7. These 49 noisy pixels are considered the feature vector of a single block.

2. Feature vectors are generated from all of the blocks of the noisy image.

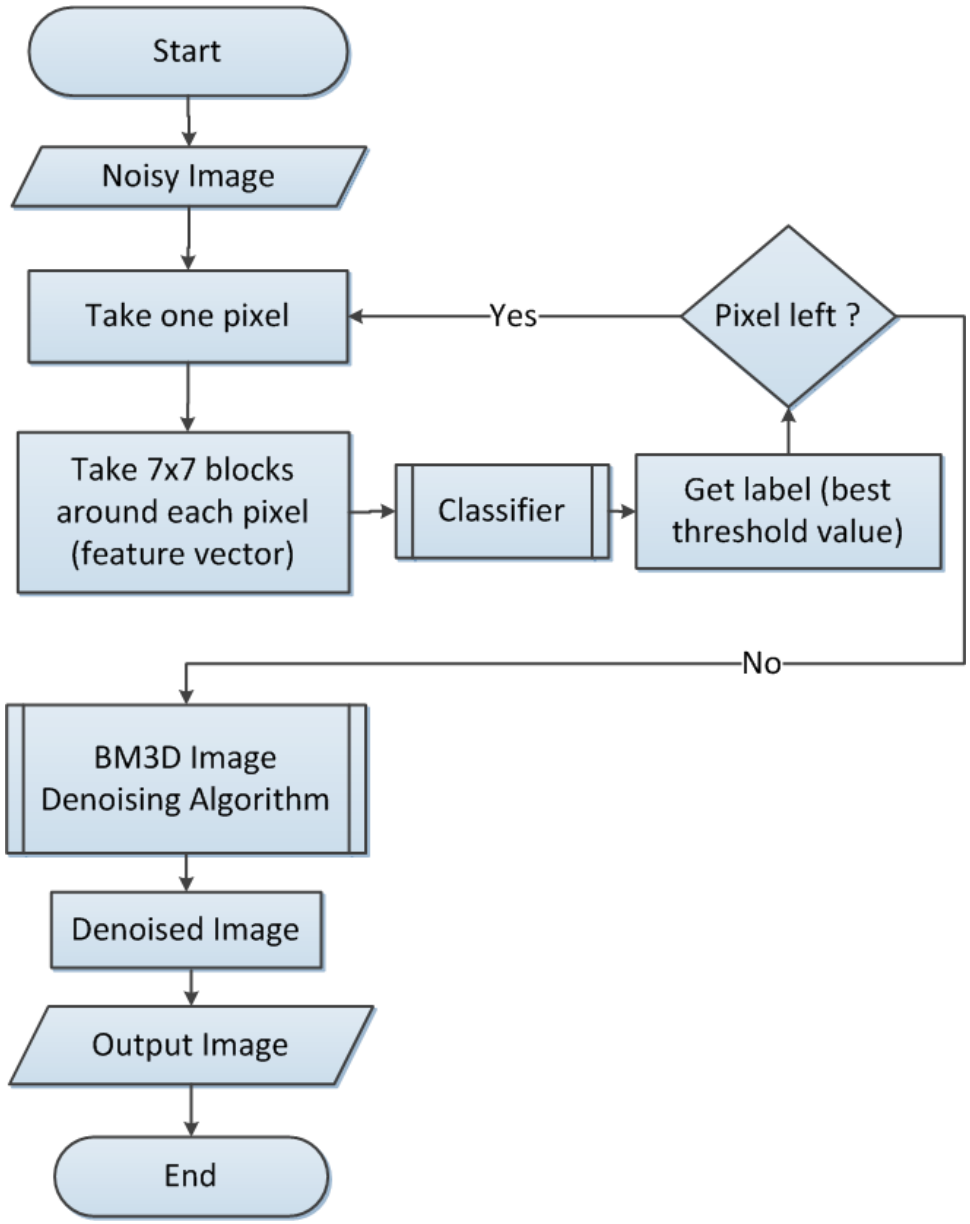


Figure 4.5: Flowchart of Testing

Classification

1. The feature set is passed to the classifier.

2. The classifier will return the label of each feature vector.

3. Each of the noisy blocks are then assigned their corresponding best threshold value.

Output Generation

1. The noisy image is filtered using BM3D image denoising algorithm with our predicted noise level where in the collaborative filtering step, adaptive thresholding is applied.

2. All other operations will remain the same as in the original BM3D. After the second step of BM3D, our final denoised image is generated.

4.3 Parameters Selection

In our proposed method, we have a few parameters which need to be adjusted. We have done several experiments to find the best values for any particular variable. Below is a detailed description of how these parameters are selected.

4.3.1 Window Size for Noise Estimation ($\eta \times \eta$)

To find the best window size in our noise estimation step (see Section 3.1.1), we have experimented on our training image set. This experiment is done in three steps. First, we applied 10 different noise levels to a black image and estimate the noise level. We found that the noise estimation method estimates the exact noise level. Then we estimated the noise level of a noise free image and found that the noise level is close to zero. Then we applied 10 different noise levels to these images and estimated the noise level using Equation (4.3). We used different windows sizes to find out the best window size value. It is observed that for window size, 2×2, our predicted noise level is almost close to the noise applied on these images. Table 4.1 shows these results. Figure 4.6 shows the graph of the experimental results.

4.3.2 Threshold Range Selection

In our proposed method, we have used 22 predefined threshold values which are used in the collaborative filtering step of BM3D. From these threshold values, we picked the best threshold

Table 4.1: Comparing estimated noise with true noise based on different window size

Noise Level	Window Size ($\eta \times \eta$)					
	2×2	3×3	5×5	7×7	9×9	11×11
10	**11.21**	14.19	16.79	18.89	20.68	22.25
20	**20.15**	22.39	24.12	25.63	26.98	28.20
30	**30.58**	31.64	32.88	34.01	35.04	35.99
40	**40.13**	41.24	42.20	43.09	43.91	44.67
50	**50.24**	51.00	51.77	52.50	53.17	53.81
60	**60.22**	60.83	61.48	62.09	62.67	63.21
70	**70.19**	70.71	71.27	71.80	72.30	72.77
80	**80.18**	80.62	81.11	81.57	82.01	82.43
90	**90.15**	90.55	90.98	91.39	91.79	92.16
100	**100.12**	100.50	100.88	101.25	101.61	101.95

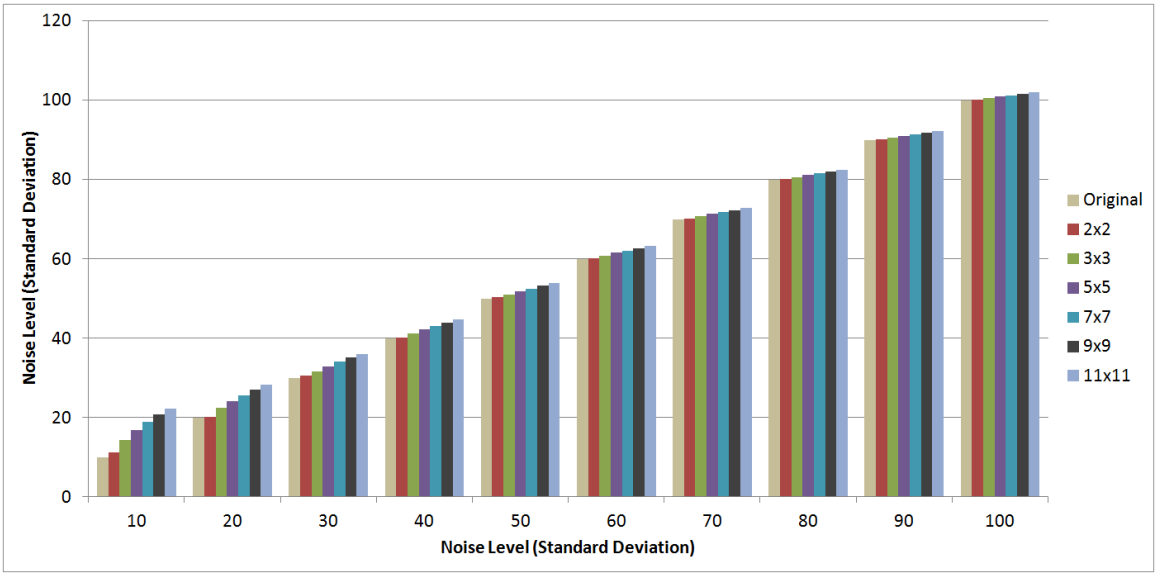


Figure 4.6: Comparing estimated noise with true noise based on different window size

value for a particular block. We have performed experiments based on different threshold ranges to find the range which provides the best image. Table 4.2 shows the result of best image based on different threshold ranges. It can be observed that if we increase the threshold range, the performance increases. But using 22 and 26 threshold values produces almost same results. However, taking 26 threshold values increases the number of class labels. So, we have taken 22 classes to reduce the time complexity. Figure 4.7 shows the graph representation of these results.

Table 4.2: PSNR comparison for different threshold range

Noise Level	Threshold Range (Total Number of Threshold Values)					
	2.7 (1)	2.3-3.1 (7)	2.1-3.4 (14)	1.9-3.6 (18)	**1.7-3.8 (22)**	1.5-4.0 (26)
10	34.45	35.42	35.95	36.39	**36.91**	36.93
20	31.21	32.27	32.82	33.32	**33.95**	33.98
30	29.39	30.51	31.08	31.60	**32.30**	32.33
40	27.95	29.43	30.05	30.58	**31.28**	31.31
50	27.05	28.12	28.70	29.25	**29.97**	30.02
60	26.25	27.33	27.91	28.47	**29.20**	29.25
70	25.56	26.67	27.26	27.81	**28.55**	28.61
80	24.97	26.10	26.69	27.25	**27.99**	28.05
90	24.45	25.60	26.19	26.75	**27.49**	27.56
100	23.98	24.47	25.75	26.30	**27.05**	27.12

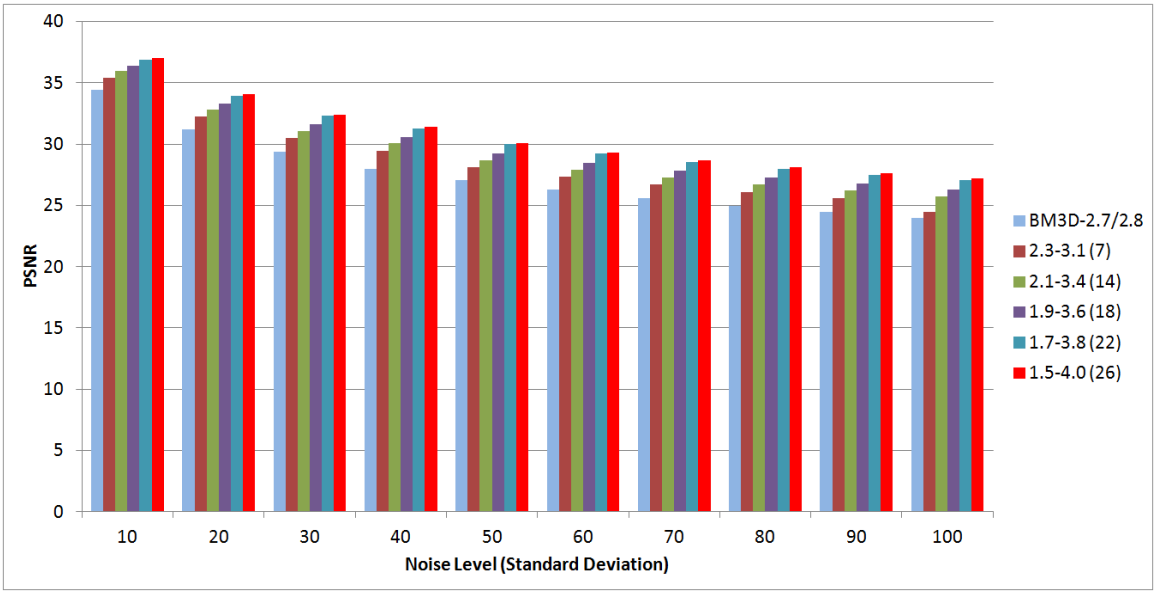


Figure 4.7: PSNR comparison for different threshold range

4.3.3 Classifier selection

Our proposed algorithm requires a classifier to train. For finding the best classifier, we have tested our method using various classifiers, such as Naive Bayes, Support Vector Machine (SVM), K-Nearest Neighbor, Decision Tree and Random Forest. Our experiment shows that using Random Forests provides the best accuracy based on PSNR of the denoised image. Figure 4.8 shows the performance of average denoising result on our test images. The Best BM3D column represents the average PSNR of best image where the actual image is known. All the

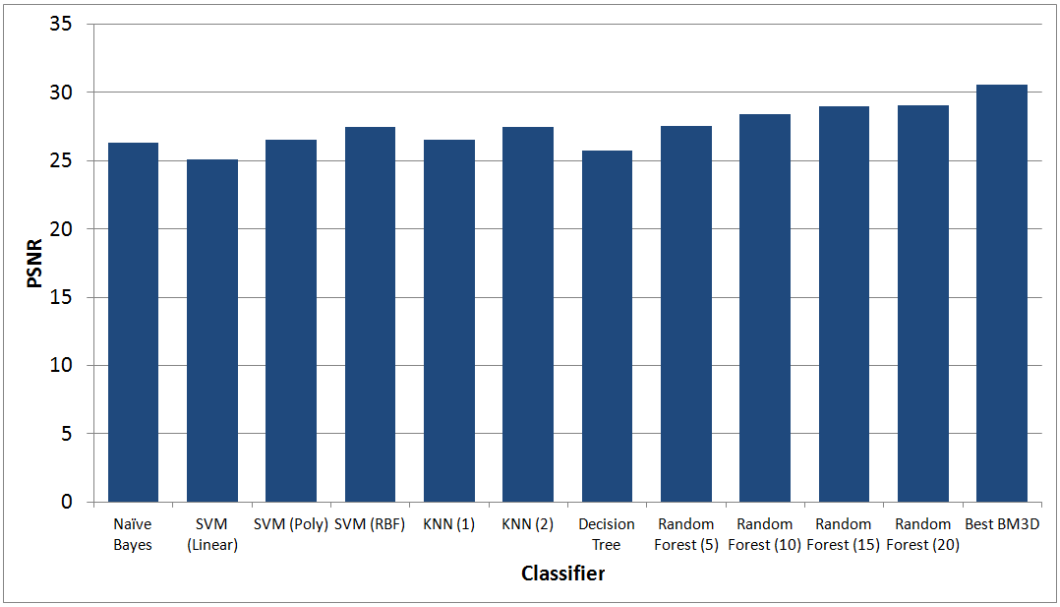


Figure 4.8: Performance comparison using different classifiers

rest column represents average PSNR based on different classification algorithm. We can observe that Random Forests with 15 and 20 trees provides the best output. The average PSNR for using 20 trees is 29.02 whereas the average PSNR with 15 trees is 28.98. So, to reduce the time and memory complexity, we have chosen Random Forest with 15 trees as our final classifier. Table 4.3 shows the whole result of this experiment.

Table 4.3: Performance comparison using different classifiers for testing set (Based on PSNR)

Noise Level	Naive Bayes	SVM (Linear)	SVM (Poly)	SVM (RBF)	K-NN (k)		RF (Number of Trees)				**Best BM3D**
					(1)	(5)	(5)	(10)	**(15)**	(20)	
10	33.25	31.90	33.48	34.25	33.54	34.34	34.42	35.11	**35.75**	35.81	**36.99**
20	29.97	28.26	29.87	31.41	29.96	30.89	31.19	32.03	**32.61**	32.66	**33.92**
30	28.17	26.88	28.40	29.46	28.41	29.25	29.37	30.17	**30.84**	30.87	**32.27**
60	26.75	25.61	27.01	28.08	27.07	28.04	27.94	28.81	**29.69**	29.71	**31.30**
50	25.84	24.67	26.09	26.99	26.12	27.01	27.06	27.90	**28.48**	28.53	**30.01**
60	25.07	23.78	25.17	26.18	25.32	26.12	26.28	27.16	**27.68**	27.73	**29.31**
70	24.42	23.31	24.68	25.50	24.74	25.67	25.59	26.52	**27.00**	27.05	**28.71**
80	23.82	22.89	24.32	24.85	24.33	25.18	25.01	25.96	**26.42**	26.46	**28.20**
90	23.31	22.21	23.62	24.27	23.24	24.19	24.48	25.46	**25.91**	25.93	**27.75**
100	22.88	21.26	22.63	23.65	22.69	23.68	23.90	25.03	**25.44**	25.47	**27.36**
Average	26.35	25.08	26.53	27.46	26.54	27.44	27.52	28.41	**28.98**	29.02	**30.58**

Chapter 5

Experimental Results and Analysis

In this chapter we discuss the details of our experiments and the obtained results. Before evaluating and analyzing our proposed algorithm, we also present a brief introduction to the experimental data sets and performance measurement metrics employed during experimentation. We compare the performance of our proposed method with other state-of-art image denoising techniques. At the end of this chapter, a set of denoised image using BM3D and our proposed method is provided to show the improvement of our method over BM3D.

5.1 Data Set

We have used two different sets of images in the thesis. One set is for training and the other one is for testing. For training purpose, we used Kodak image set which contains 24 different grayscale images, as shown in Figure 5.1. For testing, we have taken 10 grayscale images from the BM3D image set, as shown in Figure 5.2. Both of these sets contain images with both highly textured and smooth regions. As the main idea of our algorithm is to find different threshold values for textured and smooth blocks, a combination of both textured and smooth images is good for achieving average results.

(a) Beach

(b) Bike

(c) Bridge

(d) Cottage

(e) Door

(f) Flower

(g) Raft

(h) Girl

(i) Hats

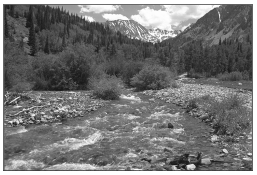
(j) House2

(k) Houses

(l) Windows

(m) Island

(n) Lake



(o) Landscape

(p) Lighthouse2

(q) Woman

(r) Parrot

(s) Plane

(t) Lighthouse

(u) Boat2

(v) Sail

(w) Statue

(x) Model

Figure 5.1: Training Image set (Kodak Image set)

(a) Cameraman

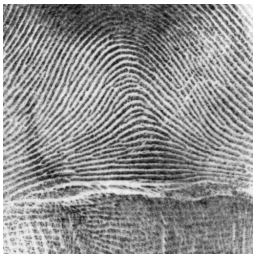
(b) House

(c) Peppers

(d) Lena

(e) Barbara

(f) Boat



(g) Fingerprint

(h) Man

(i) Couple

(j) Hill

Figure 5.2: Test Image set (Subset of BM3D Test Image set)

5.2 Performance Measurement Metrics

The performance of our experiments are measured using both subjective and objective fidelity criteria. Objective fidelity criteria gives us blind results, which is good for understanding the differences between the outputs and for subjective fidelity criteria, we observed how human eyes perceive the denoising performance.

5.2.1 Objective Fidelity Criteria

The objective fidelity criteria is defined as *blind quality assessment* since it just takes into account the intensity values of the image to be assessed. Peak Signal to Noise Ratio (PSNR) and Structural Similarity (SSIM) Index are the most widely used objective measurement techniques

for evaluating the performance of image denoising.

Peak Signal to Noise Ration (PSNR)

The peak signal to noise ratio (PSNR) represents the ratio between the maximum power of a signal to the noise which degrades the original image. This measure is based on the Mean Squared Error (MSE) which assesses the difference between the original image data and the degraded image data. In this measure, a higher value indicates a better denoised image. PSNR of a denoised image is calculated as:

$$PSNR = 10\log_{10}\left(\frac{MAX^2}{MSE}\right), \tag{5.1}$$

where MAX indicates the maximum intensity of the image. For a standard grayscale image it is 255. If $y(x)$ is an original image of size $M \times N$ and $\hat{y}(x)$ is the denoised image, then MSE is calculated as:

$$MSE = \frac{1}{M \times N}\sum_{x \in X}(y(x) - \hat{y}(x))^2. \tag{5.2}$$

Structural Similarity (SSIM) Index

Although PSNR is a good quality measure, it is sometimes seriously misleading because it does not consider anything other than intensity values [36]. To avoid such misleading error measure, Wang et al. [37, 38] proposed that modern image quality measurement metrics should not depend only on point-to-point distance, it should also consider the geometric or structural similarity between the images. Thus they proposed a new error measurement metric called Structural Similarity (SSIM). Their proposed method SSIM is defined in Equation (5.3).

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c2)} \tag{5.3}$$

Here x and y denote two blocks of the same position from the original image and the denoised image, respectively. μ_x and μ_y denote the arithmetic mean or average of x and y blocks, re-

spectively. σ_x^2 and σ_y^2 indicate the variance of x and y blocks, respectively while σ_{xy} is the co-variance between x and y. c_1 and c_2 are two variables to stabilize the division using a weak denominator and their values are $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$, respectively. Here, L is the dynamic range of the image and k_1 and k_2 are two constants whose values are $k_1 = 0.01$ and $k_2 = 0.03$, respectively. This measure is calculated block by block. Therefore, a mean of all blocks is used to represent the mean SSIM index value for the entire image.

5.2.2 Subjective Fidelity Criteria

We relied not only on these mathematical measures, but also on how human eyes perceive the denoising performance. We visually compared the denoised images generated by our proposed method and to that of BM3D. We also zoomed different significant regions of the images to better understand the denoising performance of these two methods.

5.3 Performance analysis

5.3.1 Parameter Setting

All parameters of original BM3D were kept same [1] except the threshold and noise level. For calculating the noise level, we have used a window size, $\eta \times \eta = 2 \times 2$. For threshold calculation, we used Random Forest classifier to predict our best threshold for a particular image block.

All of the results were produced using a 2.30 GHz Intel(R) Core i5 processor with 4GB RAM under Windows 7 OS. All of the test results are recorded and averaged after 10 runs.

5.3.2 Performance analysis using PSNR and SSIM

The performance of our proposed method is mainly compared with the original BM3D scheme. Also we compared our performance with another state-of-art denoising algorithm in the spatial domain namely, Non-Local Means (NLM). Tables 5.1 and 5.2 shows the average PSNR and

SSIM performance for all test images. The bolded values represent the best among all these algorithms. Figure 5.3 and 5.4 shows the graphs of average PSNR and SSIM comparison. From these results, we can easily observe that our proposed algorithm achieved better performance in both PSNR and SSIM metrics.

Table 5.1: Average PSNR comparison of test images

Noise Level	Noisy	NLM	BM3D	**Proposed**
10	28.13	33.10	34.45	**35.75**
20	22.10	29.92	31.21	**32.61**
30	18.58	27.79	29.39	**30.84**
40	16.08	26.23	27.95	**29.69**
50	14.15	25.01	27.05	**28.48**
60	12.56	24.03	26.25	**27.68**
70	11.22	23.21	25.56	**27.00**
80	10.06	22.52	24.97	**26.42**
90	9.04	21.90	24.45	**25.91**
100	8.13	21.35	23.98	**25.44**
Average	15.01	25.51	27.53	**28.35**

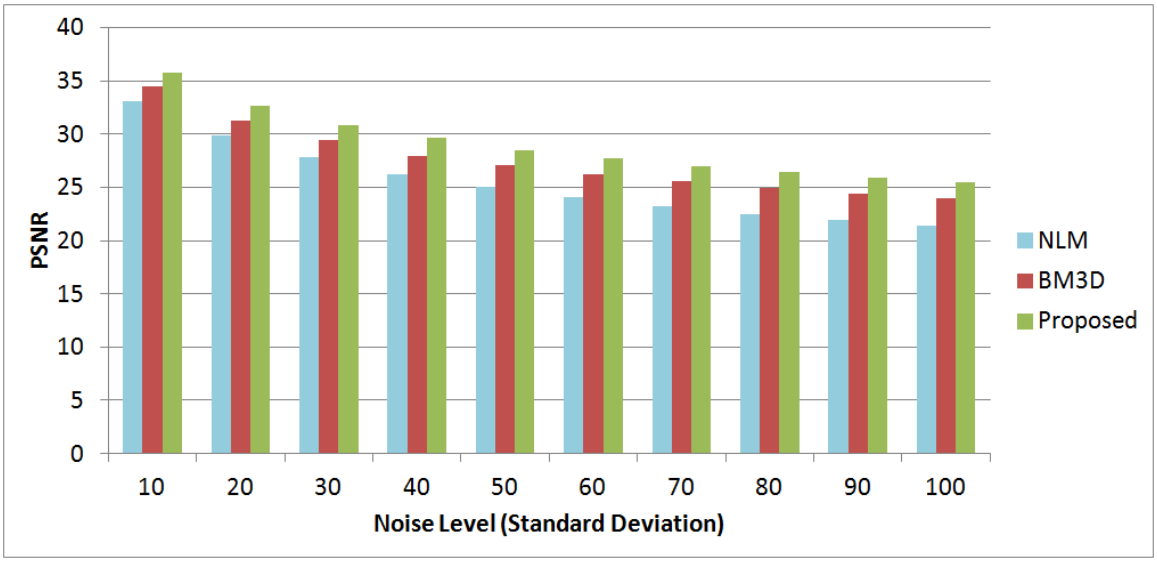


Figure 5.3: Average PSNR comparison of test images

Tables 5.5 to 5.14 shows the performance comparison for each of the test images shown in Figure 5.2. From these results, it is observed the that performance of our proposed method is significantly better than BM3D and NLM.

Table 5.2: Average SSIM comparison for test images

Noise Level	Noisy	NLM	BM3D	**Proposed**
10	0.693	0.895	0.920	**0.942**
20	0.445	0.812	0.865	**0.903**
30	0.310	0.734	0.824	**0.875**
40	0.229	0.660	0.786	**0.854**
50	0.175	0.592	0.759	**0.827**
60	0.138	0.531	0.733	**0.808**
70	0.112	0.478	0.709	**0.790**
80	0.091	0.431	0.687	**0.774**
90	0.075	0.390	0.666	**0.759**
100	0.063	0.355	0.647	**0.744**
Average	0.233	0.590	0.761	**0.828**

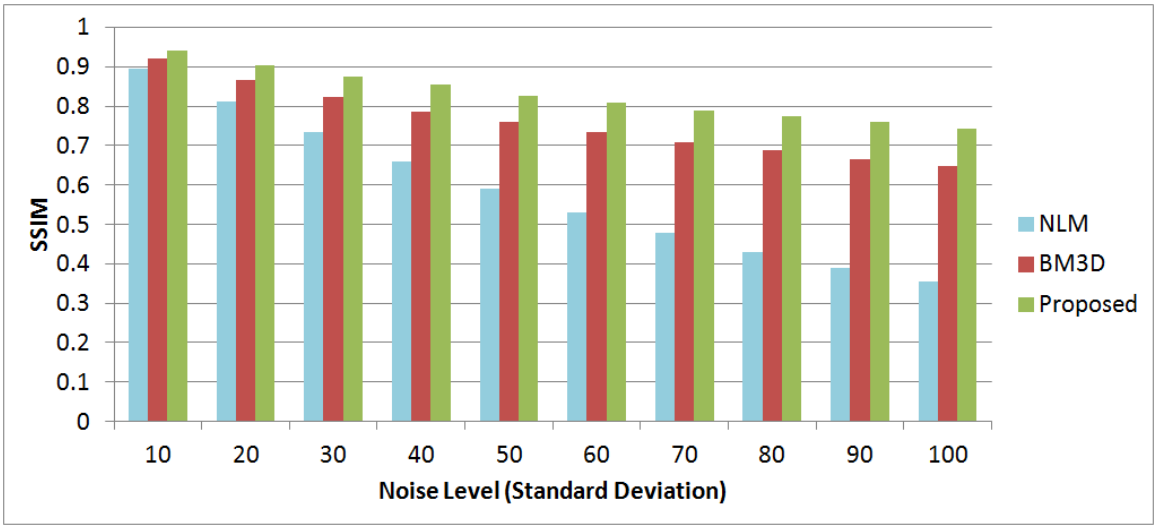


Figure 5.4: Average SSIM comparison of test images

As we have generated our output based on 10 runs on each of the testing image, Table 5.3 and 5.4 shows the standard deviation of the PSNR and SSIM output of corresponding methods to show that this 10 run is enough for testing the system.

Table 5.3: Standard Deviation of PSNR for test images

Noise Level	NLM	BM3D	**Proposed**
10	0.06	0.07	0.06
20	0.07	0.05	0.06
30	0.12	0.06	0.06
40	0.13	0.10	0.07
50	0.07	0.10	0.12
60	0.07	0.14	0.07
70	0.14	0.10	0.06
80	0.08	0.06	0.15
90	0.12	0.11	0.08
100	0.15	0.12	0.14
Average	0.10	0.09	0.09

Table 5.4: Standard Deviation of SSIM for test images

Noise Level	NLM	BM3D	**Proposed**
10	0.04	0.07	0.04
20	0.07	0.07	0.05
30	0.04	0.08	0.05
40	0.11	0.07	0.05
50	0.09	0.09	0.06
60	0.05	0.09	0.12
70	0.06	0.13	0.05
80	0.13	0.09	0.06
90	0.06	0.10	0.11
100	0.13	0.11	0.13
Average	0.08	0.09	0.07

Table 5.5: Performance comparison of Lena image

Noise Level	PSNR Comparison				SSIM Comparison			
	Noisy	NLM	BM3D	**Proposed**	Noisy	NLM	BM3D	**Proposed**
10	28.14	35.07	35.93	**37.22**	0.610	0.902	0.916	**0.938**
20	22.11	31.81	33.05	**34.46**	0.340	0.831	0.878	**0.912**
30	18.59	29.65	31.26	**32.78**	0.217	0.757	0.846	**0.892**
40	16.09	28.06	29.86	**31.64**	0.150	0.684	0.816	**0.876**
50	14.16	26.80	29.05	**30.54**	0.110	0.615	0.801	**0.857**
60	12.57	25.75	28.27	**29.76**	0.084	0.551	0.781	**0.842**
70	11.23	24.85	27.57	**29.10**	0.066	0.494	0.762	**0.829**
80	10.07	24.06	26.97	**28.53**	0.053	0.444	0.744	**0.815**
90	9.05	23.34	26.45	**28.01**	0.043	0.400	0.728	**0.803**
100	8.14	22.68	25.95	**27.54**	0.036	0.362	0.711	**0.790**
Average	15.02	27.21	29.44	**30.96**	0.171	0.604	0.798	**0.856**

Table 5.6: Performance comparison of Cameraman image

Noise Level	PSNR Comparison				SSIM Comparison			
	Noisy	NLM	BM3D	**Proposed**	Noisy	NLM	BM3D	**Proposed**
10	28.10	31.78	34.18	**35.48**	0.630	0.899	0.930	**0.950**
20	22.08	29.31	30.48	**32.00**	0.396	0.816	0.870	**0.912**
30	18.56	27.57	28.64	**30.20**	0.284	0.740	0.831	**0.884**
40	16.06	26.04	27.18	**28.99**	0.218	0.667	0.799	**0.867**
50	14.12	24.70	26.12	**27.58**	0.174	0.597	0.776	**0.842**
60	12.54	23.59	25.32	**26.75**	0.142	0.533	0.757	**0.826**
70	11.20	22.67	24.61	**26.04**	0.118	0.476	0.737	**0.813**
80	10.04	21.91	24.04	**25.44**	0.100	0.426	0.720	**0.799**
90	9.02	21.26	23.53	**24.92**	0.085	0.384	0.703	**0.786**
100	8.10	20.69	23.07	**24.46**	0.073	0.347	0.688	**0.774**
Average	14.98	24.95	26.72	**28.19**	0.222	0.588	0.781	**0.845**

Table 5.7: Performance comparison of Barbara image

Noise Level	PSNR Comparison				SSIM Comparison			
	Noisy	NLM	BM3D	**Proposed**	Noisy	NLM	BM3D	**Proposed**
10	28.14	33.67	34.98	**36.23**	0.715	0.926	0.941	**0.957**
20	22.11	30.09	31.78	**33.17**	0.476	0.850	0.904	**0.931**
30	18.59	27.61	29.81	**31.26**	0.340	0.767	0.867	**0.906**
40	16.09	25.84	27.99	**29.74**	0.254	0.687	0.821	**0.880**
50	14.16	24.56	27.23	**28.70**	0.195	0.614	0.793	**0.853**
60	12.57	23.59	26.28	**27.75**	0.153	0.550	0.757	**0.827**
70	11.23	22.83	25.47	**26.95**	0.123	0.495	0.725	**0.802**
80	10.07	22.20	24.79	**26.26**	0.100	0.448	0.695	**0.780**
90	9.05	21.65	24.16	**25.66**	0.083	0.407	0.667	**0.760**
100	8.14	21.14	23.62	**25.09**	0.070	0.372	0.642	**0.740**
Average	15.02	25.32	27.61	**29.08**	0.251	0.612	0.781	**0.844**

Table 5.8: Performance comparison of Boat image

Noise Level	PSNR Comparison				SSIM Comparison			
	Noisy	NLM	BM3D	**Proposed**	Noisy	NLM	BM3D	**Proposed**
10	28.14	32.87	33.92	**35.23**	0.689	0.866	0.888	**0.919**
20	22.11	29.70	30.88	**32.26**	0.422	0.773	0.825	**0.874**
30	18.59	27.65	29.12	**30.57**	0.284	0.694	0.778	**0.843**
40	16.09	26.18	27.74	**29.48**	0.204	0.622	0.737	**0.821**
50	14.16	25.05	26.78	**28.23**	0.153	0.558	0.704	**0.790**
60	12.57	24.14	26.02	**27.47**	0.119	0.501	0.675	**0.770**
70	11.23	23.38	25.40	**26.83**	0.095	0.452	0.651	**0.752**
80	10.07	22.71	24.86	**26.30**	0.077	0.408	0.629	**0.737**
90	9.05	22.12	24.39	**25.84**	0.064	0.370	0.610	**0.723**
100	8.14	21.58	23.97	**25.43**	0.053	0.337	0.592	**0.710**
Average	15.02	25.54	27.31	**28.76**	0.216	0.558	0.709	**0.794**

Table 5.9: Performance comparison of Couple image

Noise Level	PSNR Comparison				SSIM Comparison			
	Noisy	NLM	BM3D	**Proposed**	Noisy	NLM	BM3D	**Proposed**
10	28.14	32.70	34.04	**35.32**	0.713	0.880	0.908	**0.934**
20	22.11	29.22	30.76	**32.15**	0.451	0.779	0.846	**0.889**
30	18.59	27.01	28.87	**30.31**	0.306	0.687	0.793	**0.854**
40	16.09	25.56	27.48	**29.20**	0.219	0.609	0.746	**0.826**
50	14.16	24.53	26.46	**27.89**	0.163	0.542	0.706	**0.789**
60	12.57	23.71	25.66	**27.09**	0.125	0.486	0.671	**0.765**
70	11.23	23.03	25.00	**26.42**	0.099	0.438	0.640	**0.742**
80	10.07	22.43	24.42	**25.85**	0.080	0.396	0.612	**0.722**
90	9.05	21.89	23.94	**25.36**	0.065	0.360	0.588	**0.704**
100	8.14	21.39	23.51	**24.96**	0.054	0.328	0.567	**0.689**
Average	15.02	25.15	27.01	**28.45**	0.228	0.550	0.708	**0.791**

Table 5.10: Performance comparison of Fingerprint image

Noise Level	PSNR Comparison				SSIM Comparison			
	Noisy	NLM	BM3D	**Proposed**	Noisy	NLM	BM3D	**Proposed**
10	28.14	30.98	32.46	**33.62**	0.921	0.954	0.968	**0.975**
20	22.11	27.60	28.81	**30.05**	0.764	0.895	0.927	**0.946**
30	18.59	25.27	26.83	**28.07**	0.612	0.829	0.890	**0.918**
40	16.09	23.44	25.30	**26.88**	0.487	0.760	0.853	**0.896**
50	14.16	21.97	24.53	**25.74**	0.390	0.691	0.826	**0.870**
60	12.57	20.82	23.75	**24.97**	0.315	0.625	0.798	**0.850**
70	11.23	19.93	23.12	**24.35**	0.258	0.567	0.774	**0.832**
80	10.07	19.23	22.56	**23.80**	0.214	0.515	0.750	**0.814**
90	9.05	18.66	22.06	**23.33**	0.179	0.471	0.727	**0.797**
100	8.14	18.20	21.61	**22.90**	0.151	0.433	0.704	**0.781**
Average	15.02	22.61	25.10	**26.37**	0.429	0.674	0.822	**0.868**

Table 5.11: Performance comparison of Hill image

Noise Level	PSNR Comparison				SSIM Comparison			
	Noisy	NLM	BM3D	**Proposed**	Noisy	NLM	BM3D	**Proposed**
10	28.14	32.60	33.62	**34.98**	0.691	0.853	0.883	**0.916**
20	22.11	29.59	30.72	**32.06**	0.408	0.750	0.804	**0.858**
30	18.59	27.78	29.16	**30.54**	0.259	0.669	0.751	**0.820**
40	16.09	26.54	27.99	**29.58**	0.177	0.601	0.708	**0.796**
50	14.16	25.59	27.19	**28.58**	0.127	0.541	0.676	**0.765**
60	12.57	24.80	26.52	**27.93**	0.095	0.489	0.648	**0.746**
70	11.23	24.10	25.93	**27.37**	0.073	0.442	0.624	**0.729**
80	10.07	23.46	25.43	**26.90**	0.058	0.401	0.603	**0.715**
90	9.05	22.87	24.98	**26.49**	0.047	0.365	0.584	**0.703**
100	8.14	22.30	24.58	**26.10**	0.039	0.333	0.567	**0.690**
Average	15.02	25.96	27.61	**29.05**	0.197	0.544	0.685	**0.774**

Table 5.12: Performance comparison of House image

Noise Level	PSNR Comparison				SSIM Comparison			
	Noisy	NLM	BM3D	**Proposed**	Noisy	NLM	BM3D	**Proposed**
10	28.10	35.37	36.71	**38.10**	0.595	0.895	0.923	**0.946**
20	22.08	32.27	33.77	**35.20**	0.338	0.830	0.875	**0.911**
30	18.56	29.78	32.09	**33.59**	0.224	0.755	0.850	**0.896**
40	16.06	27.87	30.65	**32.49**	0.161	0.677	0.828	**0.886**
50	14.12	26.40	29.69	**31.23**	0.121	0.602	0.815	**0.869**
60	12.54	25.24	28.74	**30.28**	0.095	0.534	0.797	**0.856**
70	11.20	24.28	27.91	**29.47**	0.076	0.475	0.778	**0.842**
80	10.04	23.47	27.16	**28.77**	0.062	0.424	0.758	**0.829**
90	9.02	22.76	26.48	**28.12**	0.051	0.380	0.741	**0.817**
100	8.10	22.12	25.87	**27.50**	0.043	0.342	0.723	**0.801**
Average	14.98	26.96	29.91	**31.48**	0.177	0.592	0.809	**0.865**

Table 5.13: Performance comparison of Man image

Noise Level	PSNR Comparison				SSIM Comparison			
	Noisy	NLM	BM3D	**Proposed**	Noisy	NLM	BM3D	**Proposed**
10	28.14	32.88	33.98	**35.33**	0.679	0.882	0.907	**0.933**
20	22.11	29.70	30.59	**32.03**	0.409	0.784	0.833	**0.880**
30	18.59	27.82	28.86	**30.30**	0.269	0.703	0.779	**0.843**
40	16.09	26.50	27.65	**29.30**	0.189	0.631	0.737	**0.818**
50	14.16	25.47	26.81	**28.20**	0.140	0.567	0.705	**0.786**
60	12.57	24.61	26.14	**27.53**	0.107	0.510	0.678	**0.766**
70	11.23	23.86	25.56	**26.96**	0.084	0.459	0.654	**0.748**
80	10.07	23.19	25.06	**26.49**	0.068	0.415	0.633	**0.734**
90	9.05	22.58	24.63	**26.06**	0.055	0.377	0.615	**0.719**
100	8.14	22.02	24.22	**25.68**	0.046	0.343	0.598	**0.707**
Average	15.02	25.86	27.35	**28.79**	0.205	0.567	0.714	**0.794**

Table 5.14: Performance comparison of Peppers image

Noise Level	PSNR Comparison				SSIM Comparison			
	Noisy	NLM	BM3D	**Proposed**	Noisy	NLM	BM3D	**Proposed**
10	28.10	33.49	34.68	**35.97**	0.690	0.918	0.930	**0.949**
20	22.08	30.38	31.29	**32.71**	0.443	0.854	0.889	**0.920**
30	18.56	28.23	29.28	**30.75**	0.314	0.787	0.853	**0.896**
40	16.06	26.58	27.70	**29.59**	0.235	0.721	0.817	**0.878**
50	14.12	25.22	26.68	**28.09**	0.183	0.656	0.794	**0.849**
60	12.54	24.10	25.81	**27.23**	0.147	0.597	0.769	**0.829**
70	11.20	23.14	25.07	**26.48**	0.120	0.542	0.745	**0.810**
80	10.04	22.33	24.45	**25.84**	0.099	0.493	0.724	**0.792**
90	9.02	21.63	23.87	**25.28**	0.083	0.451	0.702	**0.776**
100	8.10	21.03	23.39	**24.77**	0.070	0.413	0.683	**0.759**
Average	14.98	25.61	27.22	**28.67**	0.238	0.643	0.791	**0.846**

5.3.3 Subjective Comparison

For subjective comparison, we visually compared our denoised images with the original BM3D denoised image. We zoomed into different significant locations from an image to show that our proposed method achieved better output than BM3D.

Figure 5.5 shows the visual comparison of Lena image between the original BM3D and our proposed method. We have added AWGN with $\sigma = 100$ to original Lena image. We cropped the face region from both of BM3D denoised image and proposed method denoised image. It is noticeable that our proposed method produces more clear face image than BM3D. Faces are very significant region of any image. Also it is a smooth region. So our method works better in denoising smooth regions also.

Figure 5.6 shows the visual comparison of Man image between between BM3D and proposed method. We have added AWGN with $\sigma = 100$ to original Man image. From the denoised images, we cropped a texture region from both of BM3D denoised image and proposed method denoised image. It is noticeable that our proposed method outputs clearer textured regions than BM3D. Thus our proposed method is better in preserving edges.

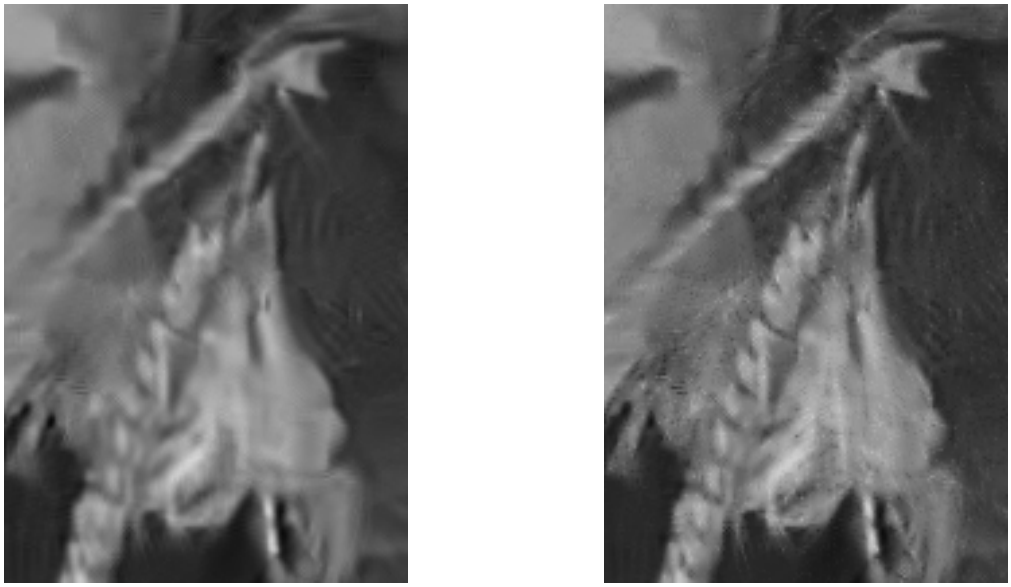
Figure 5.7 to 5.16 shows the visual comparison of Lena image for various noise levels. We have used 10 noise levels from 10 to 100, with step count of 10 and compared the output of our proposed method with BM3D. For low noise, the change is not so noticeable but for higher noise levels ($\sigma > 60$), it can be seen that performance of our proposed method is significantly better than BM3D.

(a) (b) (c) (d) (e) (f)

Figure 5.5: Subjective Comparison of Lena Image (a) Original Lena Image. (b) AWGN added with $\sigma = 100$ (c) Denoised Image using BM3D (d) Denoised Image using Proposed Method (e) Face Cropped from BM3D Output (f) Face cropped from Proposed Method Output

(a) (b)

(c) (d)



(e) (f)

Figure 5.6: Subjective Comparison of Man Image (a) Original Man Image. (b) AWGN added with $\sigma = 100$ (c) Denoised Image using BM3D (d) Denoised Image using Proposed Method (e) Cropped texture region fro BM3D output (f) Cropped texture region from proposed method output

(a) (b)

(c) (d)

Figure 5.7: Subjective Comparison of Lena Image (a) Original Lena Image. (b) AWGN added with $\sigma = 10$ (PSNR=12.57) (c) Denoised Image using BM3D (PSNR=35.93) (d) Denoised Image using Proposed Method (PSNR=37.22)

(a) (b)

(c) (d)

Figure 5.8: Subjective Comparison of Lena Image (a) Original Lena Image. (b) AWGN added with $\sigma = 20$ (c) Denoised Image using BM3D (PSNR=33.05) (d) Denoised Image using Proposed Method (PSNR=34.46)

(a) (b)

(c) (d)

Figure 5.9: Subjective Comparison of Lena Image (a) Original Lena Image. (b) AWGN added with $\sigma = 30$ (c) Denoised Image using BM3D (PSNR=31.26) (d) Denoised Image using Proposed Method (PSNR=32.78)

(a) (b)

(c) (d)

Figure 5.10: Subjective Comparison of Lena Image (a) Original Lena Image. (b) AWGN added with $\sigma = 40$ (c) Denoised Image using BM3D (PSNR=29.86) (d) Denoised Image using Proposed Method (PSNR=31.64)

(a) (b)

(c) (d)

Figure 5.11: Subjective Comparison of Lena Image (a) Original Lena Image. (b) AWGN added with $\sigma = 50$ (c) Denoised Image using BM3D (PSNR=29.05) (d) Denoised Image using Proposed Method (PSNR=30.54)

(a) (b)

(c) (d)

Figure 5.12: Subjective Comparison of Lena Image (a) Original Lena Image. (b) AWGN added with $\sigma = 60$ (c) Denoised Image using BM3D (PSNR=28.27) (d) Denoised Image using Proposed Method (PSNR=29.76)

(a) (b)

(c) (d)

Figure 5.13: Subjective Comparison of Lena Image (a) Original Lena Image. (b) AWGN added with $\sigma = 70$ (c) Denoised Image using BM3D (PSNR=27.57) (d) Denoised Image using Proposed Method (PSNR=29.10)

(a) (b)

(c) (d)

Figure 5.14: Subjective Comparison of Lena Image (a) Original Lena Image. (b) AWGN added with $\sigma = 80$ (c) Denoised Image using BM3D (PSNR=26.97) (d) Denoised Image using Proposed Method (PSNR=28.53)

(a) (b)

(c) (d)

Figure 5.15: Subjective Comparison of Lena Image (a) Original Lena Image. (b) AWGN added with $\sigma = 90$ (c) Denoised Image using BM3D (PSNR=26.45) (d) Denoised Image using Proposed Method (PSNR=28.01)

(a) (b)

(c) (d)

Figure 5.16: Subjective Comparison of Lena Image (a) Original Lena Image. (b) AWGN added with $\sigma = 100$ (c) Denoised Image using BM3D (PSNR=25.95) (d) Denoised Image using Proposed Method (PSNR=27.54)

5.3.4 Intensity Profile

Intensity profile is a measure for inspecting how sharp the edges are after denoising. In an intensity profile, we choose one scan line on the true image, noisy image and denoised image and plot them together to see how close the denoised profile is to the original profile. Also, it shows us how sharp the edges are after achieving denoising.

Let us consider the Lena image given in Figure 5.17a. Here, we consider the 150^{th} row (indicated by red line) for our intensity profile calculation. A plot of this scan line is shown in Figure 5.18a. We added AWGN with $\sigma = 50$ to the image and plot that scan line in Figure 5.18b. We can see a lot of noise added to each of the pixel. Now, to check how close our proposed method (as well as BM3D) is to the true noise free image, we perform the same task. That is, we consider the 150^{th} row from our denoised image and BM3D's denoised image and plot them. This is shown in Figure 5.18c and 5.18d respectively. In Figure 5.18e we combined the plot of true image, BM3D denoised image and our method's denoised image togehter. Same experiment is also done for House image. We consider the 100^{th} row of house image (indicated by red line) shown in Figure 5.17b. The intensity profile is shown in Figure 5.19.

It is clear from these plots that our proposed methods signal is closer to the true signal versus BM3D counterpart. Having a careful look at all the transitions of Figure 5.18e and 5.19e , we notice that the red line is closer to the blue line as compared to the green line. It shows that the proposed method is capable of achieving sharpenings than that of BM3D.

(a)

(b)

Figure 5.17: Image for Intensity Profile Calculation. Red Line shows the Scan Line taken as input to Intensity Profile (a) Lena Image (b) House Image

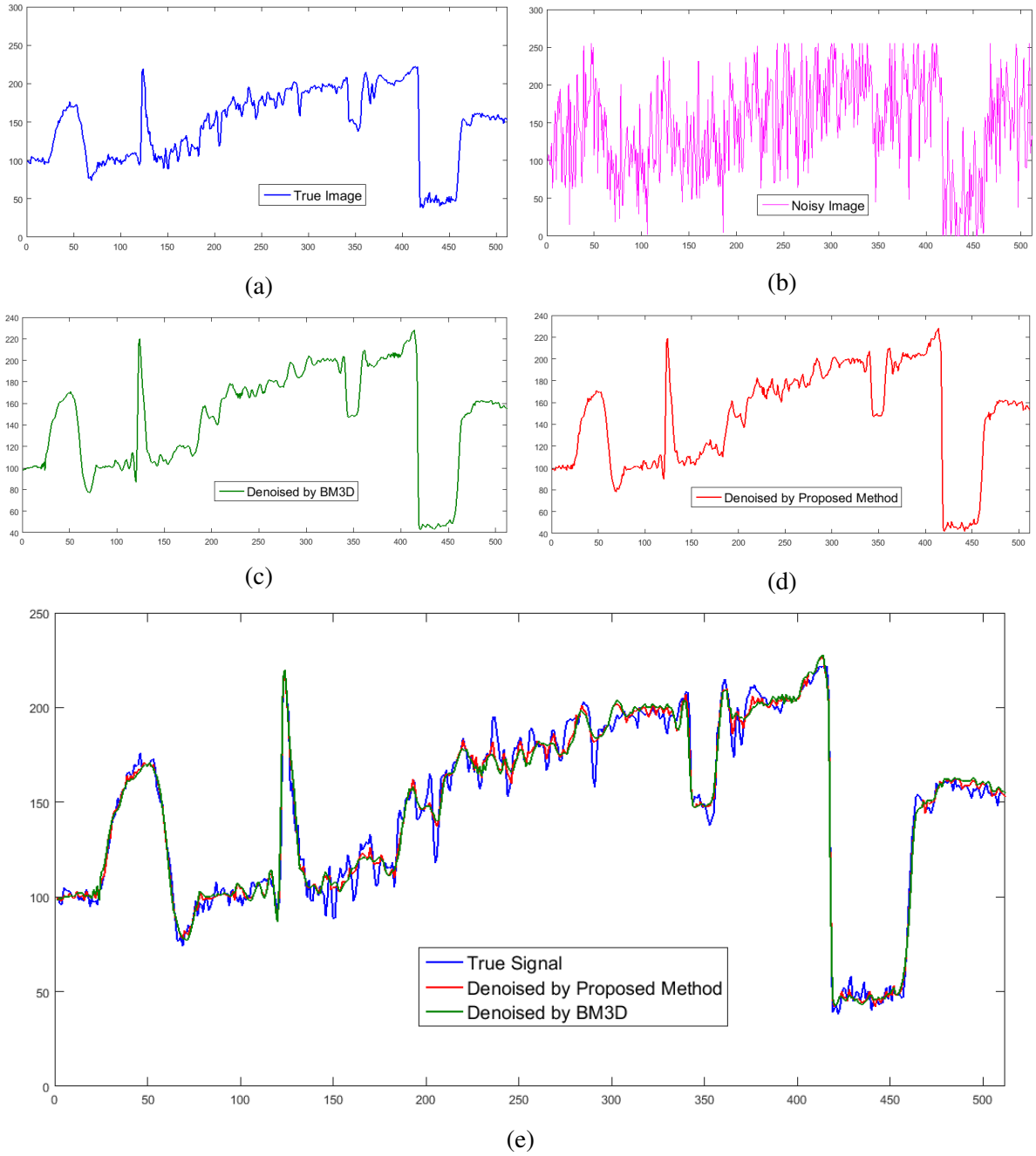


Figure 5.18: Intensity Profile for Lena Image at scan Line 100 (σ = 50) (a) Original Image (b) Noisy Image (c) Denoised by BM3D (Pearson correlation = 0.9836) (d) Denoised by Proposed Method (Pearson correlation = 0.9934) (e) Combining (a), (c) and (d)

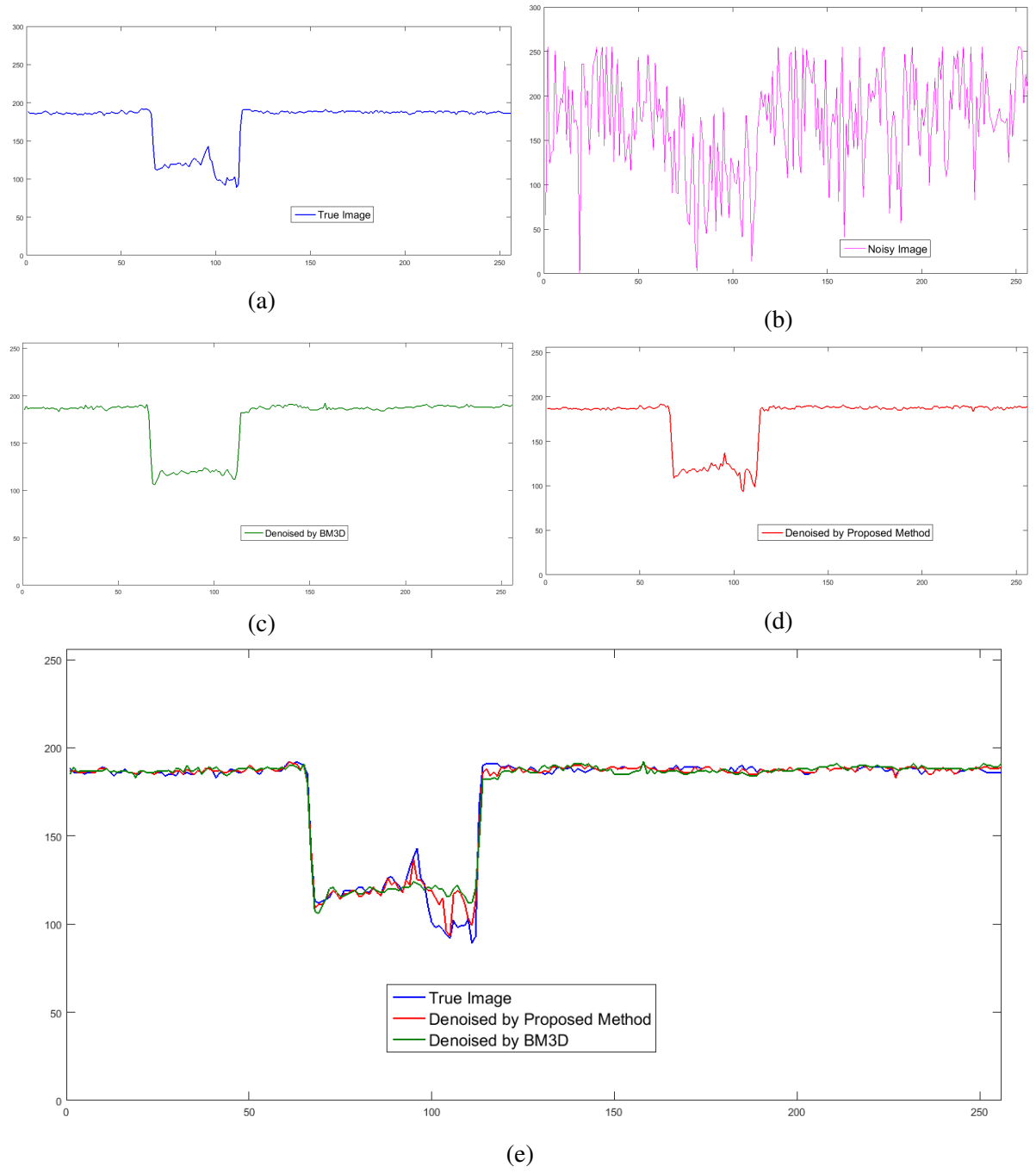


Figure 5.19: Intensity Profile for House Image at scan Line 100 (σ = 50) (a) Original Image (b) Noisy Image (c) Denoised by BM3D (Pearson correlation = 0.9766) (d) Denoised by Proposed Method (Pearson correlation = 0.9908) (e) Combining (a), (c) and (d)

Chapter 6

Conclusion and Future Works

6.1 Conclusion

In this thesis, we reviewed some noise models for digital images and reviewed some of the existing methods of image denoising. We presented the current state-of-the-art image denoising algorithm, BM3D and presented its persisting limitations. Then we focused on resolving those limitations and proposed a method to achieve better denoising performance than BM3D.

We proposed a learning-based adaptive hard thresholding scheme to overcome the shortcomings of using fixed hard thresholding in BM3D. We used the random forest classifier to train image blocks with their corresponding best threshold value. For any test image, we predicted each image block's best threshold value and use that threshold value in the collaborative filtering step of BM3D. We have also incorporated additive white Gaussian noise estimation method in our proposed method. The original BM3D relies on a user provided noise level while our method automatically calculates the noise level from a noisy image. Experimental result shows that we have achieved a significant improvement over the original algorithm both by subjective and objective measurement metrics. Our algorithm provides better denoising performance in terms of the PSNR and the SSIM. We have also showed visual quality comparisons for various test images and found that our method achieved significant improvement over

the original BM3D.

However our algorithm requires training classifiers, hence the time complexity is higher than the original BM3D. Once the classifiers are trained, it only requires a small amount of time to predict the label of each image block and hence the time complexity is similar to BM3D.

6.2 Future Work

For adaptive hard thresholding, we have used a number of classification algorithm in this thesis. In future we will experiment with other machine learning techniques. We are also planning to use deep learning in our algorithm to classify various image blocks.

We will extend our idea to color image denoising and video image denoising. In the future, we also have plan to continue our work to adaptively adjust other fixed parameters used in BM3D.

Bibliography

[1] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3d transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080 – 2095, 2007.

[2] R. Gonzalez and R. Woods, *Digital Image Processing*. Prentice-Hall Inc, 2008.

[3] E. Lukacs, "A characterization of the normal distribution," *The Annals of Mathematical Statistics*, vol. 13, no. 1, pp. 91–93, 1942.

[4] A. Bovik, *Handbook of Digital Image Processing*. Academic Press, 2005.

[5] J. Lee, "Digital image enhancement and noise filtering by using local statistics," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 2, 1980.

[6] D. Kuan, A. Sawchuk, T. Strand, and P. Chavel, "Adaptive restoration of images with speckle," *IEEE Transaction on Acoust., Speech, Signal Processing*, vol. 35, pp. 373–383, 1987.

[7] V. Frost, J. Stiles, K. Shanmugan, and J. Holtzman, "A model for radar images and its application to adaptive digital filtering of multiplicative noise," *IEEE Transactions on Pattern Analysis Machine Intelligence*, vol. 4, pp. 157–165, 1982.

[8] T. Yu and S. Acton, "Speckle reducing anisotropic diffusion," *IEEE Transactions on Image Processing*, vol. 11, no. 11, pp. 1260–1270, 2002.

[9] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629 – 639, 1990.

[10] A. Buades, B. Coll, and J. Morel, "A non-local algorithm for image denoising," in *Proc. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[11] A. Buades, B. Coll, and J. Morel, "A review of image denoising algorithms, with a new one," *SIAM Journal on Multiscale Modeling and Simulation: A SIAM Interdisciplinary Journal*, vol. 4, pp. 490–530, 2005.

[12] A. Rehman and Z. Wang, "Ssim-based non-local means image denoising," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2011.

[13] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[14] T. Thaipanich, B. Oh, P. Wu, and C. Kuo., "Adaptive nonlocal means algorithm for image denoising," in *Proc. IEEE International Conference on Consumer Electronics (ICCE)*, 2010.

[15] K. Chaudhury and A. Singer, "Non-local euclidean medians," *IEEE Signal Processing Letters*, vol. 19, no. 11, pp. 745–748, 2012.

[16] G. Maruf and M. El-Sakka, "Improved non-local means algorithm based on dimensionality reduction," in *Proc. International Conference on Image Analysis and Recognition (ICIAR)*, 2015.

[17] M. Alkinani and M. El-Sakka, "Non-local means for stereo image denoising using structural similarity," in *Proc. International Conference on Image Analysis and Recognition (ICIAR)*, 2015.

[18] S. Lim, *Two-Dimensional Signal and Image Processing*. New Jersey, USA: Prentice Hall, 1990.

[19] F. Jin, P. Fieguth, L. Winger, and E. Jernigan, "Adaptive wiener filtering of noisy images and image sequences," in *Proc. IEEE International Conference of Image Processing (ICIP)*, 2003.

[20] N. Wiener, *The Interpolation, Extrapolation and Smoothing of Stationary Time Series*. MIT press, New York, 1949.

[21] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising with block-matching and 3d filtering," in *Proc. SPIE Electronic Imaging*, 2006.

[22] M. Lebrun, "An analysis and implementation of the bm3d image denoising method," *Image Processing On Line*, vol. 2, p. 175 213, 2012.

[23] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Color image denoising via sparse 3d collaborative filtering with grouping constraint in luminance-chrominance space," in *Proc. IEEE International Conference on Image Processing (ICIP)*, 2007.

[24] K. Dabov, A. Foi, and K. Egiazarian, "Video denoising by sparse 3d transform-domain collaborative filtering," in *Proc. European Signal Processing Conference (EUSIPCO)*, 2007.

[25] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Bm3d image denoising with shape-adaptive principal component analysis," in *Proc. Workshop on Signal Processing with Adaptive Sparse Structured Representations (SPARS)*, 2009.

[26] M. Maggioni, V. Katkovnik, K. Egiazarian, and A. Foi, "A nonlocal transform-domain filter for volumetric data denoising and reconstruction," *IEEE Transactions on Image Processing*, vol. 22, no. 1, pp. 119–133, 2013.

[27] L. Dai, Y. Zhang, and Y. Li, "BM3D image denoising algorithm with adaptive distance hard-threshold," *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 6, no. 6, pp. 41–50, 2013.

[28] H. Burger, C. Schuler, and S. Harmeling, "Image denoising: Can plain neural networks compete with bm3d ?," in *Proc. European Signal Processing Conference Computer Vision and Pattern Recognition (CVPR)*, 2012.

[29] A. Mittal, A. Moorthy, and A. Bovik, "Automatic parameter prediction for image denoising algorithms using perceptual quality features," in *Proc. SPIE Human Vision and Electronic Imaging*, 2012.

[30] M. Hasan, *Adaptive Edge-guided Block-matching and 3D filtering (BM3D) Image Denoising Algorithm*. The University of Western Ontario, 2014.

[31] M. Hasan and M. El-Sakka, "Structural similarity optimized wiener filter: A way to fight image noise," in *Proc. International Conference on Image Analysis and Recognition (ICIAR)*, 2015.

[32] M. Hasan, *BM3D Image Denoising using SSIM Optimized Wiener Filter*. The University of Western Ontario, 2014.

[33] T. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc., 1997.

[34] V. Vapnik, *Statistical Learning Theory*. New York, NY, USA: Wiley, 1998.

[35] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[36] A. Hore and D. Ziou, "Image quality metrics: PSNR vs. SSIM, booktitle=Proc. International Conference on Pattern Recognition (ICPR), year = 2010,"

[37] Z. Wang, E. Simoncelli, and A. Bovik, "Multiscale structural similarity for image quality assessment," in *Proc. Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, 2004.

[38] Z. Wang and A. Bovik, "A universal image quality index," *IEEE Signal Processing Letters*, vol. 9, no. 3, pp. 81–84, 2002.

Curriculum Vitae

Name:	Farhan Bashar
Post-Secondary Education and Degrees:	Bachelor of Science Department of Computer Science and Engineering Islamic University of Technology Gazipur, Bangladesh 2009 - 2012
Honours and Awards:	Western Graduate Research Scholarship (WGRS) The University of Western Ontario 2014 - 20015
Related Work Experience:	Graduate Research & Teaching Assistant The University of Western Ontario 2014 - 2015 Software Engineer Samsung Research and Development Institute Bangladesh 2012 - 2014

Publications:

- F. Bashar and M. R. El-Sakka, "BM3D Image Denoising with Learning-Based Adaptive Hard Thresholding", *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2016. [Submitted]
- F. Bashar, "BM3D Image Denoising with Learning-Based Adaptive Hard Thresholding", *Three Minute Thesis presentation at Fallona Family Interdisciplinary Showcase*, 2015. [Submitted]