Impulse Noise Removal Using Unbiased Weighted Mean Filter

Cengiz Kandemir

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
July 2015
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Serhan Çiftçioğlu
Acting Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Asst. Prof. Dr. Önsen Toygar
Supervisor

Examining Committee

1. Prof. Dr. Hasan Demirel

2. Assoc. Prof. Dr. Alexander Chefranov

3. Asst. Prof. Dr. Önsen Toygar

ABSTRACT

Digital imaging technology has provided countless opportunities for human visual applications and many scientific disciplines such as astronomy and microbiology. Digital images are subject to various noise due to environmental factors or faults in hardware. One type of noise — impulse noise — manifests itself with the highest or the lowest intensity value in the dynamic range during digitization process. Impulse noise involves high frequency components which are undesirable. Therefore, it is vital to restore contaminated digital images before utilizing them in various applications.

In this thesis, we have investigated Nonlinear Fixed-Valued Impulse (salt-and-pepper) Noise removal methods. Restoration of a contaminated image is composed of two stages. These are noise detection and restoration. The performance of various state-of-the-art impulse noise removal methods are empirically compared for these two stages. For detection, misclassification and false-alarm rates are used for objective measurement. Restoration capabilities are compared in terms of Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM) and Mean Absolute Error (MAE).

We have also identified a common problem among impulse noise removal methods, namely, *spatial bias*. Asymmetric distribution of corruption prevents an equal contribution of uncorrupted pixels in the filtering window from a spatial perspective, effectively yielding a biased estimation of the original intensity value. In order to eliminate spatial bias, we have proposed Unbiased Weighted Mean Filter (UWMF). UWMF eliminates spatial bias by recalibrating pixel weights based on the positional distribution of corrupted pixels in the filtering window. Recalibrated weights reflect the spatial properties of corruption and compensate the missing contribution.

We have demonstrated that elimination of spatial bias improves restoration quality in terms of objective measurements (PSNR, SSIM and MAE). In addition, unbiased restoration results with least amount of disturbance in the edges and smooth regions.

Keywords: Impulse Noise Removal, Nonlinear Filters, Weighted Mean Filters, Median Filters

ÖZ

Sayısal imge teknolojisi, görsel uygulamalarda olduğu gibi, astronomi ve mikrobiyoloji başta olmak üzere birçok bilim dalı için sayısız olanaklar sağlamıştır. Sayısal imgeler, çevresel nedenlerden veya donanımsal sorunlardan ötürü çeşitli gürültülere mağruz kalabilirler. Bu gürültülerden bir tanesi olan darbe gürültüsü, kendisini sayısallaştırma sürecinde, uç noktadaki koyuluk değerlerini alması ile gösterir. Darbe gürültüsü, yüksek frekans bileşenler içerir. Bu bileşenler, çeşitli nedenlerden dolayı istenmemektedir. Dolayısıyla, darbe gürültüsüne mağruz kalmış sayısal imgelerin yenilenmesi, bu imgelerin düzgün bir şekilde kullanılması için büyük önem arz eder.

Bu tezde Doğrusal-olmayan Sabit-değerli Darbe Gürültüsü yenileme yöntemleri incelenmiştir. Sayısal bir imgenin yenilenmesi iki aşamadan oluşur. Bunlar, gürültünün tespiti ve yenilenmesidir. Çeşitli darbe gürültüsü temizleme yöntemleri, bu iki aşama üzerinden değerlendirilmiştir. Gürültünün tespit başarısının değerlendirilmesinde, hatalı tespit ve yanlış uyarı oranları kullanılmıştır. Yenileme başarısının ölçümü için ise Doruk Sinyal-Gürültü Oranı (PSNR), Yapısal Benzerlik (SSIM) ve Ortalama Mutlak Hata (MAE) kullanılmıştır.

İnceleme esnasında, bütün yöntemlerde ortak olan, uzamsal meyil ismini verdiğimiz bir sorun tespit edilmiştir. Darbe gürültüsünden kaynaklı bozulmanın, süzgeçleme penceresi üzerindeki asimetrik dağılımı, bozulmaya uğramamış imge öğelerinin eşit şekilde katkı yapmasını engeller. Bu durum, temizlenmesi amaçlanan imge öğesinin tahmin edilen koyuluk değerinin, bozulmanın yoğun olmadığı bölgelerdeki koyuluk değerlerine meyilli olması sonucunu doğurur. Bu sorunu ortadan kaldırmak için Meyilsiz Ağırlıklı Ortalama Süzgeci (UWMF) yönetmi önerilmiştir. Önerilen yöntem, süzgeçleme penceresindeki bozulmaya uğramış imge öğelerinin uzamsal dağılımlarını kullanarak, ağırlıkları yeniden ayarlar ve uzamsal meyili ortadan kaldırır. Yeniden

ayarlanmış olan ağırlıklar, bozulmanın uzamsal özelliklerini yansıtır ve dolayısı ile yapılan tahmine katkısı az olan imge öğelerinin katkılarını eşitler.

Yapılan deneyler sonucunda, uzamsal meyilin ortadan kaldırılmasının yenileme başarısını arttırdığı saptanmış ve deney sonuçları, nesnel değerlendirme ölçütleri (PSNR, SSIM ve MAE) ile gösterilmiştir. Ek olarak, meyilsiz temizleme, kenar ve düz bölgelerde en az karışıklık ile sonuçlanmıştır.

Anahtar Kelimeler: Darbe Gürültüsü Yenileme, Doğrusal-olmayan Süzgeçler, Ağırlıklı Ortalama Süzgeçleri, Orta Değer Süzgeçleri

to my family

ACKNOWLEDGMENT

I would like to thank my family for their endless support. I am lucky to have them. I also would like express my deep gratitude to Cem Kalyoncu and my supervisor Dr. Önsen Toygar. Their guidance throughout my years in EMU has thought me a lot. Finally, I would like to thank Dr. Marifi Güler and Dr. Hasan Demirel for their assistance and advices.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

AADF	Adaptive Anisotropic Diffusion Filter
ADF	Anisotropic Diffusion Filter
AMF	Adaptive Median Filter
ASMF	Adaptive Switching Morphological Filter
AWMF	Adaptive Weighted Mean Filter
BDND	Boundary Discriminative Noise Detection
CFI	Continued Fraction Interpolation
CFIF	Continued Fractional Interpolation Filter
CMF	Cloud Model Filter
DBF	Decision-based Filter
FA	False Alarm
FVIN	Fixed-Valued Impulse Noise
IBDND	Improved Boundary Discriminative Noise Detection Filtering Algorithm
IBINR	Interpolation-based Impulse Noise Removal
MAE	Mean Absolute Error
MD	Misdetection
MDBUTMF	Modified Decision Based Unsymmetric Trimmed Median Filter
NAFSMF	Noise Adaptive Fuzzy Switching Median Filter
NASMF	Noise Adaptive Soft-Switching Median Filter
NMF	Neighborhood Mean Filter
NND	Naïve Noise Detection
PSNR	Peak Signal-to-Noise Ratio
RAMF	Recursive Adaptive Median Filter

ROLD	Rank Ordered Logarithmic Difference
RVIN	Random-Valued Impulse Noise
SMF	Standard Median Filter
SSIM	Structural Similarity
UWMF	Unbiased Weighted Mean Filter

LIST OF PUBLICATIONS

Kandemir, C., Kalyoncu, C., & Toygar, O. **A Weighted Mean Filter with Spatial-bias Elimination for Impulse Noise Removal**. *Digital Signal Processing* (under review).

Chapter 1

INTRODUCTION

Digital images have found endless use since their inception. The interest over utilization of digital images has sparkled especially after 1960's, thanks to the advancements in high level programming languages and microprocessors. Today, many fields, ranging from astronomy to microbiology, require digital imaging technology.

During acquisition or transmission, digital images are subjected to various noise due to environmental factors or faulty hardware. One type of noise, called impulse noise, manifests itself with relatively high (white) or low (black) intensity value on a given pixel. Images contaminated by impulse noise involve high frequency components and they are undesirable for the following reasons.

- Human visual system is sensitive to the presence of impulse noise
- High frequency components are problematic for various image processing methods (edge detection etc.)

Impulse noise is an impediment to pictorial interpretation and decreases the suitability of images for both human and computer vision applications. Therefore, it is vital to restore these images in order for further use.

The methods aiming to remove impulse noise (also called *filters*) are categorized under two types — linear and nonlinear — based on their characteristics as described in Section 3.2. In late 1970's and early 1980's, it has been understood that nonlinear filters were superior compared to linear filters in terms of impulse noise removal capabilities,

which sparkled a burst of interest for nonlinear filters. Furthermore, the superiority of nonlinear filters are not limited to impulse noise.

Nonlinear filters for impulse noise removal exploit local (segment of image) and/or global (entire image) statistics and spatial relationship in order to estimate the original intensity value of a corrupted pixel. Restoration procedures are applied while *convolving* a contaminated image, that is, visiting the image pixels one by one. In general, the local information is sought in a square neighborhood centered on a pixel called *filtering window* with odd side lengths.

This thesis investigates nonlinear filters for impulse noise removal. The primary aims of this thesis are to conduct an empirical analysis to assess the performance of various state-of-the-art methods for impulse noise removal and to improve restoration process via preserving edges and other fine details of contaminated images. For the latter, our efforts have focused on exploiting spatial relationship between pixels and distribution of noise.

There are more than one model for impulse noise as described in Section 3.1. This thesis is mainly concerned with a specific model called Fixed-Valued Impulse Noise. Preliminary empirical studies for Random-Valued Impulse Noise are presented under Section 5.6. However, a deeper analysis of the state-of-the-art Random-Valued Impulse Noise removal methods and investigating the potentiality of extending the findings of this thesis to the other models of impulse noise are beyond the scope. The rest of the thesis involves five chapters. Chapter 2 surveys the impulse noise removal literature. In Chapter 3, various preliminary information is provided. Although, some of the topics discussed under this chapter may not be required to understand subsequent chapters, they are provided for the sake of completeness. Chapter 4 proposes a new impulse noise filter aiming to remove what we call *spatial bias*. The details about experiments,

findings and their implications are presented under Chapter 5. Finally, the conclusive remarks are rendered under Chapter 6.

Chapter 2

LITERATURE SURVEY

This chapter provides a survey of impulse noise removal literature. Section 2.1 gives an overview of the history of nonlinear filters and research directions in general. The review of the state-of-the-art methods are presented under four sections differentiated by their restoration characteristics. These are Median Filters (Section 2.2), Mean Filters (Section 2.3) and Interpolation Based Filters (Section 2.4). The filters belonging none of these groups are presented under Section 2.5.

2.1 Overview

The utilization of nonlinear filters for impulse noise removal is triggered by the success of Standard Median Filter (SMF) [35] in late 1970s [10, 12, 13]. The subsequent studies have mainly focused on improving SMF [5, 18, 26, 37]. Although properties of impulse noise and restoration methods have been studied extensively, it is still an active research topic [2, 21, 29, 38] involving applications of methods and models in other research fields such as natural language processing [42], neural networks [22] and soft computing [34, 37].

Perhaps two of the most important concepts employed in impulse noise removal are adaptiveness and the concept of switching. Adaptive filters — filters capable of changing their properties based on (mostly local) characteristics of subject image are superior when various information, e.g. noise density, are unknown. Many adaptive filters are proposed over the years [4, 14, 15, 27, 40, 41]. Among them, Adaptive Median Filter (AMF) [23] is a well-known, benchmark method that is widely used due to its fine detail preservation capabilities. Switching filters [7, 16, 25, 32, 33, 38] try to detect corruption on a given pixel before applying any restoration procedure.

The restoration process involves two objectives: accurate noise detection and a close estimation of the original intensity value. The studies focusing on the former objective are hard to unify in terms of research direction. Among detection algorithms, one naïve yet accurate and commonly employed [3, 8, 17, 36, 43] method is classification based on the maximum and the minimum possible intensity value in the dynamic range of an image. For an 8-bit monochrome image, these values are 0 and 255. This method will be called Naïve Noise Detection (NND) hereafter. For the latter objective, utilization of a weight function is a common method [3, 5, 16–18, 37, 39, 41, 42]. In general, weight functions reflect the spatial relationship.

2.2 Median Filters

2.2.1 Standard Median Filter

Standard Median Filter (SMF) [11] replaces the pixel under consideration with the median value of its neighborhood. More rigorously, SMF is defined as follows

$$r_{x,y} = \underset{(i,j) \in c_{FW}}{median} \{c_{i,j}\} \tag{2.1}$$

where r is the restored image and c is the contaminated image. Coordinates in subscripts represent the pixel intensity value at those coordinates. c_{FW} represents the filtering window of the contaminated image.

SMF does not have a switching procedure to ignore corrupted pixels. It inherently assumes that the median value is not corrupted because extreme values (i.e., corruption) will be clustered at the ends of lower and upper half. However, when noise density is higher than 50%, this is not the case. SMF performs poorly for highly contaminated images.

2.2.2 Adaptive Median Filter

Adaptive Median Filter (AMF) [11] improves two drawbacks of SMF. These are selecting a corrupted pixel in the presence of high noise density and absence of a noise detection procedure. The former issue is addressed by incrementing the size of the filtering window based on the following criterion

$$((c_{med} - c_{min}) \leq 0 \quad OR \quad (c_{med} - c_{max}) \geq 0) \quad AND \quad wsize < max_{wsize}$$

where c is a contaminated image; c_{min}, c_{med} and c_{max} are the minimum, the median and the maximum intensity values in the filtering window, respectively. $wsize$ is the current filtering window size; max_{wsize} is the maximum filtering window size. Corrupted pixels are detected based on the following criterion

$$(c_{x,y} - c_{min}) > 0 \quad AND \quad (c_{x,y} - c_{max}) < 0$$

where $c_{x,y}$ is a pixel at coordinates $(x.y)$.

Since AMF is sensitive to the presence of highly dense filtering windows, it is less likely to select a corrupted pixel as median. AMF is superior in terms of restoration performance compared to SMF.

2.2.3 Recursive and Adaptive Median Filter

Meher and Singhawat proposed Recursive Adaptive Median Filter (RAMF) in [24]. RAMF is developed based on the merits of various methods, namely, Decision-based Filter (DBF) [30], Neighborhood Mean Filter (NMF) [31] and Modified Decision Based Unsymmetric Trimmed Median Filter (MDBUTMF) [8]. It also utilizes the detection algorithm of Noise Adaptive Fuzzy Switching Median Filter (NAFSMF) [33]. For restoration, while convolving the contaminated image, RAMF selects the median of the filtering window (3×3) if there is at least one uncorrupted pixel. Otherwise, it increases the window size to 5×5. In the larger filtering window, previous condition is checked again, however, on failure to find uncorrupted pixels, the median of restored pixels of four adjacent 3×3 filtering windows is selected. RAMF provides comparable restoration quality.

2.2.4 Improved Boundary Discriminative Noise Detection Filter

Improved Boundary Discriminative Noise Detection Filtering Algorithm (IBDND) is proposed in [16]. In their efforts to improve the filtering stage of Boundary Discriminative Noise Detection (BDND) [25], the authors focus on expansion condition of the filtering window and incorporation of spatial distance. IBDND makes use of estimated noise density from detection stage in the expansion condition for filtering window, effectively making it adaptive to noise density. The obvious problem with such approach is the propagation of errors from detection stage to filtering stage, due to the usage of estimated noise density, that affects not only the restoration quality but also computational efficiency. In order to incorporate spatial correlation, IBDND inversely relates the distance to the center and contribution factor of pixels. IBDND surpasses the restoration quality of BDND both in terms of Peak Signal-to-Noise Ratio (PSNR) and Tenengrad metrics for monochrome images; ΔE^*_{ab} for color images and operates on smaller window size compared to that of BDND.

2.2.5 Modified Decision Based Unsymmetric Trimmed Median Filter

In [8], authors proposed a variant of median filter, namely, Modified Decision Based Unsymmetric Trimmed Median Filter (MDBUTMF). MDBUTMF ignores corrupted pixels while ordering intensity values in the filtering window. The noise is detected with NND. After constructing a trimmed pixel set, the central pixel is replaced by the median value. If all pixels are corrupted, then mean of the filtering window is used. At higher noise densities, median value may not provide optimal estimation as distant pixels are incorporated without considering the spatial relationship. This problem results with poor restoration quality in edges as well as smooth regions. Overall, MDBUTMF is highly efficient in terms of computational efficiency and provides high restoration quality.

2.2.6 Noise Adaptive Fuzzy Switching Median Filter

Another switching-based adaptive median filter is Noise Adaptive Fuzzy Switching Median Filter (NAFSMF) [33]. NAFSMF is a hybrid of AMF and Fuzzy Switching Median Filter [32]. It first labels two intensity values (L_{salt} and L_{pepper}) for upper and

lower impulsive peaks of contaminated image signal using image histogram. A pixel is considered as corrupted if it is equal to one of these values. The filtering process starts with expanding the filtering window while searching for uncorrupted pixels. Upon finding enough uncorrupted pixels, median in current window is selected for restoration. However, expansion stops on a predetermined window size if no uncorrupted pixels are found. In such a case, a new 3×3 window is imposed and median of the first four pixels in the upper-left region is selected for restoration. NAFSMF provides high restoration quality in terms of PSNR. However, calculation of two peak values introduces an additional stage and effectively increases the computational complexity and suitability for real-time applications.

2.2.7 A Switching Median Filter with Boundary Discriminative Noise Detection

Ng and Ma [25] incorporated Noise Adaptive Soft-Switching Median Filter (NASMF) [7] with Boundary Discriminative Noise Detection (BDND). BDND forms three intensity clusters as lower intensity impulse noise, uncorrupted pixels and higher intensity impulse noise by calculating lower and higher intensity boundary values using the intensity differences between pixels in the filtering window. BDND introduces three modification to filtering stage of NASMF. The first modification reduces maximum window size in order to eliminate blurring effect at higher noise densities. The second modification imposes an additional condition to the filtering window expansion. The additional condition allows the filtering window to expand when there is no uncorrupted pixels. The third and final modification is ignoring corrupted pixels in ranking operation. BDND shows robust detection performance even at high noise densities with total misclassification rate less than 1%.

2.2.8 Noise Adaptive Soft-Switching Median Filter

Noise Adaptive Soft-Switching Median Filter (NASMF) [7] switches the filtering method employed based on the information gathered on the detection stage. In the filtering stage, the pixels are differentiated into four classes as uncorrupted pixels, isolated impulse noise, non-isolated impulse noise and edge pixels, using local and global pixel statistics. Isolated and non-isolated impulse noise are restored with SMF; edge

pixels are restored with Fuzzy Weighted Median Filter. NASMF performs well with lower noise densities. However, at higher noise densities, it fails to restore fine details due to incorrect classification of pixels.

2.3 Mean Filters

2.3.1 Adaptive Weighted Mean Filter

In [39], authors propose a method named Adaptive Weighted Mean Filter (AWMF). AWMF operates in a similar way to AMF. In order to enhance the performance of AMF, the authors focus on decreasing detection errors and estimating a value better than median. AWMF adaptively increases the filtering window size until two successive windows have equal minimum and maximum intensity values. During adaptive process, the central pixel is considered as corrupted if it is equal to current maximum or minimum intensity values. Pixels with intensity values between current maximum and minimum intensity values are assigned to 1; the rest is assigned to 0. Then, the central pixel is replaced with a weighted mean using binary weights. Compared to AMF, the expansion of window size is relatively limited which may improve computational efficiency drastically. In addition, AWMF surpasses AMF in terms of noise detection rate and restoration quality.

2.3.2 Cloud Model Filter

Cloud Model Filter (CMF), proposed by Zhou [42], employs Cloud Model [19] for detection and filtering stages which exploits both randomness and fuzziness involved in impulse noise. CMF considers the intensity values in a windows as a normal distribution with various parameters — expected value (Ex), entropy (En) and hyperentropy (He). In detection stage, an adaptive approach is used. The central pixel is considered as uncorrupted if it is between maximum and minimum value in the filtering window. These values are calculated using Ex and En. The filtering window expands if the central pixel is less than minimum value or greater than maximum value and number of uncorrupted pixels is less than threshold δ. In filtering stage, a weighted fuzzy mean filter is used. CMF successfully detects impulse noise with total misclassification rate less than 0.01% at higher noise densities while its detection rate diminishes slightly at

lower noise densities. CMF does not require the detection stage to be completed before performing filtering. This improves computational efficiency as restoration can be done while iterating corrupted image for detection.

2.4 Interpolation Based Filters

2.4.1 Interpolation-based Impulse Noise Removal

Interpolation-based Impulse Noise Removal (IBINR) is proposed in [17]. Corrupted pixels are detected using NND. IBINR assigns weights to the uncorrupted pixels in the filtering window based on their Euclidean Distance to the center, then it replaces the central pixel with a weighted mean. The detection method, while intuitive in parallel with the nature of impulse noise, is bound to failure in binary images. This problem is solved by counting the number of black and white pixels and assigning whichever has highest occurrence when uncorrupted pixels are absent in the filtering window. At higher noise densities, IBINR enlarges the filtering window size up to 13×13. As distance to the center increases, the spatial correlation decreases which may result with lower restoration quality. This is, however, solved by a term mitigating the contribution factor of distant pixels. In general, IBINR provides robust restoration performance while maintaining computational efficiency.

2.4.2 Continued Fractions Interpolation Filter

In [3], another interpolation-based method is proposed by Bai et al. Continued Fractional Interpolation Filter (CFIF) incorporates Thiele's Continued Fraction Interpolation (CFI). NND is employed for noise detection. Then, for all four directions (0°, 45°, 90°and 135°), CFI is computed. For each CFI value in four directions, weights are calculated in an adaptive manner with an exponential weight function using four neighbors (two on the left, two on the right). If the calculated weighted mean is not in the range of minimum and maximum intensity values of pixels in any direction, then the number of neighbors used in weight function is adaptively divided by two and the weights are calculated again. The detection method of CFIF is bound to failure for binary images and CFIF has no handling mechanism as once the pixel is classified, next pixel is taken into

consideration without any further analysis. In terms of noise removal, CFIF provides comparable results in terms of PSNR.

2.5 Other Filters

2.5.1 Adaptive Switching Morphological Filter

Feng et al. [9] proposed Adaptive Switching Morphological Filter (ASMF). ASMF utilizes a morphological two-stage noise detector and conditional rank-order filter. In the first stage of detection, erosion and dilation operations are employed iteratively to find internal and external morphological gradients. These gradients are then used to identify corrupted pixels. Due to high false alarm rate at the end of the first stage, an additional detection stage is imposed. In this stage, erosion and dilation operations are applied to those pixels that are marked as corrupt in the first stage. The absolute difference D between mean of these two operations and corresponding pixel value in the contaminated image is compared with a threshold. Pixels having greater D value than the threshold are classified as corrupted. In filtering stage, the size of structuring element is adaptively determined based on uncorrupted pixels in the local and mean of r-th min (for erosion) and max (for dilation) is used as an estimation of the original intensity value. ASMF is accurate (total misclassification $< \%2$) in terms of impulse noise detection even at high noise densities and high restoration performance.

2.5.2 Adaptive Anisotropic Diffusion Filter

Veerakumar et al. [36] proposed edge preserving Adaptive Anisotropic Diffusion Filter (AADF). AADF employs Rank Ordered Logarithmic Difference (ROLD) [6] for Random-Valued Impulse Noise (RVIN) and NND for Fixed-Valued Impulse Noise (FVIN). In restoration process, the contaminated image is multiplied with noise map (a binary image containing 0s and 1s) and if the value of the pixel in consideration is greater than zero, then its value is retained. Otherwise, two types of filters are used. In a 3×3 filtering window, if all pixels are corrupted then traditional mean filter in a 5×5 filtering window is selected for restoration. If there are some uncorrupted pixels, then a modified version of Anisotropic Diffusion Filter (ADF) [28] is applied. Although AADF is not suitable for binary images, it provides high restoration performance when

applied to monochrome images. In addition, AADF involves application of many convolution kernels which are costly in terms of computational efficiency and reduces the suitability for real-time applications.

2.5.3 Impulse Noise Removal Based on Noise Space Characteristics

A method exploiting noise space characteristics is proposed in [43]. In this method, three noise patterns called *T-Single-noise-pattern*, *T-Double-noise-pattern*, *T-Triple-noise-pattern* and their respective noise removal operators are defined. Noise detection is done by incorporating NND in these operators. In order to understand which pattern is present at any given time, the difference between intensity values and square root of weighted standard deviation in the filtering window is analyzed. All patterns restore using median value, however, the window sizes are different for each pattern. The filtering window sizes for *T-Single-noise-pattern*, *T-Double-noise-pattern* and *T-Triple-noise-pattern* is 3×3, 3×4 and 4×4, respectively. In addition, the corrupted pixels are ignored in the ranking process. The restoration performance of the proposed method is comparable with the state-of-the-art methods.

2.5.4 Neuro-Fuzzy Based Impulse Noise Filter

A neuro-fuzzy based impulse noise removal method is proposed in [20]. Two first order *Sugeno*-type fuzzy inference systems containing four inputs and one output are employed. The difference between four pixels in horizontal middle line (for the first system) and vertical middle line (for the second system) and median in the filtering window, together with median itself is fed to the systems as inputs. Each input has three generalized bell-type membership functions and for all four inputs there are total of 81 rules (3^4). The output of the filter (Y) is weighted average of the outputs of the individual rules and weights of each rule is multiplication of membership values. If Y is outside of the dynamic range for intensity values (0 and 255 for 8-bit monochrome images), then it is truncated. The restoration is done by replacing each pixel with the average of the outputs of two fuzzy filters. These filters require optimization for various internal parameters. For this purpose, a hybrid training phase involving Gradient Descent and Least Squares Algorithms is conducted. The training phase is required to be performed

only once which decreases off-line time of the method, i.e., the time spend for outside of filtering, significantly.

Chapter 3

PRELIMINARIES

This chapter concerns various concepts regarding impulse noise and nonlinear filters. Two well-known impulse noise models are presented under Section 3.1. In Section 3.2, the concept of linearity is explained.

3.1 Impulse Noise Models

Impulse noise occurs due to electromagnetic interference which manifests itself with relatively high or low intensity values compared to uncorrupted pixels. Transmission of image signal in noise channel and faulty camera sensors are common causations for such contamination. It is essential to understand the nature of impulse noise in order to remove it effectively.

Although several impulse noise models are proposed in the literature, only two of them are widely adopted. These are Fixed-Valued Impulse Noise (FVIN), which this study is interested in, and Random-Valued Impulse Noise (RVIN). The former model is also known as salt and pepper noise.

3.1.1 Fixed-Valued Impulse Noise

Fixed-Valued Impulse Noise, also known as salt and pepper noise, is a type of noise where a corrupted pixel is digitized to the maximum or the minimum possible intensity value in the dynamic range. In a more rigorous sense, the model is defined as

$$c_{x,y} = \begin{cases} i_{min}, & p \\ i_{max}, & q \\ o_{x,y}, & 1-(p+q) \end{cases} \tag{3.1}$$

where o is the original image and c is the contaminated image. $c_{x,y}$ and $o_{x,y}$ are intensity values at coordinates (x, y) in c and o, respectively. p and q represents the probability of corruption for both extrema, i_{min} (pepper) and i_{max} (salt), respectively. The probability for i_{min} (p) and i_{max} (q) are generally considered as equal and their sum is the noise density. In practice, i_{min} and i_{max} correspond to 0 and 2^{N-1}, respectively. N is the number of bits representing the image pixels. For instance, the extreme values for commonly used 8-bit monochrome images are 0 or 255. Figure 3.1 illustrates an image contaminated by FVIN.

Figure 3.1: Lena image contaminated by %10 FVIN.

3.1.2 Random-Valued Impulse Noise

Random-Valued Impulse Noise can be manifested with any intensity value between the dynamic range of an image. Therefore, the detection of this model is more difficult compared to FVIN. RVIN is generally modeled by Bernoulli uniform noise model [1]. RVIN is defined as

$$c_{x,y} = \begin{cases} i, & p \\ o_{x,y}, & 1-p \end{cases} \tag{3.2}$$

where $i \sim U[i_{min}, i_{max}]$. $U[\cdot]$ represents a uniform random variable that can be valuated between i_{min} and i_{max}. $o_{x,y}$ and $c_{x,y}$ represent the pixel intensities at coordinates (x,y) in the original and contaminated image, respectively. p is the noise density. An illustration of RVIN can be seen in Figure 3.2.

Theoretically, a corrupted pixel can take any value between the dynamic range, however, since the interference to the image signal is impulsive, it is expected that the intensity values of corrupted pixels should be on one of the both ends of the dynamic range. A realistic RVIN model is proposed in [25]. In this model, the possible intensity value is defined with a length parameter m on both sides of the dynamic range, e.g., [0 9] and [246 255]. This model is defined as

$$c_{x,y} = \begin{cases} [i_{min}\ m], & \frac{p}{2m} \\ o_{x,y}, & 1-p \\ [i_{max}-m\ i_{max}], & \frac{p}{2m} \end{cases} \tag{3.3}$$

where m represents the range on both sides. In this model, all possible intensity values on the ranges have equal probability of occurrence.

Figure 3.2: Lena image contaminated by %10 RVIN.

3.2 Concept of Linearity

A filter can be considered as an operator whose operands are image segments or entire images. An operator is said to be linear if it satisfies *Additivity* and *Homogeneity* properties. Consider an operator ξ

$$\xi[i_{x,y}] = o_{x,y} \tag{3.4}$$

where i is an input; o is an output image. ξ is linear if

$$\begin{aligned} \xi[c_i f_{x,y} + c_j g_{x,y}] &= c_i \xi[f_{x,y}] + c_j \xi[g_{x,y}] \\ &= c_i f_{x,y} + c_j g_{x,y} \end{aligned} \tag{3.5}$$

where c_i, c_j are arbitrary constants; f and g are arbitrary images with same size. Equation (3.5) states that the output of applying a linear operation on summation of two

inputs is same as applying the linear operation individually and then summing the results. In addition, when inputs of a linear operation multiplied by a factor(c_i and c_j), then the result is equal to multiplication of the factor and the operation due to its inputs. Both *Additivity* and *Homogeneity* are satisfied when (3.5) holds true for any operator ξ.

Consider order-statistics operator *maximum* (in short *max*) which finds maximum value on a given image segment. Consider the following two image segments

$$f = \begin{pmatrix} 0 & 2 & 7 \\ 3 & 4 & 8 \\ 5 & 1 & 6 \end{pmatrix} \quad g = \begin{pmatrix} 6 & 5 & 1 \\ 4 & 7 & 8 \\ 1 & 3 & 9 \end{pmatrix} \tag{3.6}$$

In order to test linearity, let $c_i = 1$ and $c_j = -1$. According to *Additivity* and *Homogeneity* properties, the following two equations must yield the same result.

$$max\left\{ c_i \begin{pmatrix} 0 & 2 & 7 \\ 3 & 4 & 8 \\ 5 & 1 & 6 \end{pmatrix} + c_j \begin{pmatrix} 6 & 5 & 1 \\ 4 & 7 & 8 \\ 1 & 3 & 9 \end{pmatrix} \right\} \tag{3.7}$$

$$c_i\, max\left\{ \begin{pmatrix} 0 & 2 & 7 \\ 3 & 4 & 8 \\ 5 & 1 & 6 \end{pmatrix} \right\} + c_j\, max\left\{ \begin{pmatrix} 6 & 5 & 1 \\ 4 & 7 & 8 \\ 1 & 3 & 9 \end{pmatrix} \right\} \tag{3.8}$$

However, it is clear that (3.7) yields 7 and (3.8) yields 6. Thus, *max* operation is not linear.

Chapter 4

THE PROPOSED METHOD

This chapter proposes a novel filter for impulse noise removal, namely, Unbiased Weighted Mean Filter (UWMF). UWMF addresses our observation regarding a particular problem of impulse noise removal filters in the literature. We call this problem *spatial bias*. In Section 4.1, the nature of spatial bias and a method to eliminate spatial bias is presented. In Section 4.2, further details of UWMF are given.

4.1 Spatial Bias

Filters employing a weight function can compute the optimal estimation of the original intensity value only when the pixel under consideration is the only corrupted pixel in the filtering window. In other words, an optimal estimation can be made only when all pixels in the filtering window contribute to the estimation. However, even at lower noise densities such as %10, it is likely to encounter another corrupted pixel other than the one in the center of the filtering window. It is intuitive to consider possibilities of better estimations with respect to spatial relationship between pixels in an image. An important question is then arisen as how a better estimation can be made.

Another important observation regarding many filters in the literature is that there is a selection of uncorrupted pixels before or during filtering stage. The intensity values of corrupted pixels, assuming that the detection is accurate, can not be used. However, this kind of selective approach also ignores the positional information of corrupted pixels in the filtering window and effectively leads into a spatial bias towards uncorrupted pixel. For example, if corrupted pixels are clustered in a particular region in the filtering window, then, the contribution of pixels where corruption is not dense will be higher.

The relation between weight function and spatial bias is mutual, that is, the weights can be recalibrated to eliminate spatial bias and the information supplied by positional distribution (i.e., spatial bias) can be exploited to recalibrate weights, thus, a better estimation can be made. If a weight function assigns weights symmetrically (e.g., Euclidean Distance), then asymmetric distribution of corrupted pixels from a spatial perspective will also cause a bias.

4.1.1 Elimination of Spatial Bias

Elimination of spatial bias can be related to a physical system. Consider the filtering window as a uniform grid and the central cell to be the center of gravity. Each cell on the grid represents a pixel and each cell has its own associated weight. If a weight is removed from the grid, the center of gravity would shift away from it. The same shift occurs when a pixel is ignored. Estimations conducted on such configuration yield the estimation of a different position (center of gravity of uncorrupted pixels) instead of the actual center.

The primary objective of spatial bias elimination is to recalibrate pixel weights present in the filtering window in such a way that the center of gravity shifts back to actual center. When a pixel is corrupted, its position information and weight can still be exploited. Spatial bias can be eliminated by a combination of two operations using this information; increasing the weights that are further away from the shifted center of gravity and decreasing those weights that are close to it.

Spatial bias in a filtering window can be represented as a vector $\vec{g}$ by simply calculating the center of gravity with

$$\vec{g} = \sum_{x,y} w_{x,y} \vec{d}_{x,y} \tag{4.1}$$

where $\vec{d}_{x,y}$ is the radial vector from the center of the filtering window to coordinates (x,y) and $w_{x,y}$ is the weight at (x,y). The recalibration is achieved by solving the following linear equation with one unknown variable

$$\sum_{x,y} [w_{x,y} + w_{x,y}(\vec{d}_{x,y} \cdot \vec{g}')]\vec{d}_{x,y} = 0 \tag{4.2}$$

where $\vec{g}'$ is the counter-bias vector. Upon expanding summation and dot product of $\vec{d}_{x,y}$ and $\vec{g}'$, we obtain

$$\sum_{x,y} [w_{x,y}(g'^x d^x_{x,y} + g'^y d^y_{x,y})]\vec{d}_{x,y} = -\sum_{x,y} w_{x,y}\vec{d}_{x,y} \tag{4.3}$$

where superscripts represent the dimensional components of the vectors. Rewriting (4.3) with respect to the dimensional components yields the following two equations.

$$\begin{aligned} g'^x \sum_{x,y} w_{x,y}\left(d^x_{x,y}\right)^2 + g'^y \sum_{x,y} w_{x,y} d^x_{x,y} d^y_{x,y} &= -\sum_{x,y} w_{x,y} d^x_{x,y} \\ g'^y \sum_{x,y} w_{x,y}\left(d^y_{x,y}\right)^2 + g'^x \sum_{x,y} w_{x,y} d^x_{x,y} d^y_{x,y} &= -\sum_{x,y} w_{x,y} d^y_{x,y} \end{aligned} \tag{4.4}$$

After solving the linear system in (4.4), we obtain

$$\begin{aligned} P &= \sum_{x,y} w_{x,y}\left(d^x_{x,y}\right)^2 \\ Q &= \sum_{x,y} w_{x,y} d^x_{x,y} d^y_{x,y} \\ R &= -\sum_{x,y} w_{x,y} d^x_{x,y} \\ S &= \sum_{x,y} w_{x,y}\left(d^y_{x,y}\right)^2 \\ T &= -\sum_{x,y} w_{x,y} d^y_{x,y} \\ g'^y &= \frac{(PT)+(QR)}{-Q^2+(PS)} \\ g'^x &= -\frac{(Qg'^y)+R}{P} \end{aligned} \tag{4.5}$$

In (4.2), the dot product of terms $w_{x,y}$, $\vec{d}_{x,y}$ and $\vec{g}'$ is maximum when $\vec{d}_{x,y}$ and $\vec{g}'$ point to the same direction; minimum when these vectors point to opposite directions of each other. Therefore, weights located at opposite direction of $\vec{g}$, relative to the shifted center

of gravity, i.e., the weights closer to the corrupt pixels, will contribute more. Thus, the lost information will be compensated by increased contribution of these pixels. If contamination is dense in a particular region of the filtering window, some of the weights may be negative after recalibration.

An important property of this procedure is its applicability to different filters. As it has been stated, the spatial bias arises due to asymmetric distribution of corrupted pixels and such randomness is an inherent feature of impulse noise. Weighted mean and median filters are good candidates for employing the spatial bias elimination. However, even for methods without a weight assignment, it can be employed by assuming weights as one.

4.2 Noise Restoration

The proposed method performs three operations in a sequential manner while convolving a filtering window over a contaminated image. These are noise detection, bias elimination and calculation of new intensity value. Unbiased Weighted Mean Filter employs NND, that is, pixel is identified as noise if it has the highest or the lowest intensity value. After noise detection and spatial bias elimination, the noise is restored. The last two operations are performed in a rectangular window that is centered on corrupt pixel upon noise detection. Let N_{FW} denote filtering window size. Let I and W to be the sets of intensity values and weights in the filtering window $(-(N_{FW}-1)/2 \leq x,y \leq N_{FW}-1)/2)$ respectively and let $i_{x,y}$ denote the intensity value; $w_{x,y}$ represent the weight of a pixel located at coordinates (x,y) in I and W, respectively.

Similarly, let $w'_{x,y}$ denote the recalibrated weight (described in Sec. 4.1.1) in W' at same coordinates. Considering (i,j) as the central location in the filtering window and $o_{i,j}$ as the intensity value at these coordinates, the new intensity value of $o_{i,j}$ is computed with the following function

$$
\begin{aligned}
&w_{x,y} = [\Phi((x,y),(i,j))]^{-k} \\
&w'_{x,y} = w_{x,y} + w_{x,y}(\vec{d}_{x,y} \cdot \vec{g'}) \\
&I_{255} = \{(x,y) \,|\, i_{x,y} = 255 \wedge i_{x,y} \in I\} \\
&I_0 = \{(x,y) \,|\, i_{x,y} = 0 \wedge i_{x,y} \in I\}
\end{aligned}
\tag{4.6}
$$

$$
o_{i,j} = \begin{cases}
i_{i,j}, & i_{i,j} \notin I_{255} \wedge i_{i,j} \notin I_0 \\
255, & I \subseteq (I_{255} \cup I_0) \wedge c(I \cap I_{255}) > c(I \cap I_0) \\
0, & I \subseteq (I_{255} \cup I_0) \wedge c(I \cap I_{255}) \leq c(I \cap I_0) \\
\dfrac{\sum\limits_{i_{x,y} \in I \wedge w'_{x,y} \in W'} i_{x,y} w'_{x,y}}{\sum\limits_{w'_{x,y} \in W'} w'_{x,y}}, & \text{otherwise}
\end{cases}
$$

where c is a function which counts the number of elements in a set and $\Phi((x,y),(i,j))$ is a distance function between central location (i,j) and coordinates (x,y), $\vec{g'}$ is the counter-bias vector calculated in (4.2). I_{255} represents the set of intensity values equal to 255 and I_0 represents the set of intensity values equal to 0. In this study, we have used Minkowski Distance defined as

$$
\mathrm{D}_{Minkowski}(Q,R) = \left(\sum_{i=1}^{N} |Q_i - R_i|^p \right)^{1/p} \tag{4.7}
$$

where p is a parameter of distance metric. When p is 1 or 2, it corresponds to Manhattan or Euclidean Distances, respectively. These two values for p is found to be yielding the highest restoration performance (explained in Sec. 5.4). k is a parameter of the proposed method which controls the effect of weights based on the distance to the central pixel. The pixels that are further away relative to the center of the filtering window are less spatially-correlated than those that are close. Thus, it is important to diminish contribution of distant pixels. According to the empirical results, the value of k is estimated

to be between 4 and 6; and p to be 1 (Manhattan Distance). Details of parameter effects are presented under Section 5.4.

There are four different cases while assigning a new value to $o_{i,j}$. The first one is identity mapping when pixel is not identified as noise. The second and the third cases occur only when there is no uncorrupted pixel in the filtering window, i.e., all pixels are either black ($i_{x,y} \in I_0$) or white ($i_{x,y} \in I_{255}$). In such a case, the proposed method counts the number of black and white pixels and assigns whichever has higher occurrence in the filtering window. Finally, the fourth case is where $o_{i,j}$ is assigned to a weighted mean using the recalibrated weights. Due to linearity of recalibration and randomness of noise distribution, various numerical instabilities, e.g., weighted means falling outside of intensity range, may occur. In such cases, clipping or usage of the original weights are two possible solutions. We have estimated the occurrence of these cases to be statistically insignificant ($p \leq 0.00006$). The pseudo of the proposed method is illustrated in Algorithm 1.

A snapshot of restoration process is presented as an example in Fig. 4.1. In the upper-right matrix (Fig. 4.1b), the uncorrupted pixels are clustered around the left region of the filtering window. This configuration shifts the gravity towards left. After recalibration (Fig. 4.1d), the weights of pixels are decreased in the left region; increased in the right region. The recalibrated configuration satisfies Equation (4.2). In this particular snapshot, the original intensity value is 162. The weighted mean using the original weights is calculated to be 197 whereas the proposed method calculates a weighted mean of 164.

$$\begin{bmatrix} 192 & 105 & 78 & 81 & 78 \\ 208 & 163 & 117 & 89 & 72 \\ 224 & 205 & 162 & 103 & 64 \\ 225 & 221 & 205 & 159 & 89 \\ 227 & 228 & 222 & 198 & 133 \end{bmatrix}$$

(a)

$$\begin{bmatrix} 192 & 105 & 255 & 81 & 78 \\ 208 & 163 & 255 & 89 & 255 \\ 224 & 205 & 255 & 255 & 255 \\ 225 & 0 & 255 & 255 & 89 \\ 227 & 228 & 222 & 198 & 133 \end{bmatrix}$$

(b)

$$\begin{bmatrix} 0.004 & 0.012 & 0.063 & 0.012 & 0.004 \\ 0.012 & 0.063 & 1 & 0.063 & 0.012 \\ 0.063 & 1 & 0 & 1 & 0.063 \\ 0.012 & 0.063 & 1 & 0.063 & 0.012 \\ 0.004 & 0.012 & 0.063 & 0.012 & 0.004 \end{bmatrix}$$

(c)

$$\begin{bmatrix} -0.001 & 0.005 & 0.063 & 0.014 & 0.010 \\ -0.005 & 0.020 & 1 & 0.111 & 0.012 \\ -0.029 & 0.270 & 0 & 1 & 0.063 \\ -0.006 & 0.063 & 1 & 0.063 & 0.030 \\ -0.002 & 0.002 & 0.056 & 0.011 & 0.009 \end{bmatrix}$$

(d)

Figure 4.1: A snapshot of restoration process from Lena image contaminated by 30% noise. (a) Original filtering window. (b) Filtering window after contamination. (c) Original weights. (d) Recalibrated weights.

Algorithm 1 The pseudo code of the proposed method.

for $o_{i,j} \in$ image **do**
  if $o_{i,j} \neq 255 \wedge o_{i,j} \neq 0$ **then**
    $o_{i,j} \leftarrow o_{i,j}$, continue
  else
    $\mathrm{sum_i} \leftarrow 0$, $\mathrm{sum_{w'}} \leftarrow 0$
    $\mathrm{count}_{255} \leftarrow 0$, $\mathrm{count}_0 \leftarrow 0$
    $P \leftarrow 0$, $Q \leftarrow 0$, $R \leftarrow 0$, $S \leftarrow 0$, $T \leftarrow 0$
    for $i_{x,y} \in I \wedge \notin (I_0 \cup I_{255})$ **do**
      $d^x_{x,y} \leftarrow x - i$
      $d^y_{x,y} \leftarrow y - j$
      $w_{x,y} \leftarrow \left[\left(|x-i|^p + |y-j|^p \right)^{1/p} \right]^{-k}$
      $P \leftarrow P + w_{x,y} (d^x_{x,y})^2$
      $Q \leftarrow Q + w_{x,y} d^x_{x,y} d^y_{x,y}$
      $R \leftarrow R + w_{x,y} d^x_{x,y}$
      $S \leftarrow S + w_{x,y} (d^y_{x,y})^2$
      $T \leftarrow T + w_{x,y} d^y_{x,y}$
    end for
    $R \leftarrow -R$
    $T \leftarrow -T$
    $g'_y \leftarrow \frac{(PT)+(QR)}{-Q^2+(PS)}$
    $g'_x \leftarrow -\frac{(Qg'^y)+R}{P}$
    for $i_{x,y} \in I$ **do**
      if $i_{x,y} = 255$ **then**
        $\mathrm{count}_{255} = \mathrm{count}_{255} + 1$
      else if $i_{x,y} = 0$ **then**
        $\mathrm{count}_0 = \mathrm{count}_0 + 1$
      else
        $w'_{x,y} \leftarrow w_{x,y} + [w_{x,y}(g'^x d^x_{x,y} + g'^y d^y_{x,y})]$
        $\mathrm{sum_{w'}} = \mathrm{sum_{w'}} + w'_{x,y}$
        $\mathrm{sum_i} = \mathrm{sum_i} + i_{x,y} w'_{x,y}$
      end if
    end for
    if $\mathrm{count}_{255} + \mathrm{count}_0 = {N_{FW}}^2$ **then**
      if $\mathrm{count}_{255} > \mathrm{count}_0$ **then**
        $o_{i,j} \leftarrow 255$
      else
        $o_{i,j} \leftarrow 0$
      end if
    else
      $o_{i,j} \leftarrow \frac{sum_i}{sum_{w'}}$
    end if
  end if
end for

Chapter 5

EXPERIMENTAL RESULTS

This chapter provides the main findings of conducted experiments and their interpretation. There are three sections under this chapter. In Section 5.1, details of experimental methodology and metrics that are used for qualitative measurements are given. The analysis of noise detection performance are under Section 5.2. In Section 5.3, the restoration performance and computational complexity of the state-of-the-art methods are analyzed. Finally, in Section 5.4, the effect of various parameters of the proposed method is analyzed in order to find their optimal values.

5.1 Experimental Methodology

In simulations, two impulse noise models — Fixed-Valued Impulse Noise (FVIN) and Random-Valued Impulse Noise (RVIN) — are tested. Four distinct experiments are conducted in order to assess the performance of the various state-of-the-art impulse noise removal methods and our proposed method Unbiased Weighted Mean Filter (UWMF). These experiments are conducted in images contaminated by FVIN and involve comparison of noise detection performance, noise removal (restoration) performance, computational complexity, CPU time, and finally analysis of parameters for the proposed method. The methods that are implemented for comparison purposes are Interpolation-based Impulse Noise Removal (IBINR) [17], Improved Boundary Discriminative Noise Detection Filtering Algorithm (IBDND) [16], Cloud Model Filter (CMF) [42], Modified Decision Based Unsymmetric Trimmed Median Filter (MDBUTMF) [8], Adaptive Median Filter (AMF) [23] and finally Standard Median Filter (SMF) [11]. In the second experiment, the detection methods of CMF and BDND are compared with NND. Furthermore, aforementioned noise detection methods are tested with images contaminated

by RVIN and according to these results, best candidate is incorporated into the proposed method (UWMF) in order to assess the restoration performance on this noise model.

All methods are implemented in C++ programming language and the simulations are performed on a computer with Intel i7 CPU and 8GB of RAM. Eight commonly used 8-bit monochrome test images are used for experiments. The test images House and Checkerboard are 256×256 and the rest of the images are 512×512 in size. In addition, all filters are tested on medical, astronomical and other types of images with varying sizes. In order to prevent occasional peaks in simulation results, average of 10 iterations is taken for all experiments.

In all experiments, images are contaminated with different noise densities, ranging from 10% to 90%, with equal probability for both salt and pepper noise. Suggested window sizes are used for all methods and no pixels around the edges are ignored while calculating the performance and for CMF, δ is taken as 1. In the third experiment, the effect of different distance metrics based on the value of p, various values for k and window size are tested. In experiments involving RVIN, the model in (3.3) is used and m is taken as 5.

5.1.1 Performance Metrics

In the first experiment, two types of detection errors — Misdetection (MD) and False Alarm (FA) — are used to measure detection performance. They are defined as

$$\alpha_{MD}(v_1, v_2, v_3) = \begin{cases} 1, & v_1 \neq v_2 \wedge v_3 = 0 \\ 0, & otherwise \end{cases} \tag{5.1}$$
$$MD = \sum_{x,y} \alpha_{MD}(o_{x,y}, c_{x,y}, n_{x,y})$$

$$\alpha_{FA}(v_1, v_2, v_3) = \begin{cases} 1, & v_1 = v_2 \wedge v_3 = 1 \\ 0, & otherwise \end{cases} \tag{5.2}$$
$$FA = \sum_{x,y} \alpha_{FA}(o_{x,y}, c_{x,y}, n_{x,y})$$

where o is the original, c is the contaminated image. n is a binary noise map. α_{MD} and α_{FA} are MD and FA decision functions, respectively. The error rates (%) for MD and FA are given by

$$\begin{aligned} R_{MC} &= \tfrac{MD}{MN} \cdot 100 \\ R_{FA} &= \tfrac{FA}{MN} \cdot 100 \end{aligned} \tag{5.3}$$

where M and N are image dimensions. Note that the summation of MD and FA errors yields total detection error.

Noise detection is imperative to successful impulse noise removal. The problems of detection can be analyzed from perspective of two types of errors. MD prevents a corrupted pixel to be restored. Since corruption tends to be relatively different in terms of intensity value compared to local uncorrupted pixels, it is natural to expect such errors decrease the restoration quality drastically. On the other hand, in the case of FA, an uncorrupted pixel is subjected to restoration procedure. The primary aim restoration procedure is to guess the original intensity value of corrupted pixel with information at hand. In consideration of this fact, FA is a less destructive detection error compared to MD in terms of both objective and subjective assessments.

In the second experiment where noise removal capabilities are tested, three well-known performance metrics are used. These are Peak Signal-to-Noise Ratio (PSNR) in decibels, Mean Absolute Error (MAE) and Structural Similarity (SSIM). It is important to compare the methods in terms of Structural Similarity (SSIM) as visual quality of a restored image may not be in parallel with restoration results in terms of PSNR or Mean Absolute Error (MAE). PSNR, SSIM and MAE are defined as

$$PSNR = 10\log_{10}\frac{255^2}{\frac{1}{MN}\sum_{x,y}(o_{x,y}-r_{x,y})^2} \tag{5.4}$$

$$\begin{aligned} SSIM =& \frac{(2\mu_o\mu_r+c_1)(2\Sigma+c_2)}{(\mu_o^2+\mu_r^2+c_1)(\sigma_o^2+\sigma_r^2+c_2)}, \\ \mu =& \frac{1}{MN}\sum_{x,y}o_{x,y}, \\ \sigma^2 =& \frac{1}{MN}\sum_{x,y}(o_{x,y}-\mu)^2, \\ \Sigma =& \frac{1}{MN}\sum_{x,y}o_{x,y}r_{x,y}-\sigma_o\sigma_r \end{aligned} \tag{5.5}$$

$$MAE = \frac{1}{MN}\sum_{x,y}|o_{x,y}-r_{x,y}| \tag{5.6}$$

where $M \times N$ is the size for original and restored images o and r. μ_o and μ_r are means; σ_o^2 and σ_r^2 are variances for two images and Σ is their covariance. Finally, c_1 and c_2 are stabilization constants.

Computational complexity analysis has been conducted using O-notation. It shows how an algorithm scales with various parameters as those parameters tend to infinity. For practical algorithm speed assessment, CPU time is used.

5.2 Noise Detection Performance

The detection performance of NND, CMF and BDND are presented in Table 5.1. In that table, R_{TOTAL} is summation of R_{MD} and R_{FA} defined in (5.3). However, none of the methods produced false alarm. For this reason, false alarm metric is excluded from Table 5.1. NND naïvely assumes that corrupted pixels are either black or white, or in

a more generalized sense, the pixels with maximum or minimum intensity value in the dynamic rage. This situation makes NND suitable for images where extreme intensity values are absent. The success of NND can be seen for Lena, Barbara and House images in Table 5.1. In these images, NND operates with perfect detection accuracy while other methods result with MD. All methods provide robust detection performance across a wide range of noise densities with total misclassification rate less than 1%. Among three detection algorithms, NND slightly outperforms the other two. In most cases, such performance difference will not make a significant improvement in terms of restoration quality.

In experiments, we have observed that none of the noise detection algorithms could operate on binary images (e.g., Checkerboard). This is due to the fact that these algorithms are not designed for binary images. Figure 5.1 illustrates original and contaminated Checkerboard image. NND completely fails as it considers entire binary image to be noise. CMF and BDND also fail because these methods can not find proper upper and lower intensity bounds for noise detection. One solution to this problem is counting the number of corrupted pixels in a filtering window and acting accordingly if there is no uncorrupted pixel. IBINR and MDBUTMF are two methods solving this issue with such approach.

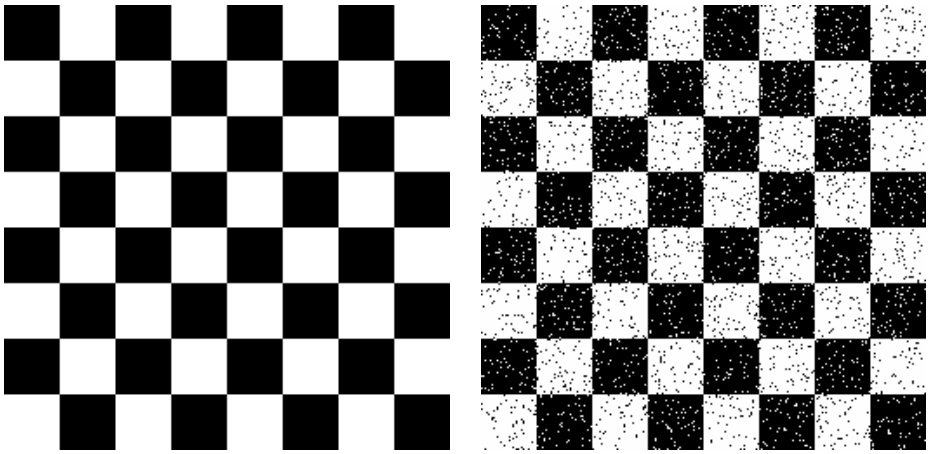


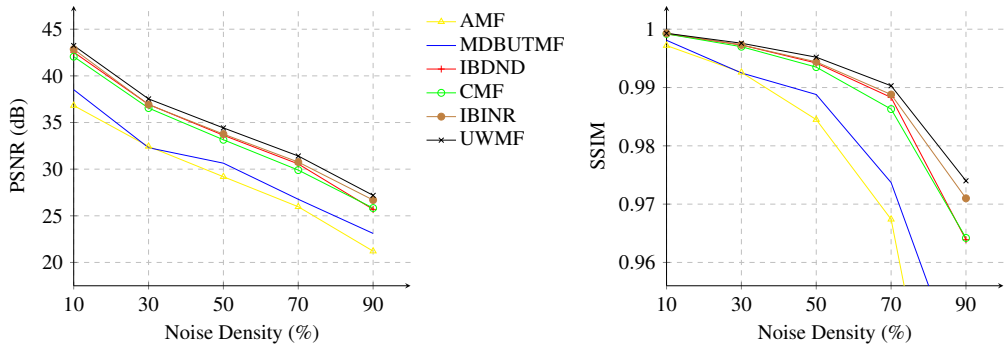
Figure 5.1: Original and contaminated (%30) Checkerboard image.

Table 5.1: Comparison of FVIN detection performance of NND, CMF and BDND.

		NND		CMF		BDND	
Image	Noise	MD	Total (R_{TOTAL})	MD	Total (R_{TOTAL})	MD	Total (R_{TOTAL})
Lena	10	0	0(0%)	44	44(0.0168%)	0	0(0%)
	30	0	0(0%)	8	8(0.0031%)	1	1(0.0004%)
	50	0	0(0%)	1	1(0.0004%)	7	7(0.0027%)
	70	0	0(0%)	0	0(0%)	17	17(0.0065%)
	90	0	0(0%)	0	0(0%)	261	261(0.0996%)
Peppers	10	1	1(0.0004%)	57	57(0.0217%)	918	918(0.3502%)
	30	1	1(0.0004%)	10	10(0.0038%)	688	688(0.2625%)
	50	1	1(0.0004%)	1	1(0.0004%)	714	714(0.2724%)
	70	1	1(0.0004%)	1	1(0.0004%)	599	599(0.2285%)
	90	0	0(0%)	0	0(0%)	719	719(0.2743%)
Baboon	10	24	24(0.0092%)	46	46(0.0175%)	393	393(0.1499%)
	30	18	18(0.0069%)	20	20(0.0076%)	315	315(0.1202%)
	50	20	20(0.0076%)	20	20(0.0076%)	255	255(0.0973%)
	70	17	17(0.0065%)	17	17(0.0065%)	130	130(0.0496%)
	90	10	10(0.0038%)	10	10(0.0038%)	215	215(0.0820%)
Barbara	10	0	0(0%)	16	16(0.0061%)	1	1(0.0004%)
	30	0	0(0%)	3	3(0.0011%)	4	4(0.0015%)
	50	0	0(0%)	0	0(0%)	9	9(0.0034%)
	70	0	0(0%)	0	0(0%)	33	33(0.0126%)
	90	0	0(0%)	0	0(0%)	304	304(0.1160%)
Boat	10	6	6(0.0023%)	22	22(0.0084%)	136	136(0.0519%)
	30	6	6(0.0023%)	6	6(0.0023%)	121	121(0.0462%)
	50	6	6(0.0023%)	6	6(0.0023%)	140	140(0.0534%)
	70	2	2(0.0008%)	2	2(0.0008%)	96	96(0.0366%)
	90	6	6(0.0023%)	6	6(0.0023%)	337	337(0.1286%)
Bridge	10	1327	1327(0.5062%)	1442	1442(0.5501%)	3159	3159(1.2051%)
	30	1200	1200(0.4578%)	1205	1205(0.4597%)	1781	1781(0.6794%)
	50	1080	1080(0.4120%)	1081	1081(0.4124%)	1707	1707(0.6512%)
	70	914	914(0.3487%)	914	914(0.3487%)	1308	1308(0.4990%)
	90	752	752(0.2869%)	752	752(0.2869%)	1237	1237(0.4719%)
House	10	0	0(0%)	10	10(0.0038%)	217	217(0.0828%)
	30	0	0(0%)	0	0(0%)	184	184(0.0702%)
	50	0	0(0%)	0	0(0%)	161	161(0.0614%)
	70	0	0(0%)	0	0(0%)	66	66(0.0252%)
	90	0	0(0%)	0	0(0%)	114	114(0.0435%)

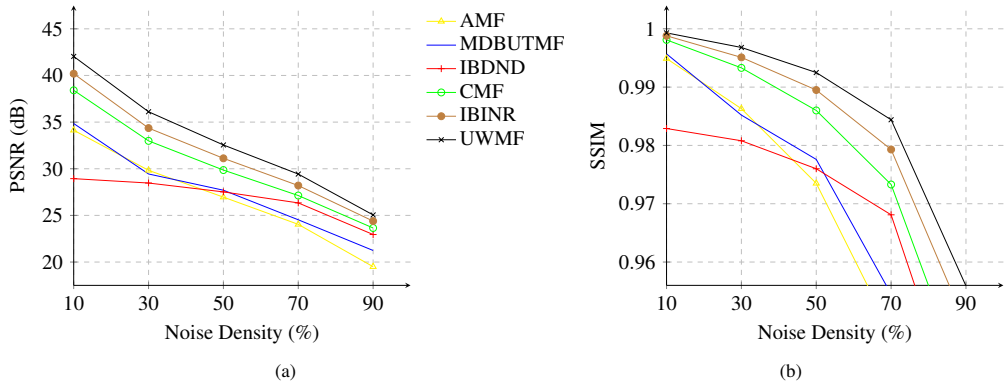
5.3 Noise Restoration Performance & Computational Complexity

The restoration capabilities of the state-of-the-art methods and the proposed method (UWMF) are compared across a wide range of noise densities, ranging from 10% to %90, using different qualitative assessment methods. In addition, all methods are compared visually. Figures 5.2 and 5.3 present the comparison of selected methods in terms of PSNR and SSIM for Lena and House images, respectively. UWMF has the highest restoration performance for both metrics. MDBUTMF, IBINR, IBDND and CMF have similar results and their results are close to the performance of UWMF. However, AMF is significantly outperformed by other methods. In House image, the performance difference between the proposed method and the other methods is even greater. This is due to the fine detail preservation capabilities of UWMF in both smooth and edge regions.



(a) (b)

Figure 5.2: Comparison of the state-of-the-art methods and UWMF in terms of PSNR (in dB) and SSIM on Lena image.



(a) (b)

Figure 5.3: Comparison of the state-of-the-art methods and UWMF in terms of PSNR (in dB) and SSIM on House image.

In order to evaluate restoration performance visually, restored Lena image and four magnified regions within this image are illustrated in Figure 5.4. The proposed method preserves the fine details in smooth and edge regions compared to the other methods. This is due to unbiased nature of UWMF, that is, the intensity value estimations of corrupted pixels reflect the local structure of the image, effectively resulting with superior restoration quality. In the lower-left region where area around nose and mouth is magnified, a better restoration quality is evident for image restored by the proposed method. Although CMF, IBDND and IBINR results with slightly lower restoration performance quantitatively, the difference in terms of visual results is great between UWMF and the other methods.

For further analysis of restoration capabilities of the impulse noise removal methods in consideration, a smaller region around nose (from dorsal to the end of sidewalls) is mapped in Figure 5.5. AMF is excluded from this experiment as it has the lowest restoration performance. From a topographical viewpoint, the proposed method (Fig. 5.5f) has the best estimation of the original image (Fig. 5.5a). In the hat and shoulder regions (upper-right and lower-right magnifications, respectively), although jagged edges occur in all of the restored images because of high noise density (90%), the proposed method results the least amount of disturbance in the edges. MDBUTMF (Fig. 5.5b)

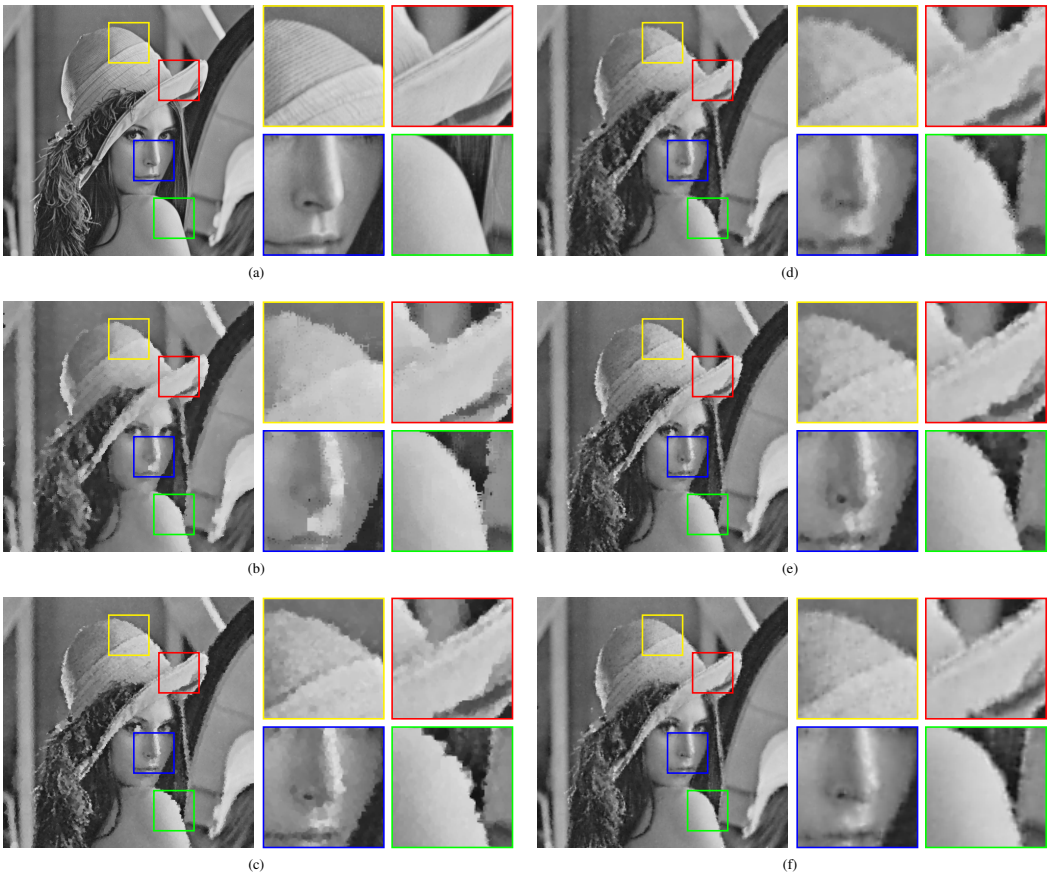


Figure 5.4: Visual results of Lena and various magnified regions (90% contamination) restored with the state-of-the-art methods and UWMF. (a) Original image. (b) MDBUTMF [8]. (c) IBDND [16]. (d) CMF [42]. (e) IBINR [17]. (f) UWMF.

and IBDND (Fig. 5.5c) have the poorest preservation of the local structure compared to the topographical map of the original image.

The simulation results for different metrics are presented in Tables 5.2, 5.3 and 5.4. Similarly, the visual results of restored images are shown in Figure 5.6. As mentioned earlier, IBDND and CMF fail to restore Checkerboard image since the detection procedure for these methods are not suitable for binary images and they do not have a handling mechanism for a case where all pixels in a filtering window are corrupted. Although NND is employed by IBINR, MDBUTMF and the proposed method, it is also not suitable for binary images. However, these methods have handling mechanism when no uncorrupted pixels are found in the filtering window.

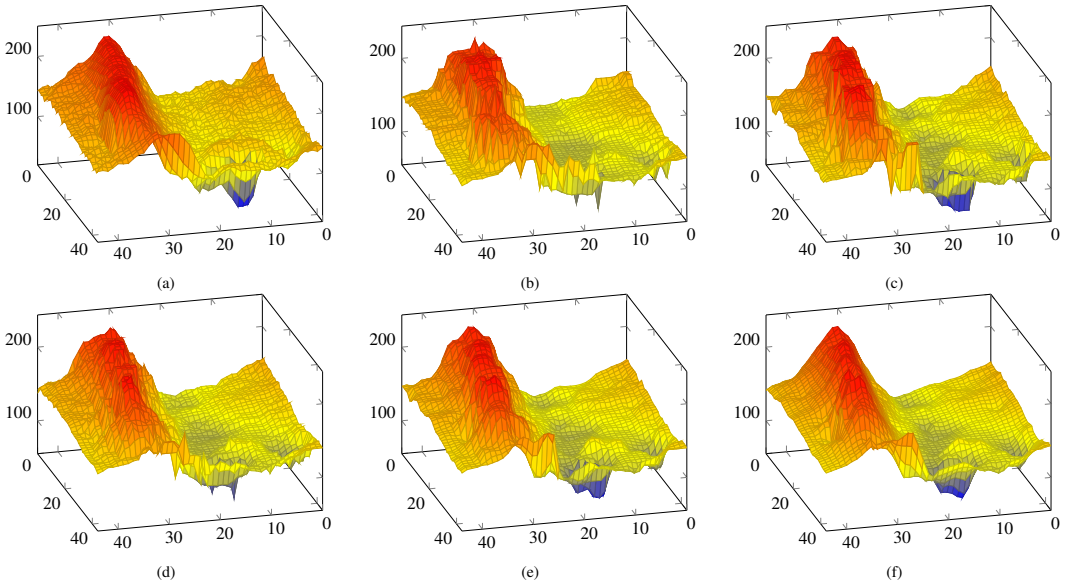


Figure 5.5: Comparison of local topographical restoration map of the selected methods around nose region on Lena image with 90% contamination. (a) Original image. (b) MDBUTMF [8]. (c) IBDND [16]. (d) CMF [42]. (e) IBINR [17]. (f) UWMF.

Table 5.2: Detailed comparison of the state-of-the-art methods and UWMF in terms of PSNR (in dB), SSIM and MAE.

		SMF			AMF			MDBUTMF			IBDND			CMF			IBINR			UWMF		
Image	Noise	PSNR	SSIM	MAE	PSNR	SSIM	MAE	PSNR	SSIM	MAE	PSNR	SSIM	MAE	PSNR	SSIM	MAE	PSNR	SSIM	MAE	PSNR	SSIM	MAE
Lena	10	30.29	0.9876	4.35	36.82	0.9972	1.03	42.48	0.9992	0.38	42.54	0.9992	0.38	42.06	0.9992	0.40	42.86	0.9993	0.37	43.25	0.9993	0.36
	30	28.73	0.9823	5.00	32.44	0.9926	2.02	36.28	0.9969	1.28	36.94	0.9973	1.22	36.55	0.9970	1.26	36.92	0.9973	1.22	37.53	0.9976	1.16
	50	23.68	0.9415	8.97	29.19	0.9845	3.49	31.77	0.9912	2.61	33.63	0.9942	2.23	33.15	0.9935	2.33	33.76	0.9944	2.21	34.42	0.9952	2.09
	70	18.93	0.8147	17.00	25.97	0.9674	5.73	29.12	0.9837	4.10	30.56	0.9883	3.62	29.90	0.9863	3.87	30.76	0.9888	3.57	31.41	0.9903	3.36
	90	18.93	0.8147	17.00	21.21	0.9020	10.90	24.96	0.9571	7.31	25.69	0.9639	6.69	25.83	0.9642	6.88	26.69	0.9710	6.22	27.19	0.9740	5.93
Chessboard	10	21.02	0.9842	2.02	15.90	0.9485	6.58	16.83	0.9524	20.88	N/A	N/A	N/A	N/A	N/A	N/A	24.50	0.9929	0.91	24.44	0.9928	0.92
	30	16.29	0.9531	5.99	12.79	0.8946	13.46	12.99	0.8662	44.57	N/A	N/A	N/A	N/A	N/A	N/A	17.27	0.9625	4.79	17.25	0.9623	4.81
	50	14.02	0.9210	10.10	10.95	0.8392	20.54	10.40	0.7059	71.81	N/A	N/A	N/A	N/A	N/A	N/A	14.12	0.9227	9.87	14.15	0.9232	9.81
	70	11.34	0.8533	18.74	8.92	0.7438	32.73	8.32	0.4721	94.16	N/A	N/A	N/A	N/A	N/A	N/A	11.17	0.8475	19.48	11.13	0.8460	19.67
	90	5.95	0.4921	64.88	5.51	0.4373	71.89	6.65	0.1540	117.66	N/A	N/A	N/A	N/A	N/A	N/A	6.69	0.5716	54.71	6.72	0.5748	54.32
Peppers	10	27.91	0.9852	4.55	35.02	0.9970	1.16	38.97	0.9986	0.44	30.97	0.9925	0.81	39.47	0.9988	0.46	40.50	0.9990	0.46	41.45	0.9991	0.45
	30	26.83	0.9812	5.19	30.89	0.9925	2.16	33.57	0.9958	1.44	30.07	0.9909	1.75	34.05	0.9962	1.45	34.91	0.9969	1.46	36.11	0.9976	1.40
	50	25.08	0.9717	6.46	27.98	0.9857	3.64	30.05	0.9909	2.70	28.70	0.9875	2.89	30.99	0.9926	2.62	31.84	0.9939	2.58	32.95	0.9952	2.46
	70	22.58	0.9488	9.13	25.00	0.9718	5.91	27.70	0.9844	4.22	27.28	0.9827	4.32	28.34	0.9864	4.20	29.10	0.9886	4.00	30.16	0.9910	3.79
	90	17.22	0.8116	19.72	20.27	0.9161	11.43	24.02	0.9635	7.35	23.89	0.9623	7.41	24.80	0.9690	7.22	25.51	0.9739	6.61	26.27	0.9779	6.32
Baboon	10	20.66	0.8352	15.53	26.30	0.9591	3.86	31.44	0.9875	1.42	29.61	0.9809	1.57	31.65	0.9881	1.43	32.52	0.9902	1.31	32.76	0.9908	1.27
	30	20.22	0.8201	16.32	23.82	0.9292	6.41	26.38	0.9596	4.45	26.28	0.9580	4.39	26.71	0.9621	4.39	27.34	0.9671	4.15	27.62	0.9693	4.01
	50	19.40	0.7792	18.37	21.58	0.8826	10.19	23.04	0.9101	8.58	24.00	0.9275	7.55	24.02	0.9279	7.74	24.57	0.9366	7.30	24.72	0.9389	7.15
	70	18.78	0.7367	20.47	19.44	0.8081	15.40	21.17	0.8598	12.60	21.96	0.8823	11.43	21.81	0.8758	11.92	22.26	0.8905	11.19	22.41	0.8937	11.02
	90	17.60	0.6343	24.49	16.97	0.6646	23.61	19.24	0.7726	18.24	19.08	0.7758	18.24	19.61	0.7844	17.74	19.69	0.7984	17.18	19.83	0.8027	16.96
Barbara	10	22.91	0.9401	9.93	28.14	0.9792	2.52	33.73	0.9916	0.89	34.63	0.9926	0.83	34.00	0.9920	0.89	34.95	0.9929	0.81	35.18	0.9931	0.77
	30	22.37	0.9333	10.67	25.90	0.9681	4.21	28.45	0.9804	2.87	29.47	0.9841	2.63	29.01	0.9827	2.75	29.56	0.9844	2.62	29.73	0.9849	2.51
	50	22.12	0.9286	11.62	23.75	0.9509	6.72	25.18	0.9622	5.63	26.72	0.9732	4.69	26.36	0.9710	4.89	26.79	0.9735	4.66	26.82	0.9737	4.49
	70	20.99	0.9059	14.07	21.66	0.9234	10.25	23.29	0.9436	8.35	24.45	0.9569	7.27	24.21	0.9542	7.63	24.55	0.9578	7.20	24.61	0.9583	7.00
	90	17.50	0.7814	23.17	18.71	0.8532	16.87	21.75	0.9200	12.02	21.46	0.9165	11.97	22.13	0.9268	11.65	22.07	0.9270	11.30	22.28	0.9299	10.98
Boat	10	26.71	0.9637	6.66	32.92	0.9857	1.67	38.98	0.9917	0.56	37.84	0.9915	0.61	38.46	0.9918	0.60	39.35	0.9922	0.56	39.66	0.9923	0.54
	30	25.65	0.9552	7.38	29.42	0.9764	2.93	32.97	0.9857	1.89	33.17	0.9871	1.85	33.13	0.9870	1.88	33.69	0.9878	1.81	34.21	0.9886	1.72
	50	23.64	0.9298	9.30	26.62	0.9613	4.86	28.65	0.9722	3.85	30.21	0.9802	3.32	29.97	0.9794	3.44	30.69	0.9818	3.23	31.17	0.9831	3.08
	70	21.58	0.8869	12.10	23.81	0.9323	7.77	26.33	0.9572	5.89	27.55	0.9680	5.26	27.06	0.9647	5.62	27.93	0.9705	5.12	28.41	0.9730	4.91
	90	18.42	0.7532	18.92	19.95	0.8456	13.75	22.70	0.9090	10.11	23.42	0.9243	9.32	23.57	0.9260	9.59	24.33	0.9384	8.63	24.71	0.9427	8.39
Bridge	10	23.76	0.9510	10.66	29.01	0.9771	2.97	33.98	0.9865	1.04	32.25	0.9846	1.33	33.32	0.9860	1.18	35.00	0.9878	0.96	35.22	0.9879	0.93
	30	22.92	0.9417	11.62	26.38	0.9671	4.73	28.77	0.9767	3.25	28.70	0.9771	3.36	29.06	0.9784	3.30	29.90	0.9809	3.03	30.28	0.9819	2.89
	50	21.41	0.9168	14.18	23.88	0.9488	7.62	25.25	0.9583	6.45	26.10	0.9654	5.81	26.41	0.9675	5.79	27.13	0.9716	5.30	27.42	0.9731	5.14
	70	19.72	0.8741	17.89	21.35	0.9154	11.98	23.12	0.9373	9.66	24.02	0.9489	8.82	23.99	0.9486	9.18	24.59	0.9553	8.33	24.88	0.9575	8.13
	90	17.14	0.7564	25.48	17.93	0.8240	20.32	20.40	0.8868	15.48	20.71	0.8976	14.73	21.14	0.9050	14.77	21.57	0.9158	13.62	21.80	0.9191	13.41
House	10	26.51	0.9697	4.77	34.13	0.9949	1.14	37.95	0.9979	0.42	28.94	0.9829	0.96	38.41	0.9981	0.48	40.19	0.9988	0.41	42.04	0.9993	0.36
	30	25.51	0.9623	5.51	29.83	0.9863	2.19	32.46	0.9924	1.44	28.47	0.9808	1.82	32.99	0.9933	1.51	34.36	0.9951	1.37	36.11	0.9968	1.22
	50	23.91	0.9445	6.84	26.96	0.9735	3.71	28.72	0.9819	2.85	27.52	0.9760	2.91	29.86	0.9860	2.77	31.12	0.9895	2.46	32.55	0.9925	2.25
	70	21.21	0.8919	9.98	24.04	0.9479	6.04	26.45	0.9693	4.46	26.34	0.9681	4.34	27.13	0.9733	4.54	28.20	0.9793	3.96	29.43	0.9844	3.71
	90	18.06	0.7644	16.42	19.50	0.8524	11.68	22.76	0.9263	7.95	22.95	0.9298	7.68	23.65	0.9386	7.85	24.39	0.9493	6.95	25.05	0.9559	6.82

Table 5.3: Detailed comparison of the state-of-the-art methods and UWMF in terms of PSNR (in dB), SSIM and MAE on additional images.

		AMF			MDBUTMF			IBDND			CMF			IBINR			UWMF		
Image	Noise	PSNR	SSIM	MAE	PSNR	SSIM	MAE	PSNR	SSIM	MAE	PSNR	SSIM	MAE	PSNR	SSIM	MAE	PSNR	SSIM	MAE
Chest	10	40.31	0.9985	0.60	28.06	0.9936	2.73	35.28	0.9984	1.57	34.55	0.9985	1.86	43.26	0.9998	0.24	44.09	0.9998	0.22
	30	36.04	0.9979	1.12	21.26	0.9647	8.34	33.24	0.9976	2.22	32.24	0.9974	2.60	28.48	0.9926	1.02	28.54	0.9926	0.98
	50	32.93	0.9968	1.92	18.53	0.9305	13.91	31.32	0.9965	2.98	30.75	0.9963	3.37	35.91	0.9978	1.35	36.20	0.9980	1.25
	70	29.39	0.9955	3.23	15.80	0.8639	19.73	29.46	0.9947	4.03	28.39	0.9935	4.74	29.73	0.9941	2.31	30.17	0.9945	2.11
	90	21.98	0.9716	7.44	13.93	0.7793	26.12	25.40	0.9866	6.82	24.32	0.9827	8.21	17.25	0.9134	8.39	17.30	0.9144	8.03
Galaxy	10	42.61	0.9992	0.55	49.67	0.9997	0.14	49.45	0.9997	0.16	49.39	0.9997	0.17	50.85	0.9997	0.14	51.42	0.9998	0.13
	30	39.53	0.9982	0.88	43.37	0.9993	0.48	44.24	0.9994	0.50	44.01	0.9993	0.53	43.86	0.9993	0.46	44.13	0.9993	0.45
	50	36.71	0.9965	1.46	39.47	0.9980	1.10	40.91	0.9986	0.92	40.85	0.9987	0.98	41.71	0.9988	0.87	42.77	0.9991	0.79
	70	33.44	0.9916	2.50	36.69	0.9961	1.74	38.28	0.9972	1.49	37.88	0.9971	1.62	38.71	0.9976	1.43	39.76	0.9981	1.31
	90	27.23	0.9664	5.26	32.01	0.9885	3.56	33.59	0.9922	2.82	33.50	0.9919	3.06	34.39	0.9934	2.64	35.16	0.9944	2.48
Head	10	26.29	0.9910	2.36	20.81	0.9662	10.00	11.12	0.6642	48.61	16.46	0.8973	26.36	24.71	0.9871	2.38	25.31	0.9888	2.08
	30	24.30	0.9861	3.54	15.90	0.8859	25.59	10.39	0.6044	53.43	10.96	0.6403	53.45	21.24	0.9719	3.80	21.43	0.9731	3.51
	50	22.30	0.9778	5.21	13.06	0.7670	42.68	9.70	0.5402	58.47	9.14	0.4704	66.53	19.30	0.9534	8.09	19.90	0.9609	7.04
	70	19.90	0.9595	8.04	10.72	0.5951	58.06	8.91	0.4608	64.61	8.17	0.3512	75.74	17.32	0.9254	11.61	17.82	0.9333	10.30
	90	14.96	0.8761	17.45	8.83	0.3754	75.17	7.72	0.3138	77.17	7.61	0.2543	83.40	11.18	0.7086	29.21	11.22	0.7138	28.29
Kidney	10	42.05	0.9989	0.52	38.68	0.9981	0.40	41.49	0.9990	0.35	40.55	0.9987	0.40	45.38	0.9995	0.21	46.42	0.9997	0.18
	30	36.79	0.9970	1.30	32.15	0.9910	1.27	37.37	0.9973	0.86	37.62	0.9975	0.88	37.77	0.9976	0.63	38.31	0.9978	0.51
	50	32.96	0.9925	2.48	29.90	0.9846	2.53	34.11	0.9944	1.59	35.01	0.9953	1.59	36.98	0.9971	1.37	38.23	0.9978	1.09
	70	28.93	0.9820	4.47	27.09	0.9702	3.96	32.25	0.9910	2.59	32.06	0.9906	2.84	33.48	0.9933	2.41	35.30	0.9956	1.94
	90	22.51	0.9205	9.88	24.43	0.9442	7.54	27.73	0.9747	5.39	27.34	0.9717	5.99	27.99	0.9760	5.02	28.78	0.9799	4.45
Moon	10	34.58	0.9933	1.44	40.27	0.9982	0.56	39.76	0.9981	0.58	40.17	0.9982	0.57	41.49	0.9987	0.49	42.19	0.9989	0.45
	30	30.13	0.9830	3.05	34.05	0.9929	1.91	33.86	0.9924	1.84	34.60	0.9937	1.83	35.55	0.9947	1.63	36.15	0.9954	1.52
	50	26.79	0.9650	5.42	28.86	0.9765	4.45	29.94	0.9814	3.54	30.91	0.9852	3.51	31.66	0.9872	3.26	32.53	0.9895	2.98
	70	23.48	0.9258	9.02	26.37	0.9581	6.84	27.31	0.9657	5.79	27.33	0.9657	6.16	28.51	0.9739	5.41	29.22	0.9778	5.05
	90	19.00	0.7939	16.52	21.87	0.8770	12.30	22.86	0.9045	10.85	22.98	0.9033	11.22	24.04	0.9265	9.89	24.48	0.9325	9.48
Skull	10	41.22	0.9990	0.52	44.37	0.9995	0.21	44.33	0.9995	0.23	44.91	0.9995	0.22	46.19	0.9997	0.19	47.25	0.9997	0.17
	30	36.01	0.9969	1.16	38.52	0.9982	0.71	38.73	0.9982	0.70	40.16	0.9987	0.67	40.65	0.9988	0.59	41.64	0.9991	0.51
	50	32.30	0.9928	2.14	34.09	0.9949	1.76	35.51	0.9963	1.32	36.44	0.9970	1.32	37.20	0.9975	1.25	38.42	0.9980	1.09
	70	28.69	0.9830	3.72	31.18	0.9902	2.80	33.07	0.9936	2.21	32.61	0.9929	2.47	33.82	0.9946	2.15	34.61	0.9954	1.95
	90	23.68	0.9464	7.43	26.66	0.9717	5.58	28.36	0.9812	4.42	27.92	0.9789	4.93	29.06	0.9839	4.19	29.37	0.9849	4.05

Table 5.4: Detailed comparison of the state-of-the-art methods and UWMF in terms of PSNR (in dB), SSIM and MAE on additional images.

		AMF			MDBUTMF			IBDND			CMF			IBINR			UWMF		
Image	Noise	PSNR	SSIM	MAE	PSNR	SSIM	MAE	PSNR	SSIM	MAE	PSNR	SSIM	MAE	PSNR	SSIM	MAE	PSNR	SSIM	MAE
Breast	10	42.08	0.9991	0.50	47.81	0.9998	0.17	48.47	0.9998	0.16	47.94	0.9998	0.17	48.82	0.9998	0.15	49.09	0.9998	0.14
	30	39.29	0.9999	0.87	42.22	0.9990	0.56	43.20	0.9992	0.51	42.82	0.9992	0.54	42.76	0.9991	0.49	42.77	0.9992	0.47
	50	36.61	0.9984	1.46	38.46	0.9978	1.19	40.21	0.9986	0.94	39.82	0.9984	0.99	40.32	0.9985	0.93	40.68	0.9986	0.87
	70	33.81	0.9958	2.34	36.06	0.9962	1.79	37.64	0.9973	1.50	37.14	0.9969	1.63	37.78	0.9975	1.48	38.12	0.9976	1.41
	90	28.26	0.9794	4.25	32.99	0.9923	2.93	33.86	0.9937	2.67	33.93	0.9938	2.72	34.42	0.9944	2.52	34.80	0.9949	2.41
Chest2	10	41.80	0.9994	0.70	48.40	0.9998	0.17	48.75	0.9998	0.20	48.18	0.9998	0.22	48.16	0.9998	0.22	47.52	0.9998	0.23
	30	40.28	0.9992	0.90	42.83	0.9995	0.54	43.76	0.9995	0.62	44.16	0.9996	0.59	42.19	0.9994	0.67	41.80	0.9993	0.70
	50	38.28	0.9987	1.32	40.73	0.9991	0.91	41.22	0.9992	1.06	41.56	0.9992	1.01	40.91	0.9992	1.09	40.84	0.9991	1.11
	70	35.57	0.9974	2.05	38.01	0.9983	1.44	39.12	0.9987	1.57	39.44	0.9988	1.52	38.95	0.9987	1.59	39.26	0.9987	1.56
	90	29.03	0.9867	4.03	34.80	0.9965	2.42	35.85	0.9973	2.41	36.35	0.9976	2.41	36.52	0.9976	2.29	37.28	0.9980	2.18
Circuit	10	40.81	0.9985	0.57	42.09	0.9989	0.30	43.14	0.9992	0.33	44.74	0.9994	0.29	46.10	0.9995	0.24	46.78	0.9997	0.22
	30	35.58	0.9945	1.31	35.55	0.9951	0.98	38.67	0.9974	0.89	39.50	0.9980	0.87	39.30	0.9979	0.75	39.70	0.9980	0.71
	50	32.07	0.9892	2.41	33.78	0.9927	1.91	34.60	0.9938	1.68	36.26	0.9956	1.59	37.25	0.9965	1.42	38.57	0.9973	1.25
	70	28.50	0.9769	4.21	30.38	0.9840	3.10	32.22	0.9892	2.65	33.15	0.9912	2.69	34.07	0.9929	2.36	35.52	0.9947	2.10
	90	22.66	0.9123	8.96	26.89	0.9640	5.92	27.46	0.9686	5.13	28.21	0.9730	5.29	29.33	0.9792	4.56	30.41	0.9835	4.15
Satellite	10	23.23	0.9795	5.33	30.08	0.9958	1.37	27.94	0.9931	2.02	30.24	0.9959	1.41	31.87	0.9972	1.19	32.13	0.9973	1.15
	30	22.75	0.9773	6.03	24.91	0.9861	4.23	25.43	0.9876	4.38	25.52	0.9879	4.19	26.59	0.9905	3.67	26.28	0.9898	3.80
	50	21.47	0.9694	8.36	23.78	0.9818	6.53	23.34	0.9796	7.16	23.29	0.9796	7.06	24.36	0.9840	6.21	24.68	0.9851	6.06
	70	19.96	0.9571	11.96	21.75	0.9709	9.72	21.61	0.9696	10.20	22.05	0.9726	9.92	22.27	0.9741	9.28	22.64	0.9762	9.08
	90	17.88	0.9311	17.79	20.29	0.9593	13.24	19.36	0.9494	14.87	20.66	0.9620	13.60	20.25	0.9588	13.49	20.57	0.9615	13.31
Spine	10	40.07	0.9986	0.60	35.32	0.9975	0.65	35.45	0.9975	0.75	35.31	0.9974	0.80	39.71	0.9991	0.35	40.65	0.9993	0.30
	30	35.65	0.9972	1.16	29.89	0.9906	1.66	33.05	0.9958	1.31	32.83	0.9955	1.39	35.12	0.9975	0.76	35.54	0.9977	0.68
	50	32.24	0.9945	2.00	27.69	0.9843	2.89	31.03	0.9925	1.99	31.05	0.9925	2.07	33.58	0.9964	1.56	33.67	0.9965	1.41
	70	28.77	0.9883	3.35	24.90	0.9714	4.28	29.56	0.9895	2.80	28.90	0.9879	3.15	30.91	0.9932	2.44	31.59	0.9943	2.21
	90	23.19	0.9592	6.64	23.02	0.9551	6.37	26.17	0.9776	4.79	25.55	0.9740	5.30	26.02	0.9769	4.41	26.40	0.9788	4.11
Woman	10	37.95	0.9981	1.28	43.65	0.9994	0.35	41.86	0.9992	0.41	43.43	0.9994	0.37	43.94	0.9993	0.36	44.13	0.9994	0.37
	30	35.88	0.9958	1.66	38.62	0.9981	1.08	37.13	0.9975	1.14	38.98	0.9983	1.07	38.74	0.9982	1.10	38.64	0.9981	1.13
	50	33.45	0.9939	2.46	35.45	0.9964	1.83	35.03	0.9960	1.91	36.24	0.9969	1.84	36.43	0.9970	1.85	36.65	0.9970	1.85
	70	30.70	0.9895	3.67	33.18	0.9940	2.73	33.29	0.9940	2.80	33.92	0.9948	2.70	34.18	0.9951	2.74	34.51	0.9954	2.70
	90	25.90	0.9707	6.06	29.72	0.9871	4.13	30.35	0.9888	4.28	30.78	0.9897	4.00	31.22	0.9907	4.00	31.66	0.9915	3.85

The complexity of algorithms in O-notation can be found in Table 5.5. In addition, the speed (CPU time) of the methods used in the experiments is presented in Table 5.6. The results in this table is obtained using relatively bigger window sizes (inner and outer) for BDND.

Table 5.5: Computational Complexity in O-notation.

Method	Complexity
AMF [23]	$O(MNW^2 logW)$
MDBUTMF [8]	$O(MNW^2 logW)$
IBDND [16]	N/A
CMF [42]	N/A
IBINR [17]	$O(MNW^2)$
UWMF	$O(MNW^2)$
M and N are image dimensions and W is filtering window size.	

5.4 Analysis of Parameters

In the third experiment, we have analyzed the effect of the parameters of the proposed method in order to find the optimal values. The proposed method has three parameters. These are k, p and window size (will be referred as *wsize* hereafter). k controls the weight reduction over the distance. Its contribution is inversely proportional to distance to the center of filtering window that is shown in (4.6). *wsize* is the size of square neighborhood in which the proposed method obtains necessary information. Finally, p is the parameter of Minkowski Distance defined in (4.7). The value of p also affects the weights of pixels contributing to weighted mean. Figure 5.7 illustrates the effect of p as its value changes. According to simulation results, restoration quality is the highest when $p = 1$ (Manhattan Distance) and $p = 2$ (Euclidean Distance). Therefore, for other parameters, only these two values of p are tested.

The effect of different k values under the effect of aforementioned distance metrics — Euclidean and Manhattan Distances — can be seen in Figure 5.8. Lower values of k do not diminish weights of pixels sharp enough as distance increases. Furthermore,

Table 5.6: Comparison of the state-of-the-art methods and UWMF in terms of CPU time (in milliseconds)

Image	Noise	AMF	MDBUTMF	IBDND	CMF	IBINR	UWMF
Lena	10	51	11	6159	31	8	18
	30	56	21	5839	48	35	46
	50	81	81	5447	79	61	364
	70	168	100	5198	155	163	803
	90	1131	304	4108	453	576	1787
Checkerboard	10	5411	3	N/A	N/A	3	38
	30	5611	5	N/A	N/A	13	32
	50	5579	18	N/A	N/A	18	137
	70	5235	22	N/A	N/A	44	223
	90	4066	72	N/A	N/A	146	453
Peppers	10	52	11	6139	31	8	17
	30	56	21	5783	48	34	44
	50	82	81	5370	78	61	361
	70	168	100	4780	155	163	812
	90	1190	304	4145	474	574	1786
Baboon	10	52	11	6177	31	8	17
	30	58	23	5794	50	36	44
	50	82	82	5453	82	64	360
	70	189	105	4746	157	168	803
	90	1268	314	3984	464	579	1791
Barbara	10	51	11	5635	54	13	17
	30	58	21	5559	49	35	44
	50	84	83	5258	80	62	360
	70	172	102	4767	161	167	799
	90	1230	311	4058	464	580	1776
Boat	10	51	10	5683	31	8	17
	30	56	21	5433	48	35	44
	50	81	81	5180	78	60	359
	70	167	100	4768	157	163	798
	90	1220	306	4077	474	581	1806
Bridge	10	330	10	5043	35	8	18
	30	313	21	5067	53	36	46
	50	280	84	4950	86	65	369
	70	364	104	4582	167	168	822
	90	1370	311	4015	487	634	1788
House	10	12	2	1486	8	2	4
	30	14	5	1395	12	9	11
	50	21	20	1299	19	15	90
	70	43	26	1154	39	41	202
	90	285	74	993	112	138	446

high values of k cause a dramatic reduction. This situation causes a peak in restoration quality as k increases. It is crucial to use optimal values for k in order to achieve the best noise removal performance. Restoration results in Table 5.2 are obtained with $k = 4$. The optimal value of k is in range of $[4\,6]$, however, as long as k is in this range, the difference is negligible, therefore, a fine-tuning is not necessary.

Unlike k, *wsize* does not affect restoration quality as long as it is large enough to contain uncorrupted pixels. If a very large *wsize* is selected, the weights of very distant pixels are mitigated by k. This is evident in Figure 5.9. However, larger *wsize* will decrease

the performance in terms of CPU time. Recommended values for *wsize* are presented in Table 5.7.

Table 5.7: Recommended window sizes for the proposed method.

Noise Density (p)	Window Size
$p < 20$	3×3
$20 \leq p < 50$	5×5
$50 \leq p < 70$	7×7
$70 \leq p < 85$	9×9
$85 \leq p < 90$	11×11
$p \geq 90$	13×13

5.5 Discussion on Experimental Results

Spatial bias is an impediment to proper estimation of the original intensity value. Our proposed method addresses this problem and provides superior restoration performance. This has been demonstrated in terms of objective measurements in Figures 5.2 and 5.3 as well as in Table 5.2. Furthermore, UWMF decreases the disturbance in the edges and gradient-like regions, resulting with the least amount of jagged edges and blotches between comparison methods, effectively increasing the suitability of images restored by UWMF for further processing or computer vision applications. Figures 5.4 and 5.5 demonstrated the effectiveness of UWMF in these terms.

UWMF does not require parameter tuning based on the image. In addition, the convolution process of the proposed method is suitable for parallel implementation using various computing architectures. This property makes it suitable for real-time applications such as surveillance systems.

5.6 Preliminary Studies on Random-Valued Impulse Noise

The model defined in (3.3) is used for RVIN. In this model, the corruption can take a value on both sides of dynamic range in a span defined with m. For example, if m is 5, the corruption can take any value in [0 5] and [250 255]. All intensity values in these spans have equal probability of occurrence. The noise detection methods are presented

in Table 5.8. Checkerboard image is omitted. Similar to detection results in FVIN on Table 5.1, all methods have failed to detect noises on this image. NND and CMF have failed to detect RVIN. Although the total detection error for these two methods is between 10% and 30% in lower noise densities, the number of pixels that are falsely detected or misclassified as noise is close to total number of corrupted pixels. In other words, these methods have failed to detect majority of the corrupted pixels. BDND provides robust detection performance up to 90% noise density. The detection values in Table 5.8 are obtained using suggested window sizes. However, we have observed that using relatively bigger window sizes than suggested improves the detection performance dramatically. The efficiency in terms of CPU time decreases as window sizes increases.

Table 5.8: Comparison of RVIN detection performance of NND, CMF and BDND

		NND			CMF			BDND		
	Noise	MC (R_{MC})	FA (R_{FA})	Total (R_{TOTAL})	MC (R_{MC})	FA (R_{FA})	Total (R_{TOTAL})	MC (R_{MC})	FA (R_{FA})	Total (R_{TOTAL})
Lena	10	0	25558	25558(9.75%)	51	13218	13269(5.06%)	166	176	342(0.13%)
	30	0	76617	76617(29.23%)	5	66679	66684(25.44%)	138	326	464(0.18%)
	50	0	127739	127739(48.73%)	4	124164	124168(47.37%)	275	220	496(0.19%)
	70	0	178829	178829(68.22%)	30	177799	177829(67.84%)	334	2130	2463(0.94%)
	90	0	230014	230014(87.74%)	88	229354	229441(87.52%)	586	32953	33540(12.79%)
Peppers	10	1	25551	25552(9.75%)	56	13397	13453(5.13%)	1767	1285	3052(1.16%)
	30	1	76758	76759(29.28%)	8	66437	66445(25.35%)	1693	2995	4688(1.79%)
	50	1	127814	127815(48.76%)	6	123486	123491(47.11%)	1797	4495	6292(2.40%)
	70	0	178886	178886(68.24%)	32	177510	177542(67.73%)	1554	7604	9157(3.49%)
	90	0	229970	229970(87.73%)	76	229208	229284(87.46%)	1220	37240	38460(14.67%)
Baboon	10	23	25572	25595(9.76%)	41	18149	18190(6.94%)	627	320	947(0.36%)
	30	17	76810	76827(29.31%)	21	71297	71318(27.21%)	623	329	952(0.36%)
	50	14	128097	128111(48.87%)	18	126218	126236(48.16%)	669	220	889(0.34%)
	70	9	178783	178792(68.20%)	47	178220	178267(68.00%)	512	2130	2642(1.01%)
	90	3	229931	229934(87.71%)	90	229361	229451(87.53%)	506	32780	33286(12.70%)
Barbara	10	0	25620	25620(9.77%)	14	15739	15754(6.01%)	474	643	1117(0.43%)
	30	0	76679	76679(29.25%)	2	68647	68649(26.19%)	753	1003	1756(0.67%)
	50	0	127886	127886(48.78%)	4	124714	124718(47.58%)	897	898	1794(0.68%)
	70	0	178823	178823(68.22%)	34	177765	177799(67.82%)	812	2663	3476(1.33%)
	90	0	230007	230007(87.74%)	83	229335	229417(87.52%)	858	33103	33961(12.96%)
Boat	10	7	25497	25504(9.73%)	23	14034	14057(5.36%)	1137	744	1881(0.72%)
	30	6	76586	76593(29.22%)	9	68056	68064(25.96%)	919	1256	2175(0.83%)
	50	5	127585	127589(48.67%)	10	124821	124830(47.62%)	1153	1658	2810(1.07%)
	70	3	178716	178719(68.18%)	32	178004	178036(67.92%)	961	3773	4734(1.81%)
	90	1	230004	230005(87.74%)	80	229425	229505(87.55%)	902	33986	34888(13.31%)
Bridge	10	1652	25571	27223(10.38%)	1756	16541	18298(6.98%)	10745	420	11165(4.26%)
	30	1289	76427	77716(29.65%)	1303	68876	70179(26.77%)	5445	1567	7012(2.67%)
	50	930	127780	128710(49.10%)	934	124741	125675(47.94%)	5745	1930	7675(2.93%)
	70	564	178746	179310(68.40%)	600	177744	178344(68.03%)	3924	4122	8046(3.07%)
	90	221	229840	230061(87.76%)	299	229163	229462(87.53%)	1801	34412	36213(13.81%)
House	10	0	6431	6431(9.81%)	6	3329	3335(5.09%)	544	24	568(0.87%)
	30	0	19161	19161(29.24%)	0	16694	16694(25.47%)	390	77	467(0.71%)
	50	0	31977	31977(48.79%)	1	31017	31018(47.33%)	522	93	614(0.94%)
	70	0	44698	44698(68.20%)	9	44378	44386(67.73%)	376	630	1006(1.54%)
	90	0	57464	57464(87.68%)	20	57285	57305(87.44%)	202	8300	8503(12.97%)

NND and CMF are not suitable for RVIN. For this reason, we have incorporated BDND into our proposed method (UWMF). The restoration performances are presented in Table 5.9. The restoration results are close to the performance on FVIN except for images House and Peppers. This is due to poor detection performance.

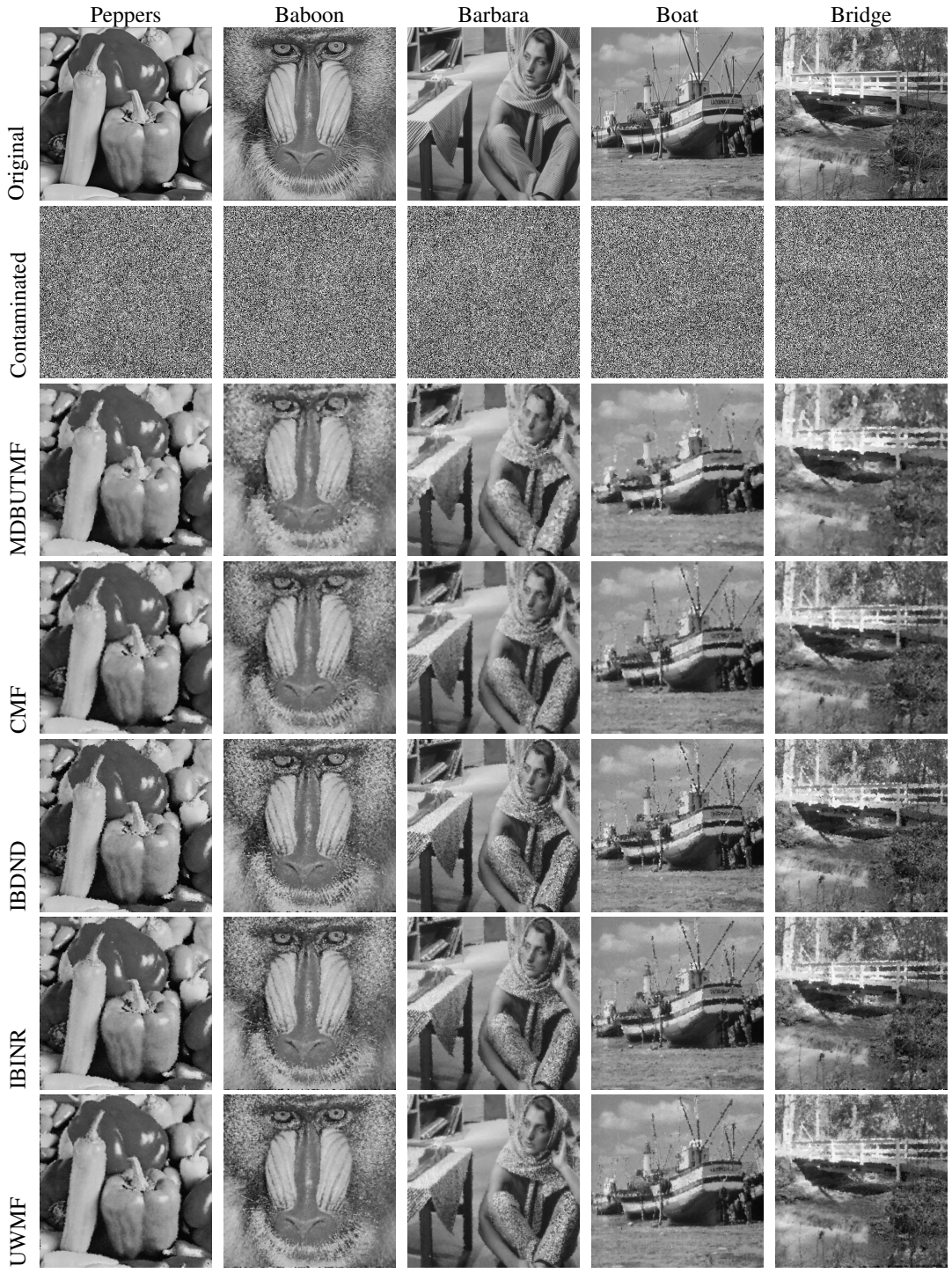


Figure 5.6: Visual results of five images restored with the state-of-the-art methods and UWMF.

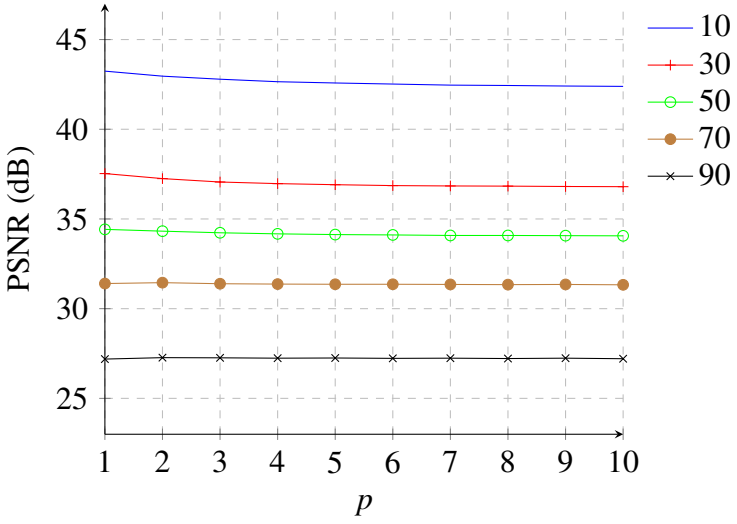


Figure 5.7: Restoration results of UWMF with different values of p.

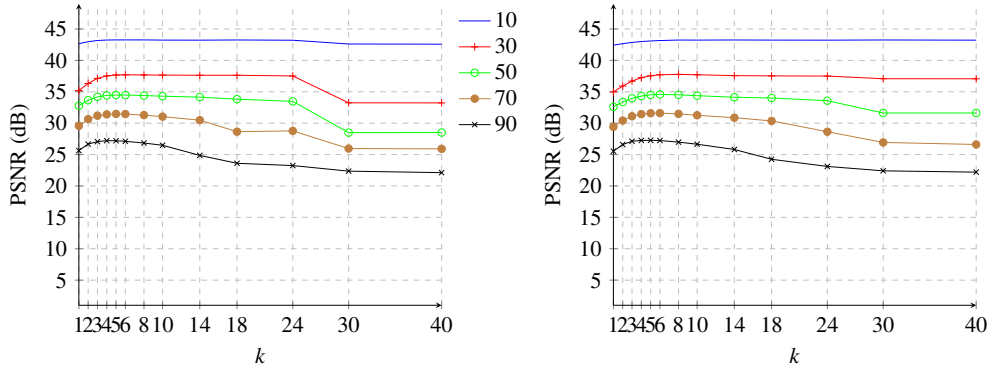


Figure 5.8: Restoration results (PSNR in dB) of UWMF with different k values on Lena image contaminated with different noise densities. (a) UWMF with Euclidean Distance. (b) UWMF with Manhattan Distance.

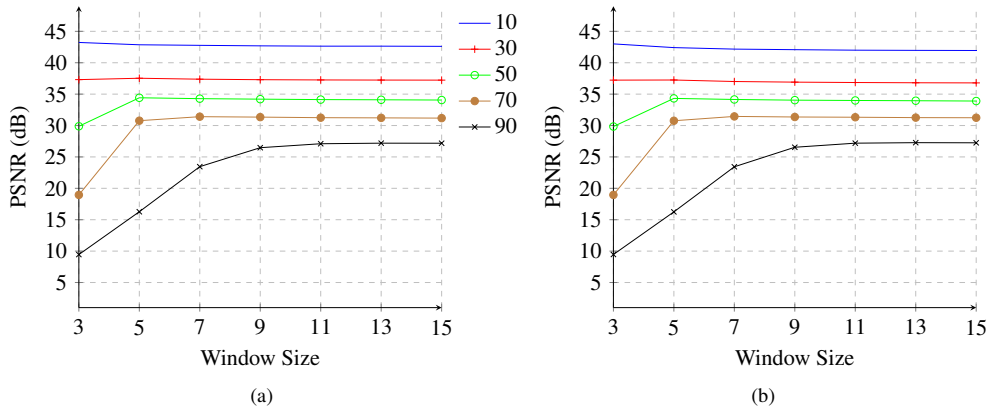


Figure 5.9: Restoration results (PSNR in dB) of UWMF with various window sizes on Lena image contaminated with different noise densities. (a) UWMF with Euclidean Distance. (b) UWMF with Manhattan Distance.

Table 5.9: Comparison of the performance of UWMF on images contaminated by RVIN and FVIN.

		Lena		Peppers		Baboon		Barbara		Boat		Bridge		House	
Metric	Noise	RVIN	FVIN	RVIN	FVIN	RVIN	FVIN	RVIN	FVIN	RVIN	FVIN	RVIN	FVIN	RVIN	FVIN
PSNR	10	42.40	43.25	31.83	41.45	30.05	32.76	35.09	35.18	37.66	39.66	33.55	35.22	29.81	42.04
	30	37.29	37.53	29.94	36.11	26.24	27.62	29.54	29.73	33.28	34.21	28.56	30.28	28.51	36.11
	50	34.27	34.42	28.55	32.95	24.20	24.72	26.93	26.82	30.62	31.17	25.68	27.42	27.54	32.55
	70	31.31	31.41	25.95	30.16	22.18	22.41	24.67	24.61	28.02	28.41	23.01	24.88	26.46	29.43
	90	26.89	27.19	22.27	26.27	19.75	19.83	22.25	22.28	24.45	24.71	20.34	21.80	23.33	25.05
SSIM	10	0.9993	0.9993	0.9940	0.9991	0.9831	0.9908	0.9964	0.9931	0.9975	0.9923	0.9952	0.9879	0.9860	0.9993
	30	0.9974	0.9976	0.9908	0.9976	0.9580	0.9693	0.9876	0.9849	0.9930	0.9886	0.9851	0.9819	0.9811	0.9968
	50	0.9949	0.9952	0.9871	0.9952	0.9309	0.9389	0.9774	0.9737	0.9866	0.9831	0.9699	0.9731	0.9756	0.9925
	70	0.9899	0.9903	0.9758	0.9910	0.8874	0.8937	0.9619	0.9583	0.9758	0.9730	0.9407	0.9575	0.9684	0.9844
	90	0.9726	0.9740	0.9424	0.9779	0.7983	0.8027	0.9324	0.9299	0.9422	0.9427	0.8897	0.9191	0.9359	0.9559
MAE	10	0.36	0.36	0.76	0.45	1.48	1.27	0.77	0.77	0.60	0.54	1.19	0.93	0.81	0.36
	30	1.16	1.16	1.83	1.40	4.29	4.01	2.47	2.51	1.79	1.72	3.24	2.89	1.72	1.22
	50	2.11	2.09	3.07	2.46	7.32	7.15	4.50	4.49	3.20	3.08	6.03	5.14	2.80	2.25
	70	3.39	3.36	4.93	3.79	11.07	11.02	7.00	7.00	5.03	4.91	9.71	8.13	4.17	3.71
	90	6.04	5.93	8.40	6.32	17.03	16.96	10.99	10.98	8.55	8.39	15.05	13.4	7.29	6.82

Chapter 6

CONCLUSION

This thesis investigated nonlinear filters for impulse noise of type Fixed-Valued Impulse Noise (FVIN). We have analyzed various state-of-the-art methods for impulse noise detection and removal, namely Adaptive Median Filter (AMF), Modified Decision Based Unsymmetric Trimmed Median Filter (MDBUTMF), Naïve Noise Detection (NND), Boundary Discriminative Noise Detection (BDND), Improved Boundary Discriminative Noise Detection Filtering Algorithm (IBDND), Cloud Model Filter (CMF) and Interpolation-based Impulse Noise Removal (IBINR). Furthermore, we have provided an empirical comparison in both objective and subjective assessments. Finally, we have found that the distribution of noise in the filtering window can be exploited to counter a problem what we call *spatial bias*.

The intensity value of corrupted pixel can not be used in restoration procedure. Many filters ignore corrupted pixels while estimating the original intensity value. In a filtering window, corrupted pixels are as spatially correlated as uncorrupted pixels. Therefore, using only corruption-free pixels leads to a spatial bias. In a contaminated image, the information of intensity values of corrupted pixels is lost, however, their position information is still present. The natural question is then arisen to ask if information supplied by corrupted pixels can be exploited to make a better estimation.

The spatial bias is characterized by the positional distribution of noise in the filtering window. In the presence of a weight function, the position information can be exploited to recalibrate weights in order to counter the spatial bias. We have utilized this idea and proposed a novel method for impulse noise removal, namely Unbiased Weighted

Mean Filter (UWMF), which eliminates spatial bias and effectively provides a better estimation of the original intensity value.

We have conducted extensive simulations to assess performance of aforementioned state-of-the-art methods in terms of detection accuracy, objective restoration metrics — Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM) and Mean Absolute Error (MAE) — as well as computational complexity in O-notation and CPU time. In general, we have observed that detection methods perform impressively and yield a total detection error rate less than 1%. We have concluded that methods utilizing a weight function which reflects the spatial relationship between pixels provide better restoration quality. Furthermore, we have demonstrated that spatial bias is an inherent problem of impulse noise due to random distribution of corrupted pixels. Experiments show that the elimination of spatial bias mitigates disturbance in the edges and smooth regions, resulting with superior visual clarity and restoration performance due to its unbiased nature. Finally, we have analyzed the parameters of the proposed method and provided optimal values to obtain the highest restoration performance.

In the future, applications of weight elimination to other filters or impulse noise models can be investigated. Additionally, effects of nonlinear weight recalibration function can be explored. Finally, extension of UWMF to color image can be studied.

REFERENCES

[1] Astola, J. & Kuosmanen, P. (1997). *Fundamentals of Nonlinear Digital Filtering*. Electronic Engineering Systems. Taylor & Francis.

[2] Bai, T. & Tan, J. (2015). Automatic detection and removal of high-density impulse noises. *Image Processing, IET*, 9(2):162–172.

[3] Bai, T., Tan, J., Hu, M., & Wang, Y. (2014). A novel algorithm for removal of salt and pepper noise using continued fractions interpolation. *Signal Processing*, 102(0):247 – 255.

[4] Bernstein, R. (1987). Adaptive nonlinear filters for simultaneous removal of different kinds of noise in images. *Circuits and Systems, IEEE Transactions on*, 34(11):1275–1291.

[5] Brownrigg, D. R. K. (1984). The weighted median filter. *Commun. ACM*, 27(8):807–818.

[6] Dong, Y., Chan, R., & Xu, S. (2007). A detection statistic for random-valued impulse noise. *Image Processing, IEEE Transactions on*, 16(4):1112–1120.

[7] Eng, H.-L. & Ma, K.-K. (2001). Noise adaptive soft-switching median filter. *Image Processing, IEEE Transactions on*, 10(2):242–251.

[8] Esakkirajan, S., Veerakumar, T., Subramanyam, A., & PremChand, C. (2011). Removal of high density salt and pepper noise through modified decision based

unsymmetric trimmed median filter. *Signal Processing Letters, IEEE*, 18(5):287–290.

[9] Feng, J., Ding, M., & Zhang, X. (2014). Decision-based adaptive morphological filter for fixed-value impulse noise removal. *Optik - International Journal for Light and Electron Optics*, 125(16):4288 – 4294.

[10] Frieden, B. R. (1976). A new restoring algorithm for the preferential enhancement of edge gradients. *J. Opt. Soc. Am.*, 66(3):280–283.

[11] Gonzalez, R. & Woods, R. (2010). *Digital Image Processing*. Pearson Education, Limited.

[12] Huang, T., Yang, G., & Tang, G. (1979). A fast two-dimensional median filtering algorithm. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 27(1):13–18.

[13] Huang, T. S. (1981). *Two-Dimensional Digital Signal Processing II: Transforms and Median Filters*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[14] Hwang, H. & Haddad, R. (1995). Adaptive median filters: new algorithms and results. *Image Processing, IEEE Transactions on*, 4(4):499–502.

[15] Ibrahim, H., Kong, N., & Ng, T. F. (2008). Simple adaptive median filter for the removal of impulse noise from highly corrupted images. *Consumer Electronics, IEEE Transactions on*, 54(4):1920–1927.

[16] Jafar, I., AlNa'mneh, R., & Darabkh, K. (2013). Efficient improvements on the bdnd filtering algorithm for the removal of high-density impulse noise. *Image*

Processing, IEEE Transactions on, 22(3):1223–1232.

[17] Kalyoncu, C., Toygar, O., & Demirel, H. (2013). Interpolation-based impulse noise removal. *Image Processing, IET*, 7(8):777–785.

[18] Ko, S.-J. & Lee, Y. H. (1991). Center weighted median filters and their applications to image enhancement. *Circuits and Systems, IEEE Transactions on*, 38(9):984–993.

[19] Li, D., Liu, C., & Gan, W. (2009). A new cognitive model: Cloud model. *Int. J. Intell. Syst.*, 24(3):357–375.

[20] Li, Y., Sun, J., & Luo, H. (2014). A neuro-fuzzy network based impulse noise filtering for gray scale images. *Neurocomputing*, 127(0):190 – 199. Advances in Intelligent Systems Selected papers from the 2012 Brazilian Symposium on Neural Networks (SBRN 2012).

[21] Li, Z., Cheng, Y., Tang, K., Xu, Y., & Zhang, D. (2015). A salt & pepper noise filter based on local and global image information. *Neurocomputing*, 159(0):172 – 185.

[22] Liang, S.-F., Lu, S.-M., Chang, J.-Y., & Lin, C.-T. (2008). A novel two-stage impulse noise removal technique based on neural networks and fuzzy decision. *Fuzzy Systems, IEEE Transactions on*, 16(4):863–873.

[23] Lin, H.-M. & Willson, J., A.N. (1988). Median filters with adaptive length. *Circuits and Systems, IEEE Transactions on*, 35(6):675–690.

[24] Meher, S. K. & Singhawat, B. (2014). An improved recursive and adaptive median filter for high density impulse noise. *{AEU} - International Journal of Electronics and Communications*, 68(12):1173 – 1179.

[25] Ng, P.-E. & Ma, K.-K. (2006). A switching median filter with boundary discriminative noise detection for extremely corrupted images. *Image Processing, IEEE Transactions on*, 15(6):1506–1516.

[26] Nodes, T. & Gallagher, J., N.C. (1982). Median filters: Some modifications and their properties. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 30(5):739–746.

[27] Perlman, S., Eisenhandler, S., Lyons, P., & Shumila, M. (1987). Adaptive median filtering for impulse noise elimination in real-time tv signals. *Communications, IEEE Transactions on*, 35(6):646–652.

[28] Perona, P. & Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(7):629–639.

[29] Ramadan, Z. (2014). A new method for impulse noise elimination and edge preservation. *Electrical and Computer Engineering, Canadian Journal of*, 37(1):2–10.

[30] Srinivasan, K. & Ebenezer, D. (2007). A new fast and efficient decision-based algorithm for removal of high-density impulse noises. *Signal Processing Letters, IEEE*, 14(3):189–192.

[31] Tan, X., Zhang, Q., & Zha, D. (2012). A new salt and pepper noise removal filtering algorithm. In *World Automation Congress (WAC), 2012*, pages 129–131.

[32] Toh, K., Ibrahim, H., & Mahyuddin, M. (2008). Salt-and-pepper noise detection and reduction using fuzzy switching median filter. *Consumer Electronics, IEEE Transactions on*, 54(4):1956–1961.

[33] Toh, K. & Isa, N. (2010). Noise adaptive fuzzy switching median filter for salt-and-pepper noise reduction. *Signal Processing Letters, IEEE*, 17(3):281–284.

[34] Toprak, A., Ozerdem, M. S., & Guler, I. (2008). Suppression of impulse noise in mr images using artificial intelligent based neuro-fuzzy adaptive median filter. *Digital Signal Processing*, 18(3):391 – 405.

[35] Tukey, J. (1977). *Exploratory Data Analysis*. Addison-Wesley series in behavioral science. Addison-Wesley Publishing Company.

[36] Veerakumar, T., Esakkirajan, S., & Vennila, I. (2014). Edge preserving adaptive anisotropic diffusion filter approach for the suppression of impulse noise in images. *{AEU} - International Journal of Electronics and Communications*, 68(5):442 – 452.

[37] Yang, R., Yin, L., Gabbouj, M., Astola, J., & Neuvo, T. (1995). Optimal weighted median filtering under structural constraints. *Signal Processing, IEEE Transactions on*, 43(3):591–604.

[38] Zhang, C. & Wang, K. (2015). A switching median–mean filter for removal of high-density impulse noise from digital images. *Optik - International Journal for Light and Electron Optics*, 126(9–10):956 – 961.

[39] Zhang, P. & Li, F. (2014). A new adaptive weighted mean filter for removing salt-and-pepper noise. *Signal Processing Letters, IEEE*, 21(10):1280–1283.

[40] Zhang, X., Yin, Z., & Xiong, Y. (2008). Adaptive switching mean filter using conditional morphological noise detector. *Electronics Letters*, 44(6):406–407.

[41] Zhang, X. & Xiong, Y. (2009). Impulse noise removal using directional difference based noise detector and adaptive weighted mean filter. *Signal Processing Letters, IEEE*, 16(4):295–298.

[42] Zhou, Z. (2012). Cognition and removal of impulse noise with uncertainty. *Image Processing, IEEE Transactions on*, 21(7):3157–3167.

[43] Zuo, Z., Zhang, T., Hu, J., & Zhou, G. (2013). A new method for removing impulse noise based on noise space characteristic. *Optik - International Journal for Light and Electron Optics*, 124(18):3503 – 3509.

APPENDICES

Appendix A: Source Code

Listing A.1: C++ Implementation of UWMF

```
/*****************************************************
** this function calculates inverse Minkowski Distance **
** to the given center of a filtering window          **
** the weights are stored in a flaot vector           **
*****************************************************/
void CalculateWeights(std::vector<float> &weights,
                      const gge::utils::Point2D &center,
                      int wsize, int k, int p)
{
    // the side length of the square window
    int width = wsize * 2 + 1;
    int height = width;
    float weight = 0;
    for(int y = 0; y < height; y++) {
        for(int x = 0; x < width; x++) {
            // make distance zero if its the central pixel
            if(center.x == x && center.y == y) {
                weights[height * y + x] = 0.f;
                continue;
            }
            // calculate inverse distance
            float l = std::abs(center.x - x);
            float r = std::abs(center.y - y);
            l = std::pow(l, p);
            r = std::pow(r, p);
            float distance = std::pow(l + r, 1.f / p);
            weight = 1 / std::pow(distance, k);
            weights[height * y + x] = weight;
        }
    }
}

/*****************************************************
** this function compares two floating-point number   **
** such function is required since the underlying     **
** representation of floating-point numbers is not    **
** precise                                            **
*****************************************************/
bool IsEqual(float first, float second)
{
    // the smallest representable fixed value
```

```
    float epsilon = std::numeric_limits<float>::epsilon();

    // compare the difference rather than equality
    first = std::fabs(first);
    second = std::fabs(second);
    float large = (first > second) ? first : second;
    float right = epsilon * large;
    float left = std::fabs(first - second);
    return left <= right;
}

/*****************************************************
** get cpu-time in terms of ms                      **
*****************************************************/
std::clock_t DiffInMS(std::clock_t tstart, std::clock_t
   tend)
{
    return 1000.f * (tend - tstart) / CLOCKS_PER_SEC;
}

/*****************************************************
** c++ implementation of Unbiased Weighted Mean Filter **
** takes two image containers as parameters         **
** first image is corrupted image, the second one   **
** stores the resotred image, other arguments are   **
** filter parameters                                **
*****************************************************/
std::clock_t UWMF(graphics::basic_Canvas<float> &image,
                  graphics::basic_Canvas<float> &output,
                  int wsize = 1, int p = 1, int k = 4, int
                      offset = 0)
{
    // start clock for CPU time measurement
    std::clock_t tstart = std::clock();

    // some constants
    const int WIDTH = image.GetWidth();
    const int HEIGHT = image.GetHeight();
    const float MAX = 1.f;
    const float MIN = 0.f;

    // initialize variables and calculate required weights
        etc.
    std::vector<float> orgweights(((wsize * 2 + 1) * (
       wsize * 2 + 1)));
    std::vector<float> modifiedweights(((wsize * 2 + 1) *
       (wsize * 2 + 1)));
```

```
CalculateWeights(orgweights, gge::utils::Point(wsize,
    wsize), wsize, k, p);

int startx, starty, endx, endy, weightstart, index;
int mincount, maxcount;
float curweight, modifiedsum, modifiedwsum, orgsum,
    orgwsum;
float pixelval, P, Q, R, S, T, term1, term2, term3;
bool hasnonnoisy;
gge::utils::Point2D Gp; // counter-bias vector

// start the main loop
for(int y = offset; y < HEIGHT - offset; y++) {
    for(int x = offset; x < WIDTH - offset; x++) {

        // reset the weights for each loop
        modifiedweights = orgweights;

        pixelval = image(x, y);

        // noise detection
        if(!IsEqual(pixelval, MIN) && !IsEqual(
            pixelval, MAX)) {
            output(x, y) = image(x, y);
            continue;
        }

        // compute the dimensions of filtering window
        startx = std::max(-wsize, -x);
        starty = std::max(-wsize, -y);
        endx = std::min(wsize, WIDTH - 1 - x);
        endy = std::min(wsize, HEIGHT - 1 - y);
        weightstart = (wsize + starty) * (wsize * 2 +
            1) + wsize + startx;

        // compute spatial bias
        index = weightstart;
        P = Q = R = S = T = 0.f;
        for(int yy = starty; yy <= endy; yy++) {
            for(int xx = startx; xx <= endx; xx++) {
                pixelval = image(x + xx, y + yy);
                if(!IsEqual(pixelval, MIN) && !IsEqual
                    (pixelval, MAX)) {
                    curweight = orgweights[index];
                    P += (xx * xx) * curweight;
                    Q += xx * curweight * yy;
                    R += xx * curweight;
                    S += (yy * yy) * curweight;
```

```
                T += yy * curweight;
            }

            index++;
        }
        index += wsize * 2 + startx - endx;
    }

    R = -R;
    T = -T;

    // a solution for  numerical instabilities
    // !!! this is just a work-around
    term1 = (P * T - Q * R);
    term2 = (S * P - (Q * Q));

    if(term2 == 0.f) {
        term1 += 0.000001f;
        term2 += 0.000001f;
    }

    Gp.y = term1 / term2;

    term3 = (R - Q * Gp.y);


    if(P == 0.f) {
        term3 += 0.000001f;
        P += 0.000001f;
    }

    Gp.x = term3 / P;

    // recalibrate weights using counter-bias
       vector (Gp)
    index = weightstart;
    for(int yy = starty; yy <= endy; yy++) {
        for(int xx = startx; xx <= endx; xx++) {
            pixelval = image(x + xx, y + yy);
            if(!IsEqual(pixelval, MIN) && !IsEqual
               (pixelval, MAX)) {
                curweight = orgweights[index];
                modifiedweights[index] = curweight
                    + ((Gp.x * xx + Gp.y * yy) *
                   curweight);
            }
            index++;
        }
```

```
        index += wsize * 2 + startx - endx;
    }

    // initialize variables for original intesnity
    // value estimation, i.e., weighted mean
    index = weightstart;
    mincount = 0, maxcount = 0;
    modifiedsum = 0.f, modifiedwsum = 0.f;
    orgsum = 0.f, orgwsum = 0.f;
    hasnonnoisy = false;

    // compute weighted mean using recalibrated
       weights
    for(int yy = starty; yy <= endy; yy++) {
        for(int xx = startx; xx <= endx; xx++) {
            pixelval = image(x + xx, y + yy);
            if(IsEqual(pixelval, MIN)) {
                mincount++;
            }
            else if(IsEqual(pixelval, MAX)) {
                maxcount++;
            }
            else {
                hasnonnoisy = true;

                modifiedwsum += modifiedweights[
                   index];
                modifiedsum += modifiedweights[
                   index] * pixelval;

                orgwsum += orgweights[index];
                orgsum += orgweights[index] *
                   pixelval;
            }
            index++;
        }
        index += wsize * 2 + startx - endx;
    }

    // if there aren't any uncorrupted pixels
    // pick either black or wihte depending on
    // occurence count
    if(!hasnonnoisy) {
        if(mincount > maxcount) {
            output(x, y) = 0.f;
        }
        else {
            output(x, y) = 1.f;
```

```
                }
            }
            else {

                // if there is a problem with the
                   estimation
                // use weighted mean computed using the
                   original weight
                // clipping is another option
                if(IsEqual(modifiedwsum, MIN) ||
                   (modifiedsum / modifiedwsum) < MIN ||
                   (modifiedsum / modifiedwsum) > MAX) {
                    output(x, y) = orgsum / orgwsum;
                }
                else {
                    output(x, y) = modifiedsum /
                       modifiedwsum;
                }

            }
        }
    }

    std::clock_t tend = std::clock();
    return DiffInMS(tstart, tend);
}
```

Appendix B: Additional Images

Figure B.1: Chest image. 600×493 in size.

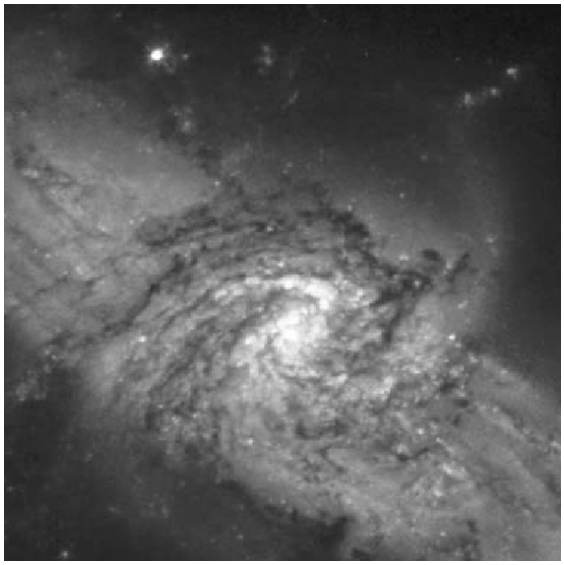


Figure B.2: Galaxy image. 566×598 in size.

Figure B.3: Head image. 512×512 in size.

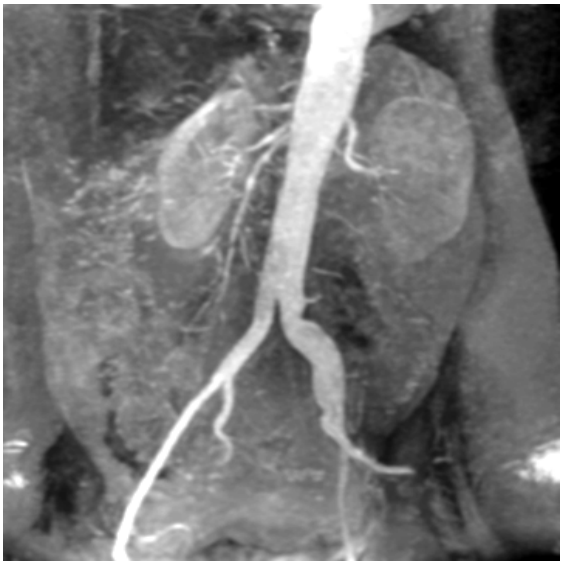


Figure B.4: Kidney image. 360×414 in size.

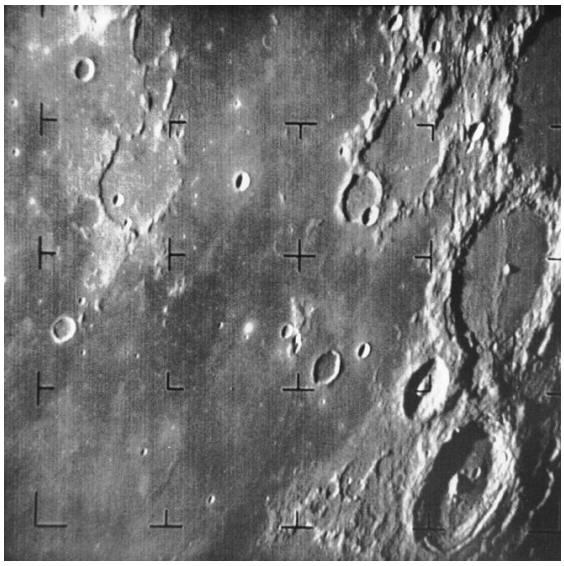


Figure B.5: Moon image. 662×640 in size.

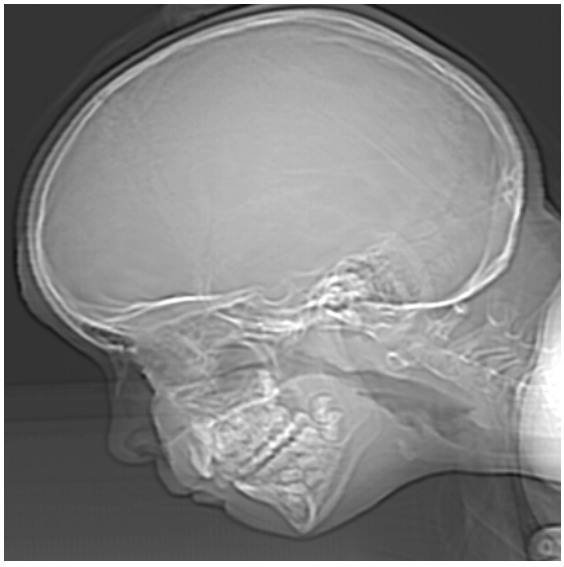


Figure B.6: Skull image. 374×452 in size.

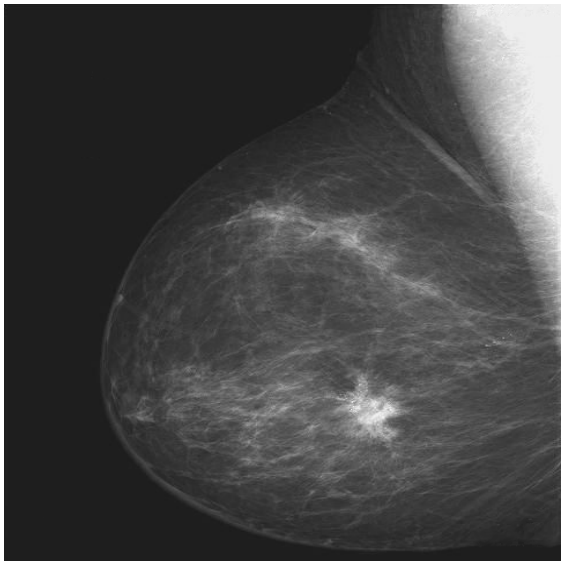


Figure B.7: Breast image. 482×571 in size.

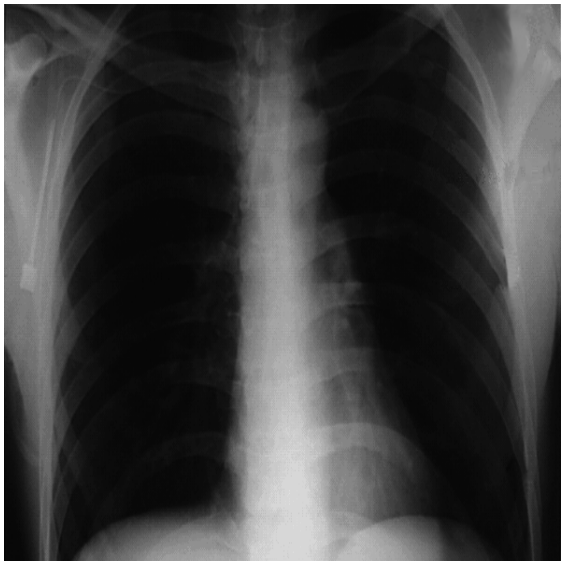


Figure B.8: Chest2 image. 596×416 in size.

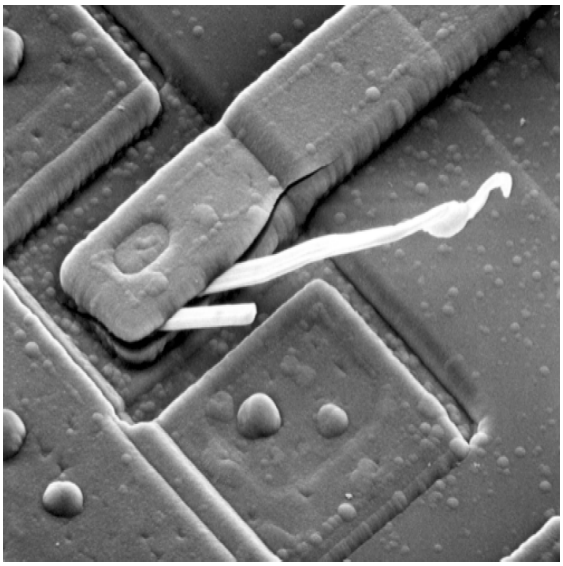


Figure B.9: Circuit image. 906×678 in size.

Figure B.10: Satellite image. 754×808 in size.

Figure B.11: Spine image. 512×512 in size.

Figure B.12: Woman image. 732×785 in size.