An Iterative Approach to Remove High-Density Salt and Pepper Noise from Medical MRI Images using Non-Local Mean

This thesis is submitted in partial fulfilment of the requirement for the degree of Bachelor of science in Computer Science & Engineering

Presented By:

Sahadat Hossen

ID: 181008112

Jafran Bin Zakaria

ID: 181007612

Supervised By:

Golam Moktader Daiyan

Associate Professor

Department of Camputer Science and Engineering

School of Science, Engineering, and Technology

East Delta University

EDU Permanent Campus, Noman Society, East Nasirabad, Chittagong- 4209

Bangladesh

April 2022

The undergraduate thesis titled An Iterative Approach to Remove High-Density Salt and Pepper Noise from Medical MRI Images using Non-Local Mean Recursive Filter submitted by Sahadat Hossen (181008112) and Jafran Bin Zakaria (181007612) has been accepted as satisfactory in fulfillment of the requirement for the degree of Bachelor of Science (B.Sc.) in Computer Science and Engineering (CSE) to be awarded by East Delta University.

Board of Examiners

Dr. Mohammed Nazim Uddin **Chairman**

Associate Dean & Professor

School of Science, Engineering and Technology East Delta University

Sohrab Hossain **Member**

Associate Professor

School of Science, Engineering and Technology East Delta University

Linkon Chowdhury **Member**

Assistant Professor

School of Science, Engineering and Technology East Delta University

Golam Moktader Daiyan **Supervisor**

Associate Professor

School of Science, Engineering and Technology East Delta University

Declaration

I, Sahadat Hossen (181008112) hereby declare that I have produced the work presented in this thesis, during the scheduled period of study. I also declare that I have not taken any material from any source except referred to wherever that amount of plagiarism is within the acceptable range.

Date: 27/04/2022

Sahadat Hossen

ID: 181008112

I, Jafran Bin Zakaria (181007612) hereby declare that I have produced the work presented in this thesis, during the scheduled period of study. I also declare that I have not taken any material from any source except referred to wherever that amount of plagiarism is within the acceptable range.

Date: 27/04/2022

Jafran Bin Zakaria

ID: 181007612

ACKNOWLEDGMENTS

First and foremost, we would like to express our genuine gratefulness to the Almighty for granting us the opportunity and direction we needed to accomplish our goal and be successful so far. Then, we would like to thank our supervisor, Mr. Golam Moktader Daiyan, for his patience, support, and guidance. This thesis would not have been completed if it hadn't been for his help and contributions.

We would additionally like to say our obligation to our respected faculty, seniors, friends, and juniors for their efforts in encouraging and inspiring us throughout the project.

we would like to express our gratitude to our parents and family for their constant efforts and unconditional love and support on every step we take toward realizing our dreams and ambitions.

Table of Contents

Table of Figures

List of tables

ABSTRACT

For digital medical images, noise is a very unpleasant thing nowadays. Digital images are nothing but a stream of electrical signals that convert from an optical image that is later quantization into pixel technology. In the quantization process, disturbances of other signals create noise in the digital image. Different algorithms and methods are used to remove different kinds of noise from digital photos. Digital medical images are often corrupted by a type of noise known as Salt and Pepper noise. Most of the available developed algorithms and noise removing methods are used to remove low-level or low-density noise from a digital image. Different types of filter techniques are used to remove Salt and Pepper noise. But the limitation of those filters is that they cannot remove high-density noise from a digital image. For medical images, those filters perform very worst than natural pictures. We present an algorithm to remove Salt-and-Pepper noise from digital medical images in this thesis work. The algorithm is a mean filter algorithm that works recursively after initially removing most of the noise from the medical digital image. The recursive function does not even work for the natural images because the algorithm removes all noise from the image at the first iteration. Experimental results show that our proposed algorithm performance is better than other Salt-and-Pepper noise removal filters. Up to 95% of the noise level, our algorithm can preserve the visual quality and necessary details.

1 INTRODUCTION & LITERATURE REVIEW

1.1 Introduction

Noises usually corrupt medical MRI images. Various types of noise in digital images are generally generated by electric transmission data, image acquisition, light level, etc. For denoising the picture, there are a variety of methods and algorithms. Some work well or partially depending on the noise pixel number, noise type, and noise level intensity. There are some popular methods and algorithms to denoise images—for instance, Mean Filter, Median Filter, etc. Mean and median work well with low-level noise and low-level intensity noise. With time, many adaptive and new methods and algorithms come for denoising images, where fewer filters work well in high-level noise. And some ways of work are also based on noise type. As mentioned above, noise has a variety of styles. For this paper, we work with the Salt and Pepper noise type. Salt and pepper noise is a noise type where some pixels of an image are Salt meaning White or in grayscale value is 255, and Pepper means black or in grayscale value is 0. In a word, grayscale values 0 and 255 are generated in a digital image as a noise which is called Salt and Pepper noise. In medical MRI images, Salt and Pepper noise is common noise.

1.2 Literature Review

In this paper, we select some improved algorithms, such as

- Adaptive Switching Weight Mean Filter (ASWMF)
- Based on Pixel Density Filter (BPDF)
- Different Applied Median Filter (DAMF)
- Noise Adaptive Fuzzy Switching Median Filter (NAFSMF)

To compare with our proposed methods in medical MRI images. Those improved methods work exceptionally well with high-density noise levels.

Adaptive Switching Weight Mean Filter **(ASWMF)** is an image denoise method. It is an advanced method of the standard mean filter. It considers fixed adaptive window size for each pixel of the corrupted image[1]. If the center pixels are corrupted pixels, it eliminates all corrupted noise pixels of its adaptive window, fixed window size. After that, the process put a low weight on pixels of diagonals and high importance on pixels outside of diagonals of the adaptive window to measure the mean values. The process is the core process of the Adaptive Switching Weight Mean Filter, and it works well in standard digital images, even in high-intensity level noise. But in medical MRI images, ASWMF performs very poorly. Because, in MRI images, images are always present with high segmentation versions.

Noise Adaptive Fuzzy Switching Median Filter **(NAFSMF)** provides a method to remove high-density noise levels. In detecting noise pixels, they use an adaptive window size. Noisy pixels are applied to a median filter, then two other parameters, T1 and T2. In the process, a new pixel value is determined with a fuzzy method[2].

Different Applied Median Filter **(DAMF)** methods are also suitable to remove salt and pepper noise with high levels of density noise. This method works like an improved median filter where the window size is the adaptive window size. It means not fixed window size[3].

Based on Pixel Density Filter **(BPDF)** is also an image denoise method. It generally works at a low and medium noise intensity level[4]. But in the improved version, it works at a high-intensity noise level [5]. It makes a denoise image based on neighbors' uncorrupted pixels. BPDF process starts with clipping windows which are centered at each noise pixel. Clipping windows are not fixed, and it extends until the clipping window doesn't find any uncorrupted pixels. After that, evaluate the median value of pure pixels inside the clipping window and the median value with centered corrupted pixels. The BPDF performs well in high- and low-density noise levels[4]. But in medical MRI images, BPDF does not work well because it is designed for usual digital images where MRI images are always segmented.

2 DIGITAL & MEDICAL MRI IMAGES

2.1 Digital Image

Digital images illustrate authentic images stored in a computer as a set of numbers. Each set index is called pixels. Each pixel contains its pixel value which has a range[6]. For instance, gray images' pixel value ranges are between 0 to 255. All pixel values combined make a digital image.

Figure 2.1: A Digital Image

Figure 2.1 represents a digital image. It has a matrix, and each matrix index is called a pixel. Each pixel contains a pixel value depending on the image type.

All the matrix index values help to make a *Figure 2.1* digital image.

2.2 MRI Image

MRI is a medical scanning process that uses radio waves and a solid magnet to visualize inside the patient body[7]. MRI's complete form is Magnetic Resonance Imaging. MRI can get a closer look at bones, organs, and tissue, and this process is painless and safe for patients. MRI and CT scans look similar, but they have a significant difference. CT scan uses radiation to create images[8], but MRI uses magnetic fields to create images. So, it's clear that MRI much safer than CT scan[9].

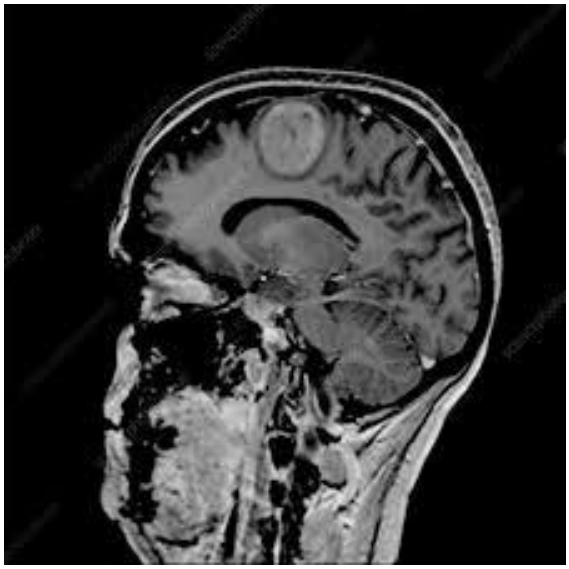


Figure 2.2: Magnetic Resonance Imaging

2.2.1 Working process of MRI system.

The major components of the MRI system are the Magnetic coil, Radiofrequency coil, and Gradient coil. All components are controlled through a digital computer[7]. A large, strong magnetic field is the heart of the MRI system. It produces two distinct effects, which are Tissue Magnatraction and Tissue Resonance. And work together to make an image[7], [10], [11]. It also has two magnetic characteristics such as field direction and field strength.

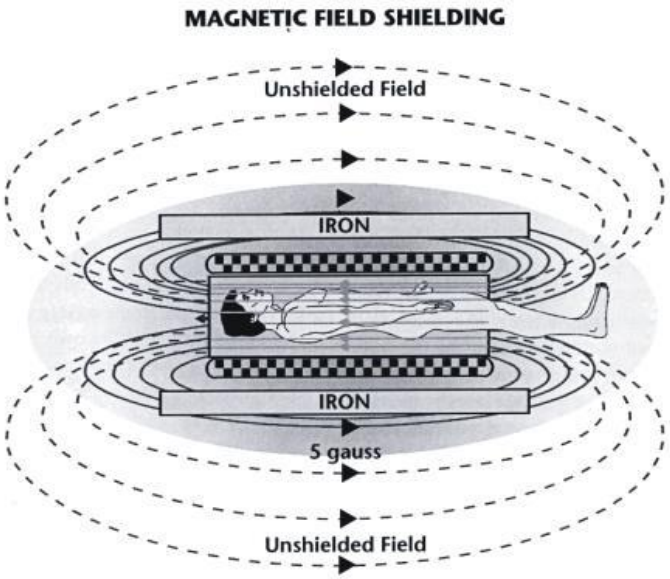


Figure 2.3: Magnetic Field

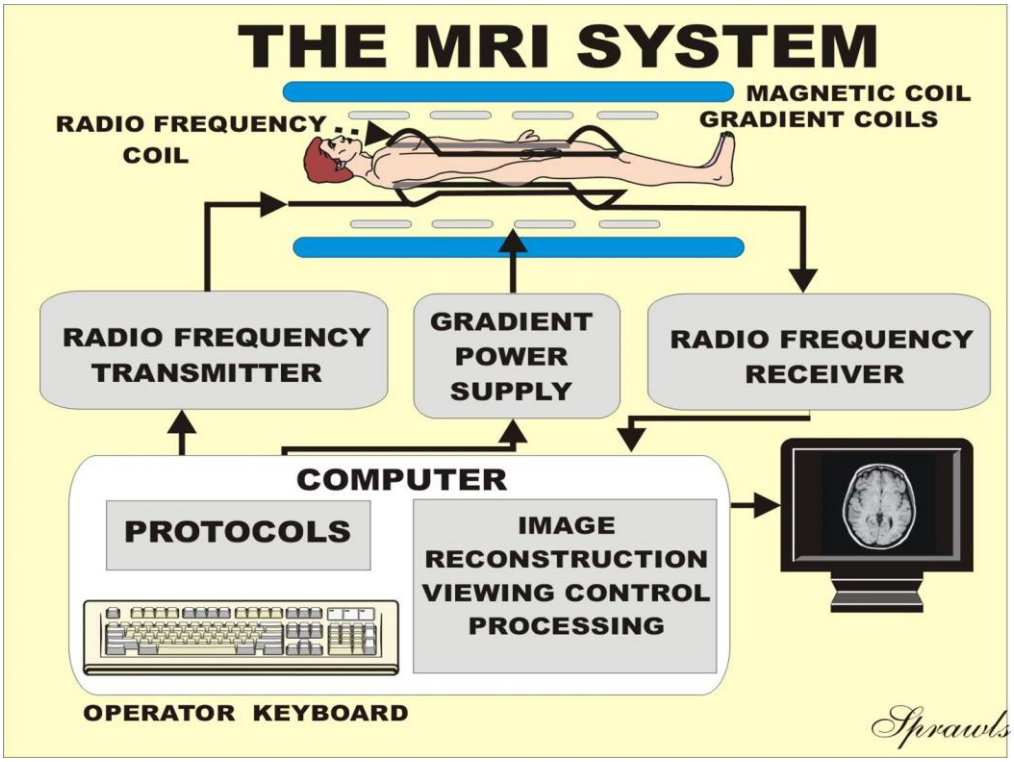


Figure 2.4: MRI process system

During radio waves, magnetic field, and gradient processing, the RF signal received from the patient's body and computer reconstruct the image[11]. And that's to get an MRI image like *Figure 2.1*.

2.2.2 Use cases of MRI scanner

As earlier mentioned, Magnetic Resonance Imaging (MRI) is used for patient health conditions. So, MRI use cases are related to health. It is commonly used to detect the disease such as:

- Internal bleeding
- Tumors
- Damage of injury
- Stroke
- Inflammation
- Brain development problem
- Blood vessels
- Tissue or Muscles Problem

It helps doctors with clear scanning images. An MRI scan is usually used for separate body parts depending on the patient's problem. MRI is beneficial for soft tissue and muscle problem detection[7], [11].

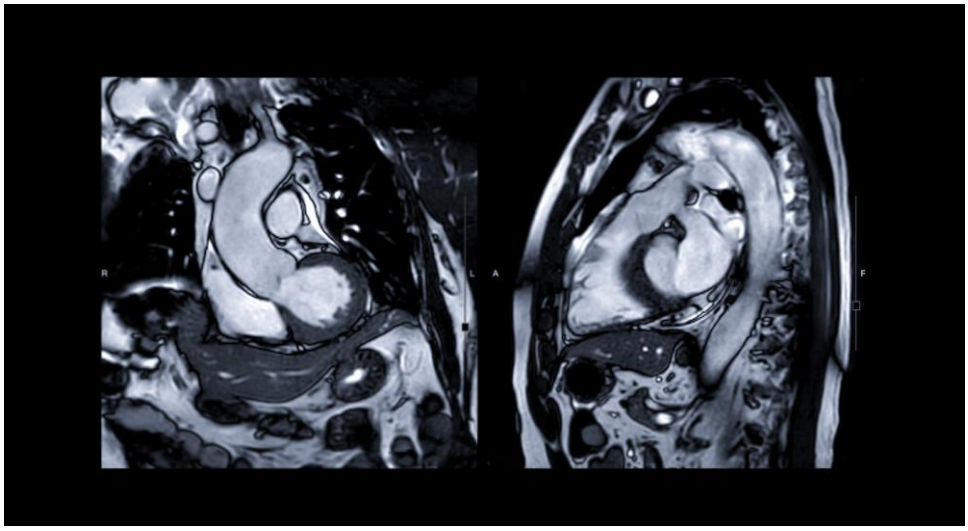


Figure 2.5: Stroke MRI Image

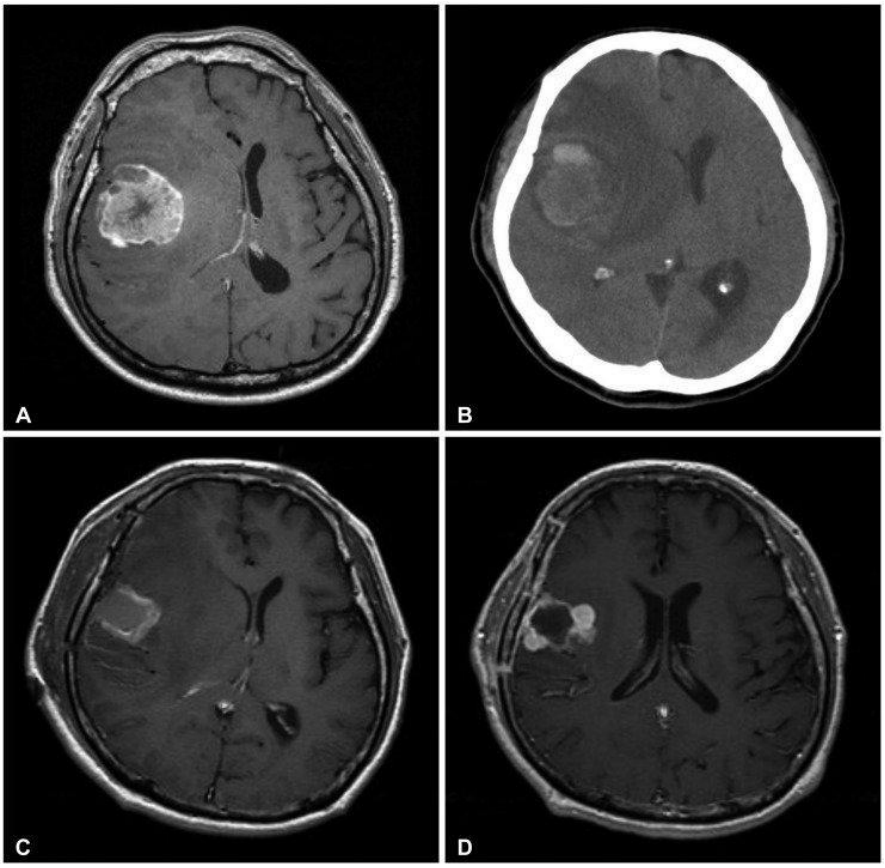


Figure 2.6: Brain Tumor MRI Image

2.3 Type of images

Images have different types, such as:

- Gray Image
- RGB Image
- Binary Image
- Indexed Image
- True color Image

These are the primary image types. The Gray Scale Images are in black and white color, and The RGB image is a combination of three colors. And the binary image is made of only black and white pixels [6], [12]. In this research, we work with only grayscale images. The study is based on MRI images, and we convert MRI images into gray photos.

2.3.1 Binary Images

Binary means it contains only two values. One is 0, and another is 1. That's why binary images are combinations of 0 and 1 images where every pixel has a 0 weight or 1. If the pixel value is 1, its color is white, and the pixel value is 0, then its color is black[13].

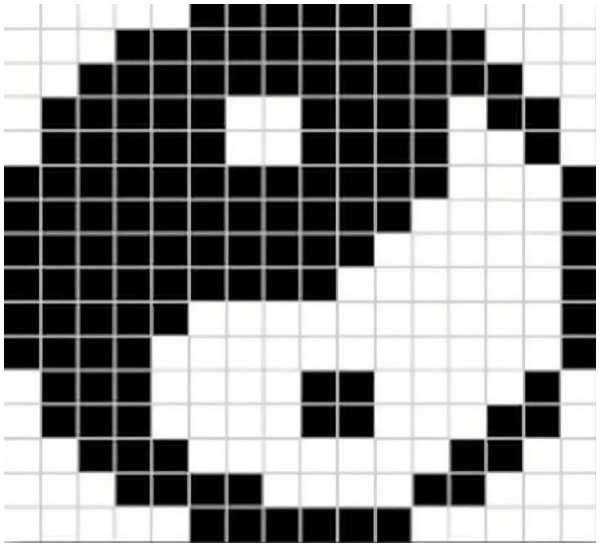


Figure 2.7: A Binary image

2.3.2 Grayscale Images

Gray images contain all pixel colors intensity between black and white. In gray ideas, pixel values are between 0 to 255. The lowest value, 0, defines pure black, and the highest value, 255, represents pure white color[6], [12]. All the pixel values are stored in a matrix value. And matrix classes can be uint8, uint16, int16, single, and double[14]. For single or double class matrices, intensity 0 is black, and intensity value 1 is white.

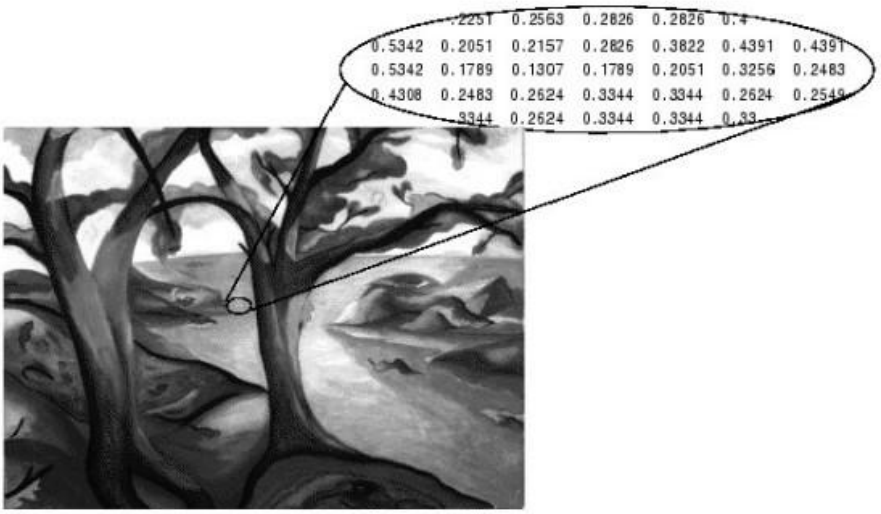


Figure 2.8: Gray image in double matrix class

2.3.3 RGB Images

RGB is a color model that combines three primary colors such as red, green, and blue. Different image processing systems use different color models[15]–[17]. Nowadays, most systems use RGB color models. RGB model represented by a 3-dimensional cube[12].

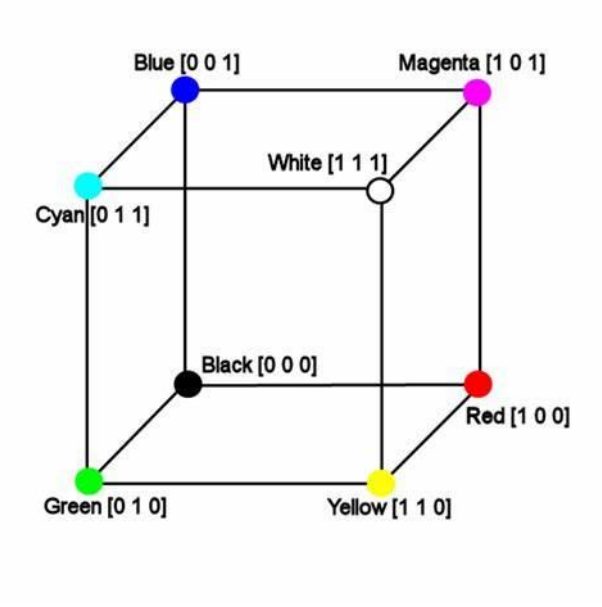


Figure 2.9: RGB 3-dimensional Cube

Each Cube axis represents the blue, green, and red colors. Every color contains three values, and the combination of three colors makes an RGB color. The origin of the cube is black (0,0,0)[18], [19].

Some standard methods are to convert an RGB color image to a grayscale image for a hardcopy of white and black print.

Gray image Intensity = $0.299R + 0.58G + 0.114B$ *(2.1)*

Another common method converter equation is:

Gray image Intensity= $0.333R + 0.333G + 0.333B$ *(2.2)*

These are very commonly used for converting RGB to a gray image.

3 DIFFERENT TYPES OF NOISE MODEL IN DIGITAL IMAGES

3.1 Noise and its types

Noise is something that appears in a digital image in image processing time. There are various causes to generate noise in digital images. It randomly changes pixel value depending on noise percentage and noise types[1]–[5], [14], [20], [21]. There have various ways to add noise to an image, such as:

- Image Acquisition
- Sensor size effect
- Sensor Hit

Mostly all the causes are related to the device. There have various types of noise, such as:

- Uniform Noise
- Gaussian Noise
- Salt and Pepper Noise
- Exponential Noise
- Rayleigh Noise

These are commonly detected in digital images. In the research, we work only on Salt and pepper noise. Salt and pepper noise is a powerful noise that changes pixel value with a long-range.

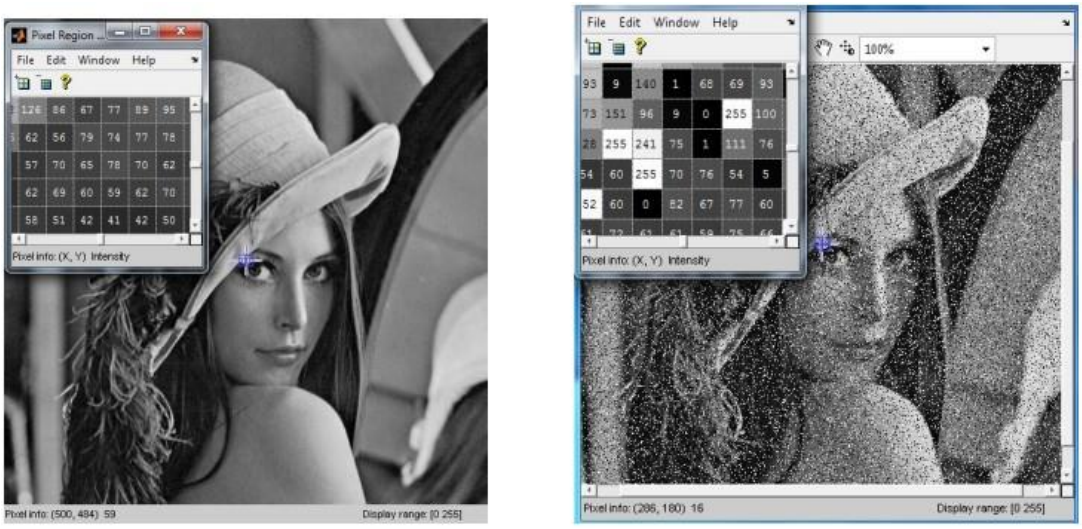


Figure 3.1: Before and after Noise Images

3.2 Noise Model

As earlier mentioned, there are various Noise types or models. Different noise models produce different noise patterns[22]. Gaussian and Salt and Pepper are commonly used in research to denoised images. Some noise model patterns are easy to denoise, and some are not. In the denoised process, many algorithms are applied to noisy images. For comparison of various algorithms, researchers should stick with one noise model to get the best evaluation result.

3.2.1 Gaussian Noise

The primary source of Gaussian noise is during the image acquisition, hitting issues, noisy sensor, etc.[23]. The probability density function (PDF) of a random gaussian variable x is given by:

$$p(x) = (1/\sigma\sqrt{2\pi}) * e^{-(x-\mu)^2/(2\sigma^2)} \quad (3.1)$$

here,

$x = gray\ level,$

$\mu = mean,$

$\sigma = standard\ deviation,$

$\sigma^2 = variance,$

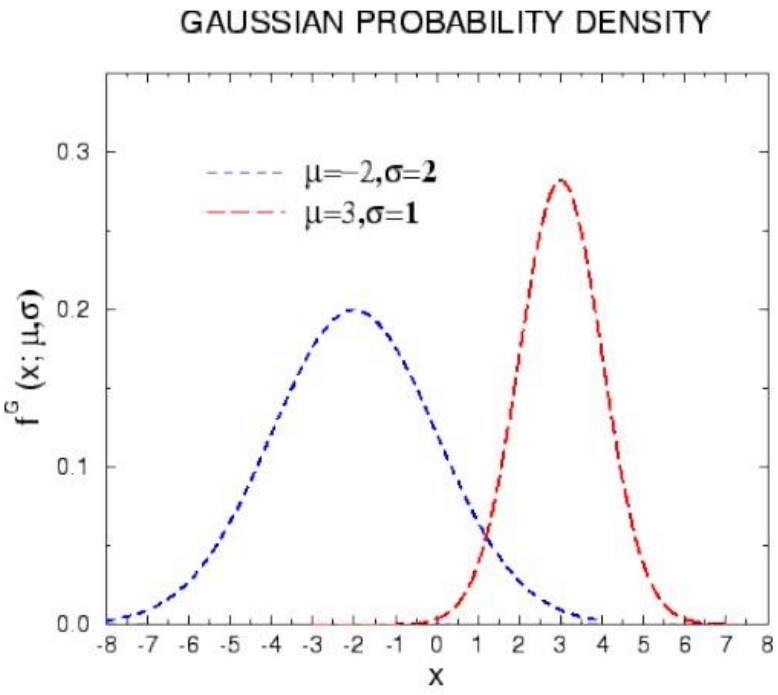


Figure 3.2: Gaussian noise with Gaussian amplitude distribution

Figure 3.3: Gaussian Noise comparison with the original image

3.2.2 Salt and Pepper Noise

Salt and pepper noise is a noise model where randomly selected pixels convert into Salt means white, and Pepper means black ultimately. It also has some variables to add noise to the image, such as density[24]. High-density Salt and pepper noise make it very difficult to denoise photos, and that part comes to perform denoise algorithms. As earlier mentioned, this research is based on Salt and pepper noise. We used 50% salt noise and 50% black noise for research purposes. That means total noisy pixels, 50% are white, and the other 50% are black. All processes are applied to gray images, so noise addition should be between black and white intensity color. This noise is also known as Impulse noise[24], [25].

The PDF of Salt and pepper noise is given by:

$$p(z) = \{P_a \quad for\ z = a, P_b \quad for\ z = b, 0\ otherwise\} \qquad (3.2)$$

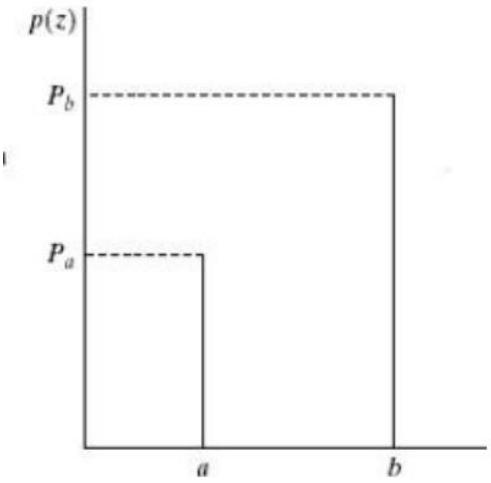


Figure 3.4: SAP Noise Plot

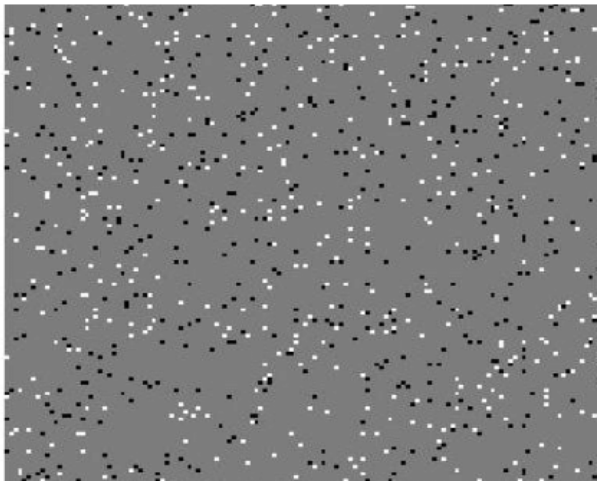


Figure 3.5: Salt and Pepper Noise Example

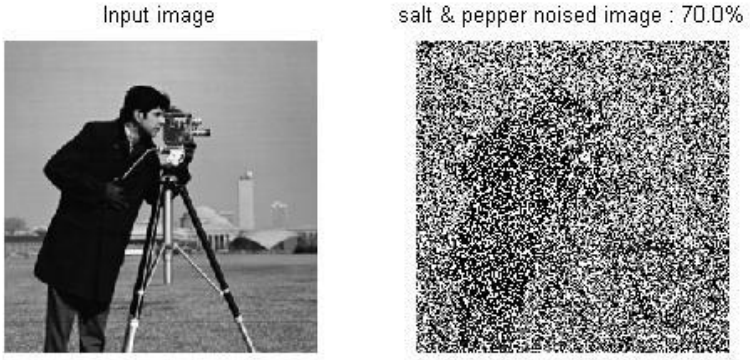


Figure 3.6: Salt and pepper noise comparison with the original image

4 PROPOSED ALGORITHM

4.1 Background

Recently image denoising methods are taking the attention of the signal processing community[1], in [26] that says that the Non-Local mean filter uses some different techniques or methods known as the local smoothing methods and frequency domain filters. Those methods primarily aim at noise reduction, and another essential part of an image is the reconstruction of the main geometrical configurations. But those methods are not aimed at the details of an image, texture, and maintenance of the fine structure. As we know that the Salt and Pepper noises are nothing, but the intensity level of a pixel is 0 or 255[24], [27]. If the intensity level of a particular pixel is 0, then we indicate that pixel as a pepper noise.

The pixel's intensity level, on the other hand, is 255. We then label that pixel as salt noise. Salt and pepper noise appears as a black or white pixel in a digital picture at random[34]. The destruction of the visual signal is frequently used to describe image noise. Because a digital image is nothing more than the intensity of pixels assigned in a 2D matrix. For external disturbance, the image is influenced by noise. A digital picture is frequently affected by signal noise known as salt and pepper noise as a result of connection failures [24].

Different methods for reducing Salt & Pepper noise from a digital picture exist. Every technique is distinct. One method for reducing Salt and Pepper noise from a digital image is to use a non-local mean filter. Based on the Non-Local Mean Filter, there are several types of Salt and Pepper noise reduction strategies. However, for conventional black and white photographs, the limitations of such approaches or filters work effectively. If we apply that technique to a medical digital image, those techniques cannot perform well because the medical image contains too many black pixels or black pixels. Those techniques work based on searching for black or white pixels, searching for Salt and Pepper in the digital image. Here creates a problem with the medical image of those techniques. Filters are unable to discriminate between original and noisy pixels because there are too many black pixels. Because such strategies operate, every pixel with an intensity level of 0 or 255 is considered noisy. The result is 0 because the medical picture comprises pixels; those algorithms regarded the original pixels as noisy pixels.

4.1.1 Grayscale Image

Figure 4.1: A black and white image

In *Figure 4.1*, we see a digital black and white image. In the selection part, we also see the darkest region in which the intensity level of those pixels could be zero.

228	228	226	235	232	221	214	190	123	54	34	34	37	40	37	33	38	47	48	44	44	48	48	45	45	49
228	228	235	232	228	228	221	181	103	34	37	36	39	41	39	37	42	50	47	44	44	47	48	46	48	52
228	229	235	227	224	222	195	135	69	28	43	41	43	48	48	44	46	50	46	43	43	47	49	48	50	55
228	229	227	223	223	209	154	81	39	35	40	39	42	48	49	44	42	44	46	42	42	47	49	49	52	57
228	230	221	221	212	178	118	60	35	37	43	41	39	38	39	42	46	48	49	50	50	49	49	50	54	57
226	226	214	215	192	139	78	41	35	41	41	41	41	41	42	43	44	45	46	48	51	51	52	53	57	60
222	219	210	193	147	84	41	33	39	41	41	42	43	44	44	44	42	42	44	47	51	53	54	56	58	60
219	211	194	150	89	46	36	42	44	38	44	44	44	44	44	43	43	43	44	48	52	54	54	54	55	56
209	194	149	99	52	40	45	45	42	43	48	46	44	42	41	43	45	46	45	48	52	53	51	50	50	51
180	154	90	59	42	49	49	36	37	51	49	47	43	41	41	44	47	50	47	49	51	51	49	48	48	49
129	92	52	43	43	51	49	39	39	50	46	45	44	44	44	46	48	49	48	50	51	50	49	49	51	53
86	42	44	42	42	46	51	51	46	41	42	43	45	47	48	48	48	48	49	50	50	50	49	51	54	58
54	48	36	41	48	54	55	50	43	37	42	46	51	53	52	50	49	49	52	48	46	47	53	57	57	56
46	47	49	48	45	44	43	43	42	42	42	44	48	50	50	51	53	55	58	55	51	51	52	53	50	48
43	47	52	47	42	39	40	42	43	43	41	43	46	47	49	52	57	61	61	58	56	54	53	51	47	43
43	44	40	40	41	45	48	47	43	38	41	44	47	49	50	52	56	59	55	55	55	55	54	52	49	47
41	41	39	39	41	46	51	49	42	35	42	46	51	53	53	52	52	53	51	53	54	55	53	51	50	49
45	49	51	45	40	41	45	46	42	37	44	48	53	55	54	52	50	49	53	54	55	53	49	47	46	46
46	51	55	47	39	39	44	47	44	40	46	49	52	53	52	51	51	52	53	54	54	51	46	43	44	46
40	42	49	43	39	42	50	52	47	41	48	50	51	50	50	50	53	56	50	52	52	48	44	42	45	49
45	46	43	41	43	49	51	48	48	51	44	52	58	58	52	49	54	60	49	52	54	52	52	52	46	38
44	46	46	43	45	50	52	50	50	54	44	49	53	53	49	47	50	53	50	52	52	49	49	50	46	39
43	46	43	40	40	45	47	45	47	51	52	53	54	55	54	53	52	51	51	52	50	45	45	48	46	40
42	46	43	39	40	44	46	44	46	51	58	56	54	55	57	57	53	49	51	51	47	41	42	47	46	42
41	45	49	46	47	52	53	52	53	57	56	53	50	51	54	54	49	43	50	49	44	39	41	47	49	46
41	44	48	46	49	54	56	53	53	56	57	54	52	53	56	55	51	46	47	47	43	39	41	49	52	49

Figure 4.2: Intensity Levels of pixels of the grayscale image

In *Figure 4.2*, we see the actual intensity level of those pixels. Here we see that the darkest part of the image has an intensity level of around 40 to 60. It means the image has no 0-intensity level in the whole

picture. As a result, the traditional non-Local mean filter performs well on those images to remove Salt and Pepper noise.

4.1.2 Medical Image

Magnetic Resonance Image (MRI), Computer Tomography (CT), Positron Emotion Tomography (PET), Magnetic Resonance Spectroscopy (MRS), and others are typical tools for any form of diagnostic. In the clinic, these procedures are commonly employed to diagnose any condition. In the medical world, every technique has its advantages[36]. Some of these processes result in a 2D image, while others result in a 3D image. [37], [38] Both dimensions' views are valuable.

4.1.3 Registration

For many applications, registration of two photographs of the same portion of the body is required because the correspondence between the two photos gives the relevant information. These two pictures can be created using separate modalities, such as CT and MRI, or they can be from the same patient obtained with the same device at different periods, or they can be from two distinct patients.

Now let's see how to look at the medical digital image. We already discussed that digital medical photos contain a lot of black pixels whose intensity levels are 0.

Figure 4.3: A medical (MRI) image

In *Figure 4.3*, we can see that the selected portion of the image is entirely black. It means that the intensity levels of that area are 0. Now let's see the actual intensity levels of the selected region.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4.4: intensity levels of medical image

In *Figure 4.4*, we see that the actual intensity levels of the area of the medical image are all 0s.

If we use a typical mean filter to process a noisy (Salt and Pepper) medical picture, the filters must treat the black pixels as noisy pixels and operate on their value.

4.2 NOISE REMOVAL TECHNIQUES

4.2.1 Standard Mean Filter

The pixel's values are arranged in a 2D matrix in the digital image. Every pixel has its position denoted as a row and column, and its intensity level is represented as a pixel value. A total number of rows and columns is indicated by the width and height of that image. On the other hand, we can call it the size of that image. Intensity levels describe how look the image is. A grayscale image has one channel of intensity levels, and an RGB image or colored image has three channels of intensity levels Red, Green, and Blue[12]. All kinds of images can be affected by different types of noise. Such as Gaussian noise, Salt and Pepper noise, Shot noise, Quantization noise, Film grain, Periodic noise, etc.[28]–[30]. We remove only Salt and Pepper noise from the medical image in our work. To remove any noise, scientists developed different types of noise removal techniques, such as mean filter, median filter, etc., only the mean filter. Our work is based on the mean filter to remove Salt and Pepper noise from the medical image.

4.2.2 Non-Local Mean Filter

Non-Local Mean Filter is a recently developed noise removal technique widely used for denoising images. The Non-Local Mean Filter modification is the 'Yaroslavsky Filter' [21], which averages image pixels according to their intensity distance. SUSAN and bilateral filters, for example, are based on the same idea. The main difference between the Non-local Mean filter and those techniques is that the similarity of the intensity levels is very much the strength of the noise level by using region comparison rather than intensity contrast. Pattern redundancy is not limited to being local. That's why the filter is known as non-local[21].

Now discuss how the Non-Local Mean Filter works precisely. The word non-local means that the value for the noisy pixels that we wanted to fix or repair has come from another part of the image that is not local for the pixel. We can find many similar patches in different locations[31], [32].

Figure 4.5: Similar patch in natural image

In *Figure 4.5,* A natural digital image contains many similar patches in different locations. From *Figure 4.5, we* can say that *a1*, *a2*, *a3*, and *a4* are similar patches, *b1*, *b2* and *b3* are identical, and *c1*, *c2*, and *c3* are similar.

Those similar patches are beneficial for removing noise from the image. The main goal of the Non-local Mean Filter is to replace a noisy pixel value with an average value which is also called the mean value o nearby pixels that have a similar surrounding region. On the other hand, we can say that a similar patch.

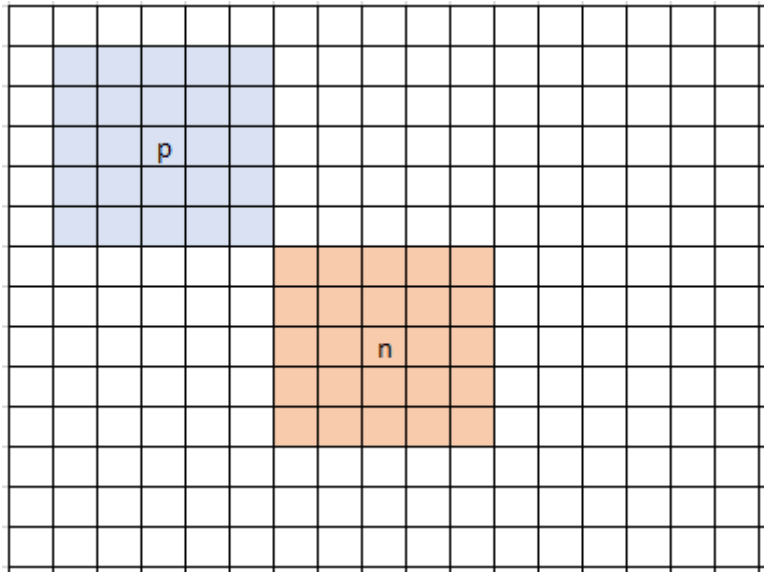


Figure 4.6: Tow similar patches, p: non-noisy patch, n: noisy patch

Figure 4.6 p is the center pixel of the non-noisy patch with a similar patch where *n* is a noisy center pixel of that patch. Since they are identical patches and *n* is a noisy pixel, we need to calculate the average or mean value of that region where *p* is in the center pixel. We cannot calculate the average value of that region where *n* is the center pixel. Because if we estimate that, it will not give us a better result, and it will not be a non-local mean. That's why we need to calculate the average of the pixels of the *p* region. The development of the Standard will be the value of n or the noisy pixel. After replacing the noisy pixel, the filter searches for the next noisy pixel, and so on. That's how the non-local mean filter work in most cases.

The above discussion might clarify why it’s called Non-Local. Because the corrupted pixel value can not come from its neighbor pixels, it’s come from another part of the same image which is very much similar to that corrupted pixel region. The value is not a came from its local pixels, it came from a non-local part or so-called region. that’s why it’s named Non-local. We calculate the average value of those pixels which is known as the mean value for the corrupted pixel.

4.3 Non-Local Mean Recursive Filter

Above, we discuss the Non-Local Mean Filter. What is the Non-Local Mean filter, description, method, etc.?

Now we will discuss our filter or technique, which is also a Non-Local Mean filter-based filter, but it has one other approach: it works recursively. That's why we call it Non-Local Mean Recursive or NLMR. The method of NLMR is very much like traditional NLM, but NLMR works much better for denoising medical images. Here recursive means that when NLMR iterates the first time and denoise as much as the noise, it can check if there are any noisy pixels exit. If so, then the function NLMR calls itself and sends the semi denoised image as a noisy image input to perform the denoising technique. Let's see how NLMR works.

First, we input an image. And make the image grayscale. There is no difference between a grayscale image and a 2D matrix. The grayscale image contained rows and columns. For example, the 'lena.jpg' image size is 512×512, which means the image has 512 rows and 512 columns. We store the grayscale image in X. Now we need to add noise, in this case, "Salt & Pepper" noise. We added 70% "Salt & Pepper" noise in image X and the noisy image we stored in "*Xnoise*." Here we calculate the initial performance of noisy images and non-noisy images. After calculating performance, we call a function named "NLMR()," which takes one parameter, the noisy image, and returns another image, a cleaned or denoised image.

4.3.1 Working process of NLMR function

"NLMR" means non-Local means recursive. On the other hand, we can say that the function work based on non-local mean methos recursively. This function filters out most of the noisy pixels based on non-local mean value. After filtering out if there are any noisy pixels exist the function call it again with the parameter of the filtered images as a noisy image. And do the same process until there are no noisy pixels exist as shown as in Figure 4.7.

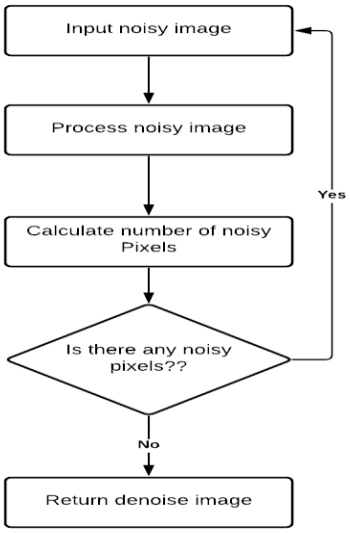


Figure 4.7: Working Process of NLMR

4.3.2 Process Noisy Pixels

In the function, the noisy image is named "Xnoise." Then we add padding to the noisy image on both sides symmetrically. The padding values are not random or copied from the image. That's why padding in an image does not make any difference. Now we need to mark noisy pixels and non-noisy pixels.

We create a matrix of zeros the same size as the padding image and name the matrix "isPadNoisy." Then we check the intensity level of every pixel of the padding image is 0 or 255 or in between them. If the intensity level is 0 or 255, we mark it as noisy by changing the value of the matrix of zeros to one. Now in the "isPadNoisy" matrix, there are only two different types of values. One zero (0) and another one is one (1).

1	0	1	0	0	0	0	1	0	1	1	0	1	1	0	1	1
1	1	0	1	1	1	1	0	1	1	1	0	0	1	1	0	1
1	1	1	0	0	0	0	1	1	1	0	0	0	0	0	1	1
1	0	1	1	0	0	1	1	0	1	0	1	1	1	0	1	0
1	0	1	1	1	1	1	1	0	1	0	0	1	1	1	1	0
1	0	1	1	1	1	1	1	0	1	0	0	1	1	1	1	0
1	0	1	1	0	0	1	1	0	1	0	1	1	1	0	1	0
1	1	1	0	0	0	0	1	1	1	0	0	0	0	0	1	1
1	1	0	1	1	1	1	0	1	1	1	0	0	1	1	0	1
1	0	1	0	0	0	0	1	0	1	1	0	1	1	0	1	1
0	0	0	1	1	1	1	0	0	0	1	1	1	0	0	1	1
0	0	1	1	1	1	1	1	0	0	0	1	1	1	1	1	0
1	1	0	0	1	1	0	0	1	1	0	1	0	0	0	1	0
1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0
1	1	1	1	0	0	1	1	1	1	1	1	0	1	1	0	1
1	1	1	0	1	1	0	1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0
1	0	0	1	1	1	1	0	0	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1	1	0	1	1	0	1	1	1	1
1	1	1	1	0	0	1	1	1	1	1	1	0	0	1	1	1
1	1	0	1	0	0	1	0	1	1	1	0	1	0	0	0	0

Figure 4.8: A 15×15 window where the center pixel is noisy

We calculate the number of rows and columns in "M" and "N." Then, we iterate each pixel by columns and columns. In each iteration, first, we check that the pixels of isPadNoisy(i,j) are 1(one) or not. We execute the following process if the value is 1(One). If not, then we skip the pixel and check the next pixel. The pixels of isPadNoisy(i,j) are 1(one) or noisy, then we select a window size of 15×15 and name it "window" where the center pixel is the noisy pixel. Again, select another 7×7 window in the center of "window" and call it "center window." Now we check whether all pixels of the center window are noisy or not. On the other hand, we check that any noiseless pixels exist. If not, the function will skip the noisy pixel and do the same operation for the next noisy pixel.

Every natural image has similar patches from different locations [31], [32]. For that, now we search a similar patch of the center window in the main window. We iterate the main window and make the best match of the equivalent window of the central window. It means the similar window looks like the center window. It could be the same as the center window without noisy pixels.

Now let's see how NLMR finds a similar window.

164	163	166	164	163	162	164
162	165	166	163	165	166	166
165	162	166	162	162	162	166
165	164	166	165	165	162	165
162	166	165	163	166	165	165
162	165	166	166	163	163	165
164	164	165	162	164	165	166

Figure 4.9: a window of the original image

For example, we consider a 7×7 window for understanding better. In *Figure 4.9*, we a 7×7 window actual image. The pixels value is very similar because 7×7 is a tiny portion of the original image. Now let's see what happens when we add 70% of Salt and Pepper noise to the original image.

255	255	0	0	255	0	164
255	0	166	163	0	0	255
0	162	0	162	0	0	255
255	0	0	0	165	162	165
162	0	165	0	255	0	0
255	255	0	166	0	0	0
255	0	165	162	164	0	166

Figure 4.10: A 7×7 main window

In *Figure 4.10*, we see a noisy portion of the noisy image after adding 70% Salt and Pepper noise to the original image. Here we can see that most of the around 70% pixels value is affected by Salt and Pepper noise. And also, the center pixel is noisy. As we consider the main window to be a 7×7 window at this

point, we need to select another window whose size will be 3×3, and it also is in the center of the main window, and we call it the center window its position in the center.

0	162	0
0	0	165
165	0	255

Figure 4.11:Center window

Figure 4.11 is the center window of that main window. Now we need to search for similar patches in the main window. There is a similar patch exit inside the main window since it is a natural image. To find a similar patch, we need to search the main window from 1st to last by a 3×3 search window. In our example, the total number of searches will be 24. Let's see how the number became 24.

$S = 3$

$mw = (S * 2) + 1$

$mw = (3 * 2) + 1$

$mw = 6 + 1 = 7$

It means that the center window size is 7×7. Now for the search window,

$l = 1$

$sw = (l * 2) + 1$

$sw = (1 + 2) + 1 = 3$

It means that the search window size is 3×3. If we search a 7×7 window with a 3×3 search window, then the total number of search windows will be, or the total number of iterations will be

$Total\ number\ of\ iterations = \{(mw - sw + l) * (mw - sw + l)\} - 1$

$= \{(7 - 3 + 1) * (7 - 3 + 1)\} - 1$

$= \{5 * 5\} - 1$

$= 25 - 1$

$= 24$

Here -1 is for we ignore the center window as a search window. If we count the middle window as a search window, the central window will match the maximum similarity every time. We must ignore the center window when we iterate for searching for a similar patch.

For every iteration of the loop, the search window will be different from each other, or it can be similar too. For our example, all the search windows are given below.

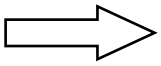
255	255	0	255	0	0	0	0	255	0	255	0	255	0	164
255	0	166	0	166	163	166	163	0	163	0	0	0	0	255
0	162	0	162	0	162	0	162	0	162	0	0	0	0	255
255	0	166	0	166	163	166	163	0	163	0	0	0	0	255
0	162	0	162	0	162	0	162	0	162	0	0	0	0	255
255	0	0	0	0	0	0	0	165	0	165	162	165	162	165
0	162	0	162	0	162				162	0	0	0	0	255
255	0	0	0	0	0				0	165	162	165	162	165
162	0	165	0	165	0				0	255	0	255	0	0
255	0	0	0	0	0	0	0	165	0	165	162	165	162	165
162	0	165	0	165	0	165	0	255	0	255	0	255	0	0
255	255	0	255	0	166	0	166	0	166	0	0	0	0	0
162	0	165	0	165	0	165	0	255	0	255	0	255	0	0
255	255	0	255	0	166	0	166	0	166	0	0	0	0	0
255	0	165	0	165	162	165	162	164	162	164	0	164	0	166

Figure 4.12:Search windows for every iteration

In *Figure 4.12*, We see that the total number of iterations is 25. But the number of search windows is 24 because we ignore iteration 13, as shown in *Figure 4.12.*

Here our work is not done yet. We cannot find critical information from those windows to find the best matching patch. After selecting a search window, we must mark those non-noisy pixels as 1(one) and noisy pixels as 0. For the center widow, the result will be

0	162	0
0	0	165
165	0	255



0	1	0
0	0	1
1	0	0

Figure 4.13:Conversion of center window, noisy = 0, non-noisy=1

All search windows are shown below.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	1	1	1	1	0	1	0	0	0	0	0
0	1	0	1	0	1	0	1	0	1	0	0	0	0	0
0	0	1	0	1	1	1	1	0	1	0	0	0	0	0
0	1	0	1	0	1	0	1	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	1	1	1	1	1
0	1	0	1	0	1	0	1	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	1	1	1	1	1
1	0	1	0	1	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	1	1	1	1	1
1	0	1	0	1	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	1	0	0	0	0	0
1	0	1	0	1	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	1	0	0	0	0	0
0	0	1	0	1	1	1	1	1	1	1	0	1	0	1

Figure 4.14: Convert search windows. 0 for noisy and 1 for non-noisy

After that conversion, we multiply all center window values by every search window value. As the middle and search windows have the same rows and columns, the multiplication work as scalar multiplication or element by element multiplication. After the multiplication of all search windows values, the result of all windows is given below. For 1st search window, the result will be like this,

0	1	0		0	0	0		0	0	0
0	0	1	×	0	0	1	=	0	0	1
1	0	0		0	1	0		0	0	0

Figure 4.15: Element by element multiplication of center window and 1st search window

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	1	0	0	0	0				0	0	0	0	0	0
0	0	0	0	0	0				0	0	1	0	0	1
1	0	0	0	0	0				0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	1	0	0	1	0	0

Figure 4.16:The result of the element by element of search windows and center window

At this point we again multiply element wise the last result to the search windows pixels values. In this stage every search windows element could be 0 or a non-noisy value. If we apply this rule the result will be,

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	166	0	0	163	0	0	0	0	0	0	0	0	0
0	0	0	162	0	0	0	0	0	162	0	0	0	0	0
0	0	0	0	166	0	0	163	0	0	0	0	0	0	0
0	0	0	0	0	162	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	165	0	0
0	162	0	0	0	0				0	0	0	0	0	0
0	0	0	0	0	0				0	0	162	0	0	165
162	0	0	0	0	0				0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	165	0	0	162	0
0	0	165	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	166	0	0	0	0	0
0	0	0	0	165	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	166	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	165	0	0	162	0	0	164	0	0

Figure 4.17: Extract the non-noisy values from the search windows.

Now we calculate the difference of center window's elements and search window's elements.

0	162	0	0	162	0	0	162	0	0	162	0	0	162	0
0	0	1	0	0	2	0	0	165	0	0	165	0	0	165
165	0	0	3	0	0	165	0	0	3	0	0	165	0	0
0	162	0	0	4	0	0	1	0	0	162	0	0	162	0
0	0	165	0	0	3	0	0	165	0	0	165	0	0	165
165	0	0	165	0	0	165	0	0	165	0	0	0	0	0
0	0	0	0	162	0				0	162	0	0	162	0
0	0	165	0	0	165				0	0	3	0	0	0
3	0	0	165	0	0				165	0	0	165	0	0
0	162	0	0	162	0	0	162	0	0	3	0	0	0	0
0	0	0	0	0	165	0	0	165	0	0	165	0	0	165
165	0	0	165	0	0	165	0	0	1	0	0	165	0	0
0	162	0	0	3	0	0	162	0	0	162	0	0	162	0
0	0	165	0	0	1	0	0	165	0	0	165	0	0	165
165	0	0	165	0	0	0	0	0	3	0	0	1	0	0

Figure 4.18:Element wise absolute difference from the center window

When we get the element-wise absolute difference from the middle window, as shown in *Figure 4.18,* we add up all the elements of every search window. For all search windows, the result looks like this.

328	167	492	330	492
492	172	331	492	327
168	492		330	327
327	492	492	169	330
492	169	327	330	328

Figure 4.19: Summations of search windows elements

Finally, we got our desired values to compare those search windows or very similar patches of the center window. *Figure 4.19* shows the dissimilarity value from the middle to the search window. The bigger the value is more significant the difference from the center window to the search window. That means we must select that search window that has the smallest number. The smallest number indicates that the center and search windows have a similar patch. Those numbers in *Figure 4.19* are helping us to find out a similar patch. Here we can see that the smallest number is 167. The position of value 167 is in the 1^{st} row and 2^{nd} column. That means that the search window of $(1\times2) = 2^{nd}$ iteration is the most like the center. For that, we select the 2^{nd} search window for the denoise pixel value of the center window. The search window will be similar. When we get a similar window, we need to cluster the pixel intensity of the equivalent window between two clusters, one with noisy and noiseless pixels. When we get those noiseless pixels' intensity levels, we then calculate those noiseless pixels' average (mean). The average or mean value will be the intensity level of the center noisy pixel.

255	0	0
0	166	163
162	0	162

Figure 4.20: Similar window

From *Figure 4.20: Similar window*, we calculate the average or mean value for the noisy pixel. The average value is

$$\frac{summation\ of\ non-noisy\ values}{total\ number\ of\ non-noisy\ values} \qquad (4.1)$$

$$= \frac{166 + 163 + 162 + 162}{4}$$

$$= \frac{653}{4} = 163.25$$

The value of pixel intensity cannot be a floating-point number. In image processing, the value of pixel intensity level has a limit of 0 to 255. And those values must be an integer. On the other hand, the data type of those values is uint8. unit8 is a data type ranging from 0 to 255 or an 8-bit number system[33]. So, we cannot take the result of the average value of 163.25. instated of 163.25, we must take 163 as the intensity level of the noisy pixel. That's how the noisy value pixel replace.

After all iterations, we get a denoise image that can contain some number of noisy pixels if the noise level is high. If any noisy pixels exist, the denoised image will be marked as noisy. Finally, the function will call itself with the parameter of the denoise image, which is known as recursive. It will be calling itself until there are no noisy pixels.

When all noise is removed, the function will return the denoise image.

4.4 Algorithm

The steps of the algorithm

Step 1) Input an image, make it grayscale, add Salt and Pepper noise, add 5×5 padding to the noisy image, and save it as padding_noisy_image.

Step 2) Create a binary matrix where 1 indicates noisy pixel and 0 indicates noiseless pixel. Calculate the total number of rows and columns of the matrix.

Step 3) initialize i and j = padding size, which is 5. Run a for loop which runs until the i and j are less than or equal to the $number\ of\ rows - padding$ and $number\ of\ colums - padding\ siez.$

> ***Case (i):*** Select a 2-dimensional window W of size 11×11. Select the center pixel and mark it $p(cx, cy)$. And check the $p(cx, cy)$is a noisy pixel. If not noisy, then increase the value of i and j, which means checking the next pixel.
>
> ***Case (ii):*** Now select another window CW of size 3×3 in the center of W.
>
> ***Case (iii):*** Find a similar patch of CW in the window W. To find out the similar patch of CW, we calculate dissimilarity. The value of dissimilarity indicates how similar are two noisy patches. Minimum dissimilarity means the most similar patch.
>
> ***Case (iv):*** From a similar patch of CW, take the non-noisy values, calculate their mean, and replace the pixel intensity level with the mean of non-noisy values.

Step 4) After replacing all noisy pixels in the first iteration, calculate the number of noisy pixels from the denoised image. If any noisy pixels exist, then repeat Steps 2 and 3 with the input of the denoise image as a noisy image.

Step 5) if there are no noisy pixels in the denoised image, calculate its performance and exit.

4.5 Flowchart

The flowchart of our filter is given below

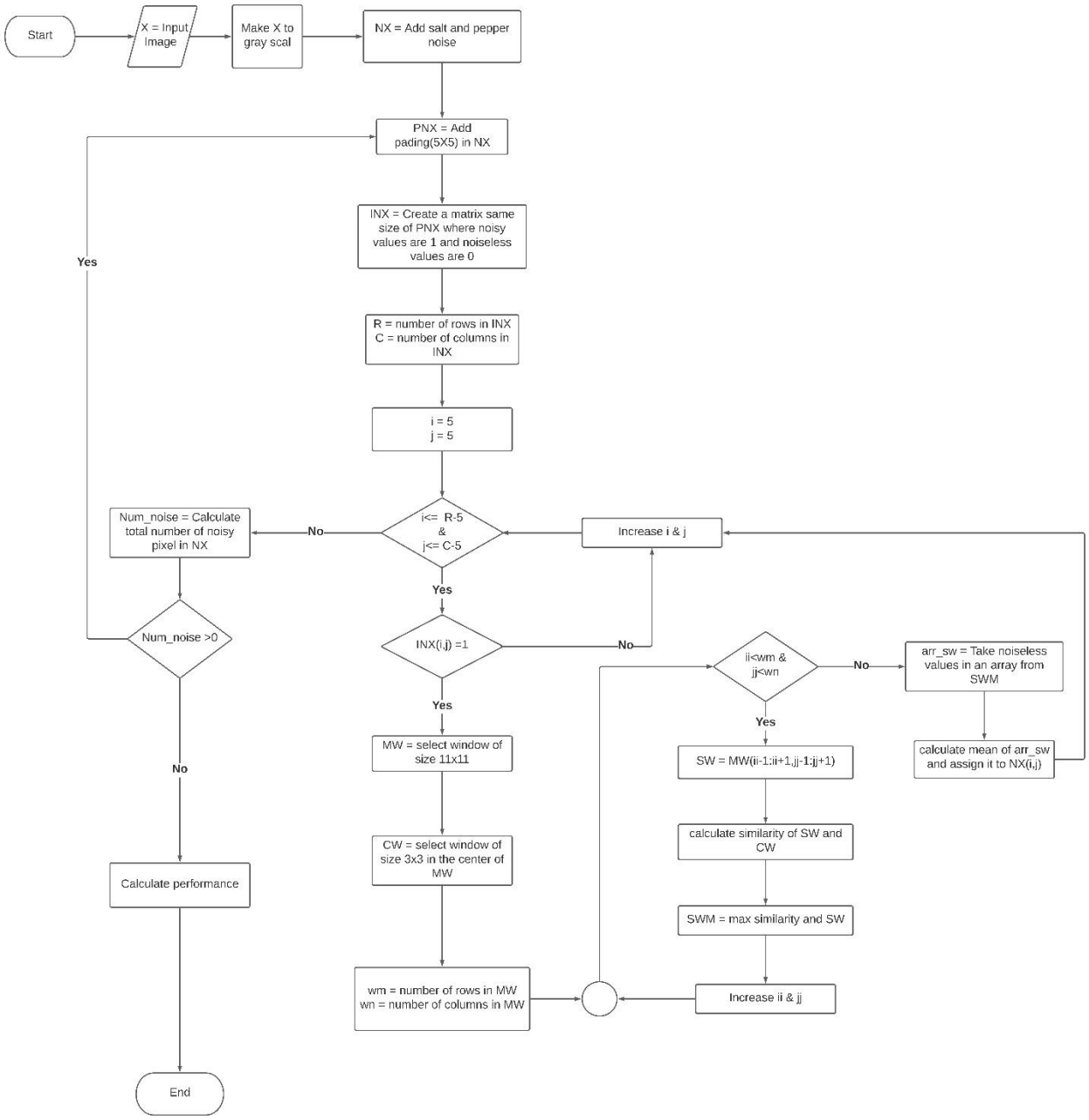


Figure 4.21: Flowchart of the Non-Local Mean Recursive Filter

4.6 Medical Image Denoising

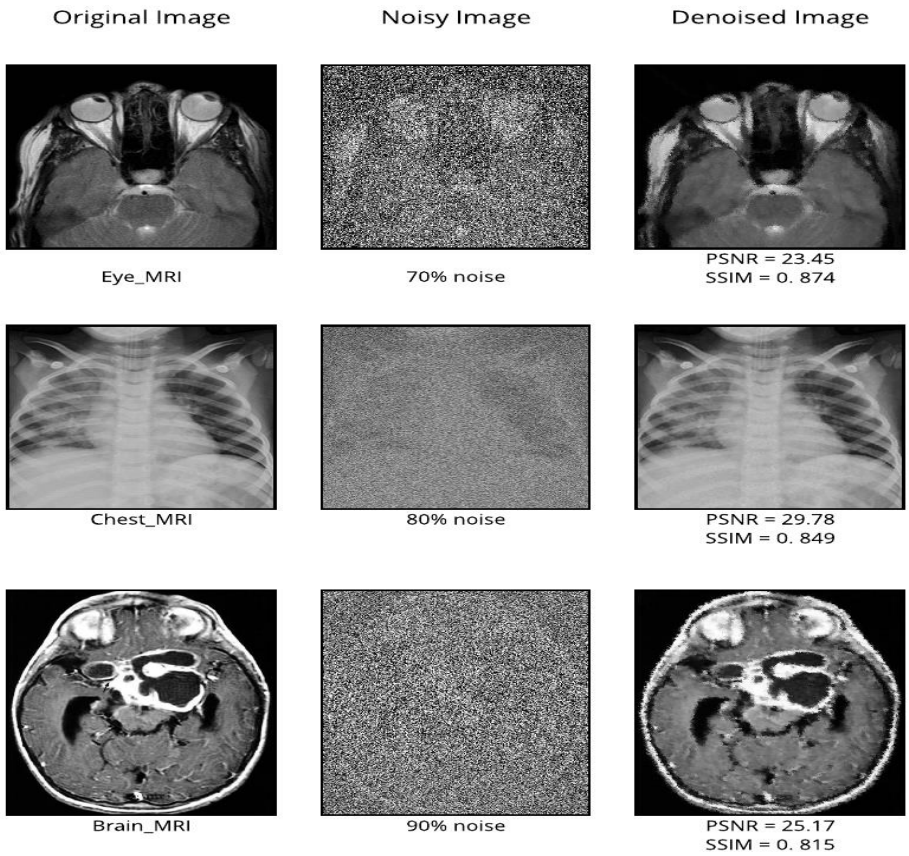


Figure 4.22: non-local mean recursive image denoised

In *Figure 4.22* we show our main image denoised process. At first, we input the digital medical image. 2nd we add different types of noise to the image. And finally, we apply our proposed algorithm to the noisy image to remove noise as much as we can. The proposed algorithm which is known as the Non-Local Mean Recursive filter work to remove noise from the given image as much as possible. The working process of the algorithm is recursive. Because, the first iteration can never remove all noise from an image if it is a medical image. Because medical images contain pure black region. To remove noise from a pure black region we use recursiveness which helps us to remove all noise from the pure black region.

5 SIMULATION & PERFORMANCE ANALYSIS

5.1 Simulation

To evaluate the performance of our proposed algorithm, we implement four more methods, such as the Adaptive Switching Weight Mean Filter (ASWMF), which is Based on Pixel Density Filter (BPDF). Different Applied Median Filter (DAMF) and Noise Adaptive Fuzzy Switching Median Filter (NAFSMF) algorithms. Those algorithms work well in Salt and pepper noise images. But the challenge is how they work with Medical MRI images. As we know, MRI images are segmented images. Measure the methods' performance, the value of Peak Signal and Noise Ratio (PSNR) and Structural Similarity Index (SSIM) are used.

PSNR is measured by "dB," and It's defined as follows:

$$PSNR = 10\, log_{10}(255^2/MSE) \tag{5.1}$$

And,

$$MSE = \left(\Sigma_{R,c}(O(r,c) - D(r,c))^2/R \times C\right) \tag{5.2}$$

$$here,$$

$$O = original\ Image,$$
$$D = Denoised\ Image,$$
$$R = Number\ of\ Rows,$$
$$and\ C = Number\ of\ Columns$$

Eq 5.1 & *Eq 5.2* measure the PSNR value to determine algorithms' performance. SSIM is also used to evaluate the denoised performance. It measures the structural resemblance of denoised and original images.

$$SSIM(i,j) = \frac{(2\mu_i\mu_j + P_1)(2\sigma_{ij} + P_2)}{(\mu_i^2 + \mu_j^2 + P_1)(\sigma_i^2 + \sigma_j^2 + P_2)} \qquad (5.3)$$

$$here,$$

$$\mu_i = image^i\ mean\ ,$$

$$\mu_j = image^j\ mean$$

$$\sigma_i = image^i\ standard\ deviation\ ,$$

$$\sigma_j = image^j\ standard\ deviation\ ,$$

$$\sigma_{ij} = covariance\ of\ i\ and\ j$$

We used various MRI scan images of body parts in our research. Such as the brain, eyes, chest, and brain side view MRI scan images. We select one photo from each category to analyze and visualize the purpose.

5.2 Performance analysis

In the analysis and visualization, we use PSNR and SSIM to measure the performance of different algorithms. Those data are showcased as line charts, tables, and comparison images.

Figure 5.1: Eye MRI

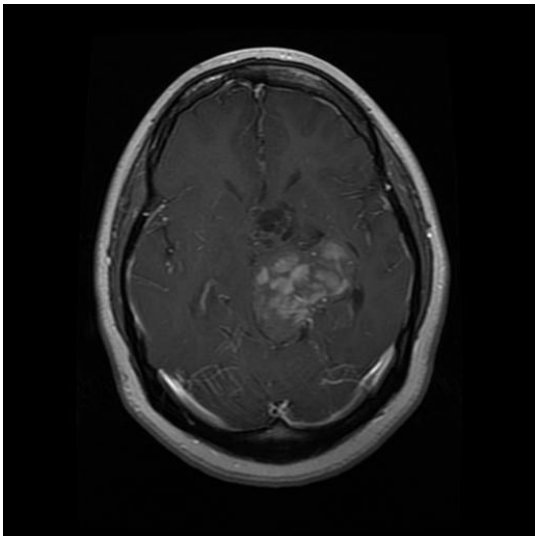


Figure 5.2: Brain MRI

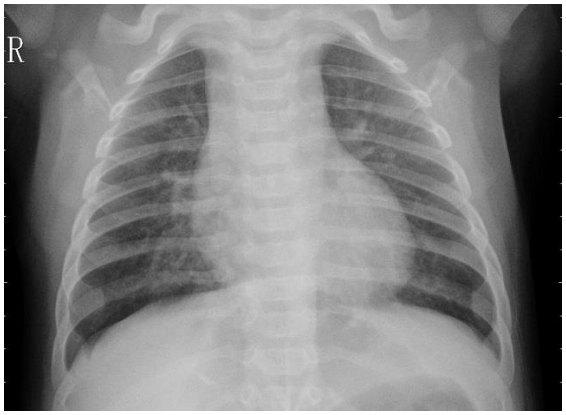


Figure 5.3: Chest MRI

*Figure 5.1**,** Figure 5.2, and Figure 5.3 are Original Images to measure the performance**.***

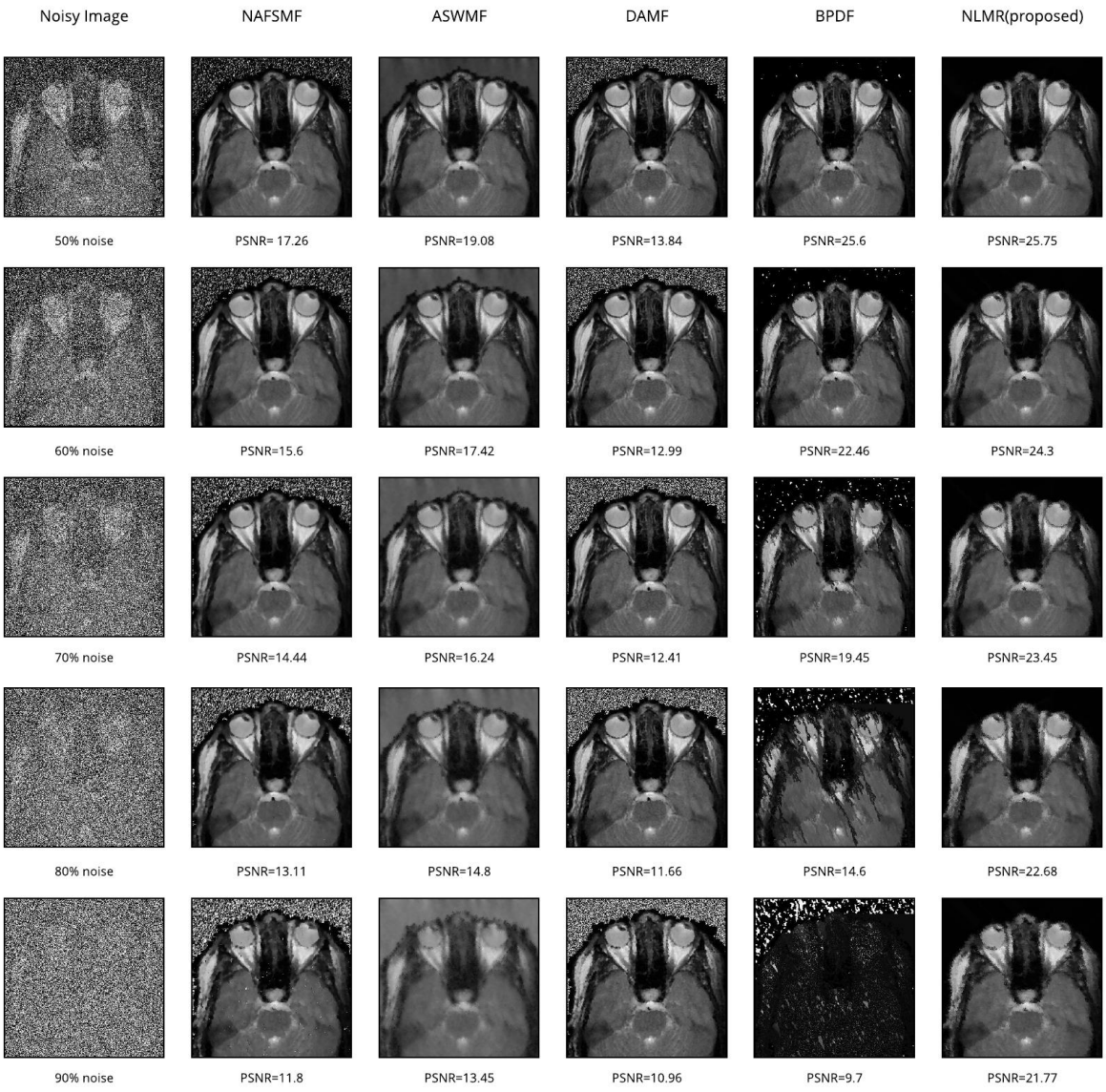


Figure 5.4: Eye MRI image De-noise comparison with PSNR

In *Figure 5.4* we displayed the final result and compare other algorithms with our proposed algorithm. Some noise removal algorithms are working very well, but their limitation is those algorithms has not the recursive functionalities. For this reason, those algorithms work very well with a normal image that contains no pure black or white region. But they cannot work properly for medical digital images. To overcome this particular problem we proposed our algorithm. Which work recursively and follow a developed algorithm called Non-Local Mean filter. The result of our algorithm and other algorithms for the eye MRI image is shown in *Figure 5.4*.

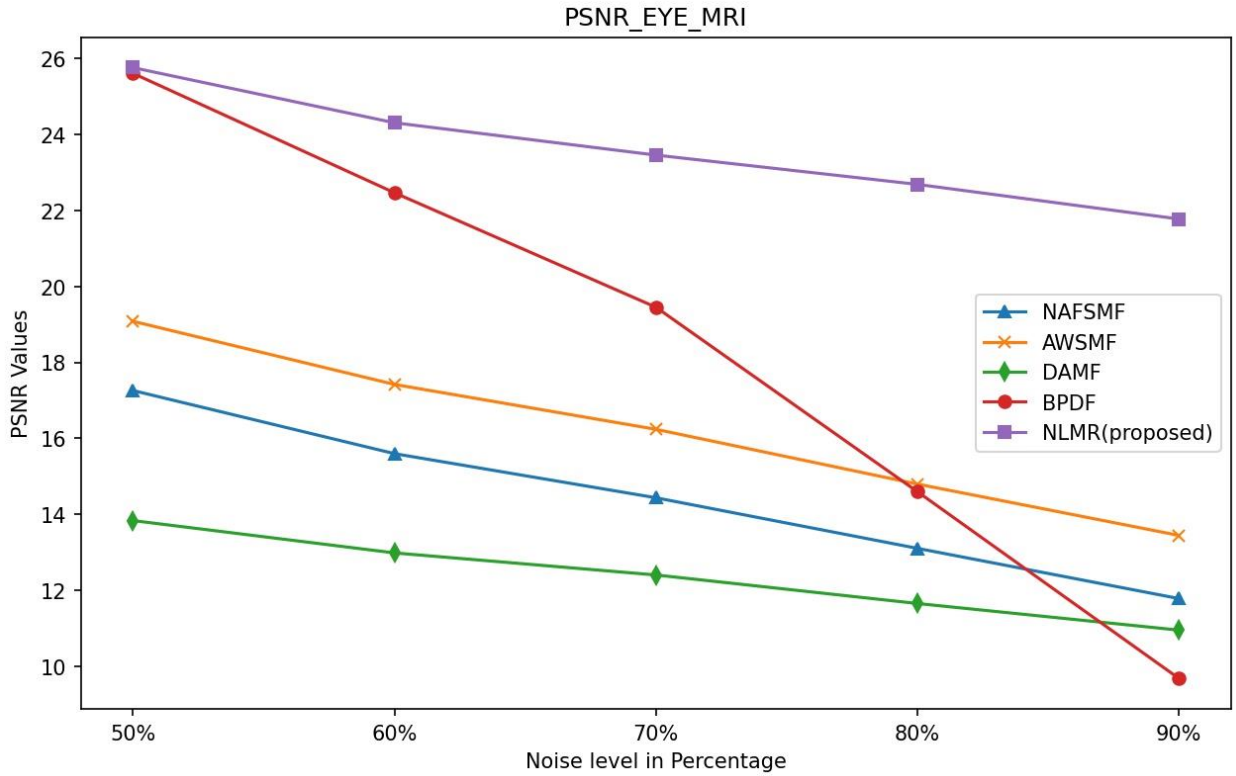


Figure 5.5: Eye MRI image PSNR line chart for all algorithms

In *Figure 5.5* we show a line chart of all algorithms including our proposed algorithm performance. This figure shows that some algorithms like BPDF work very well at low noise. But when increasing the noise level above 50% the performance drop instantly. But our proposed algorithm works perfectly at a high noise level.

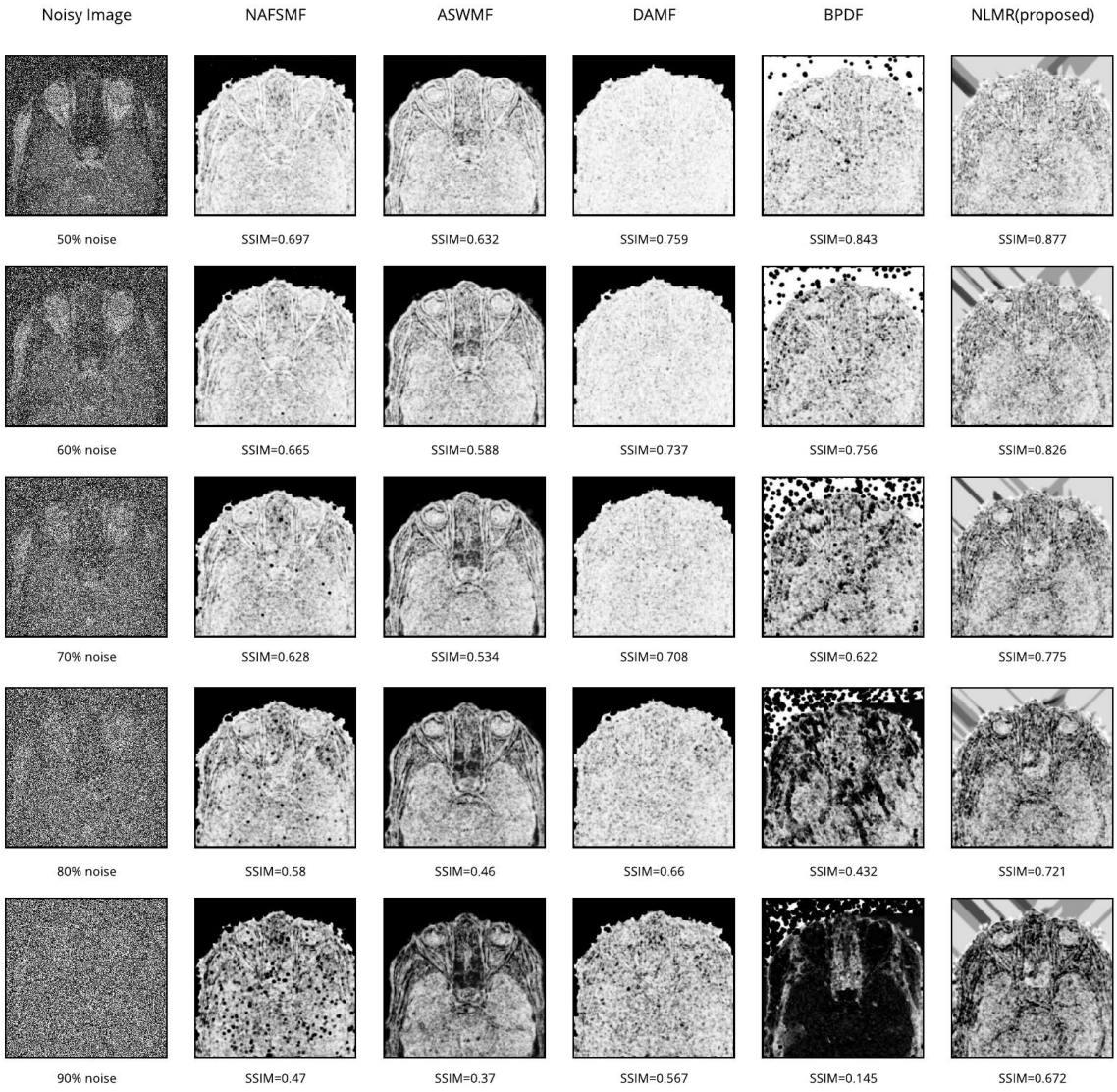


Figure 5.6: Eye MRI image De-noise comparison with SSIM

In *Figure 5.6* we displayed the SSIM or structural similarity result and compare other algorithms with our proposed algorithm. The result of our algorithm and other algorithms for the eye MRI image is shown in *Figure 5.6.*

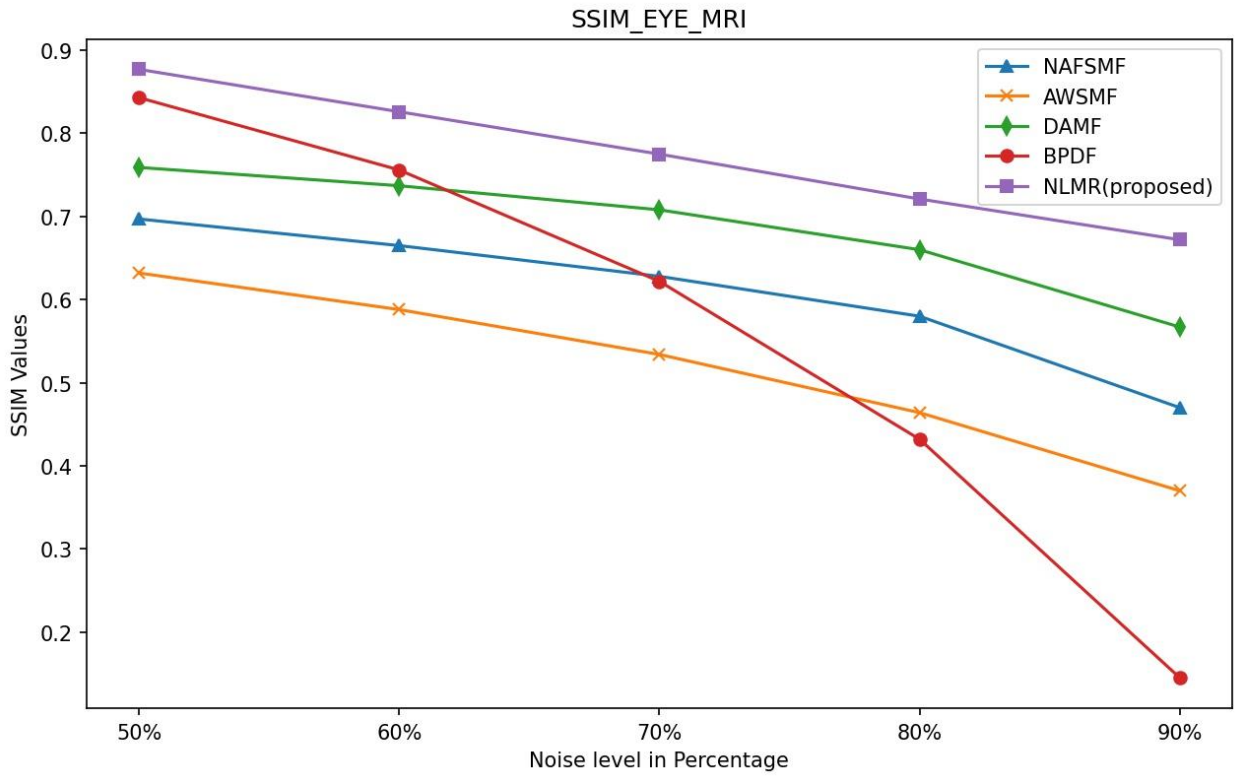
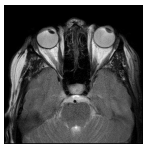


Figure 5.7: Eye MRI image SSIM line chart for all algorithms

In Figure 5.7 we show a line chart of SSIM values of all algorithms including our proposed algorithm. This figure shows that some algorithms like BPDF and other algorithms work very well at low noise. But when increasing the noise level above 70% the performance drop instantly. But our proposed algorithm works very well at a high noise level.

Table 5.1: Eye MRI & SSIM values

MRI Images	Noise in Percentage	Result Types	NAFSMF	AWSMF	DAMF	BPDF	NLMR (proposed)
 (a)Eye MRI	50%	PSNR	17.26	19.08	13.84	25.6	**25.75**
		SSIM	0.697	0.632	0.759	0.843	**0.877**
	60%	PSNR	15.6	17.42	12.99	22.46	**24.3**
		SSIM	0.665	0.588	0.737	0.756	**0.826**
	70%	PSNR	14.44	16.24	12.41	19.45	**23.45**
		SSIM	0.628	0.534	0.708	0.622	**0.775**
	80%	PSNR	13.11	14.8	11.66	14.6	**22.68**
		SSIM	0.58	0.464	0.66	0.432	**0.721**
	90%	PSNR	11.79	13.45	10.96	9.69	**21.77**
		SSIM	0.47	0.370	0.567	0.145	**0.672**

For different algorithms and 50% to 90% noise level, performance measurement function PSNR and SSIM result are listed.

Figure 5.4 presents denoise image for different algorithms including with their PSNR result. In each noise level, NLMR(proposed) ensure that their has no salt and pepper noise remaining in denoise image which is the most important feature in our algorithm. And Other algorithms don't make their image completely noise free. Their PSNR result value visualized in *Figure 5.5* with line chart which is clearly define that NLMR(proposed) works well compare to others algorithms specially in 80% to 90% noise level.

Figure 5.6 and *Figure 5.7* represents the SSIM performance measurement result.

Figure 5.6 represents denoise and real image structural similarity index matrix. SSIM result is similar with their PSNR result. When noise level goes up to 80%, all algorithms performance dropped except NLMR(proposed) in SSIM as well.

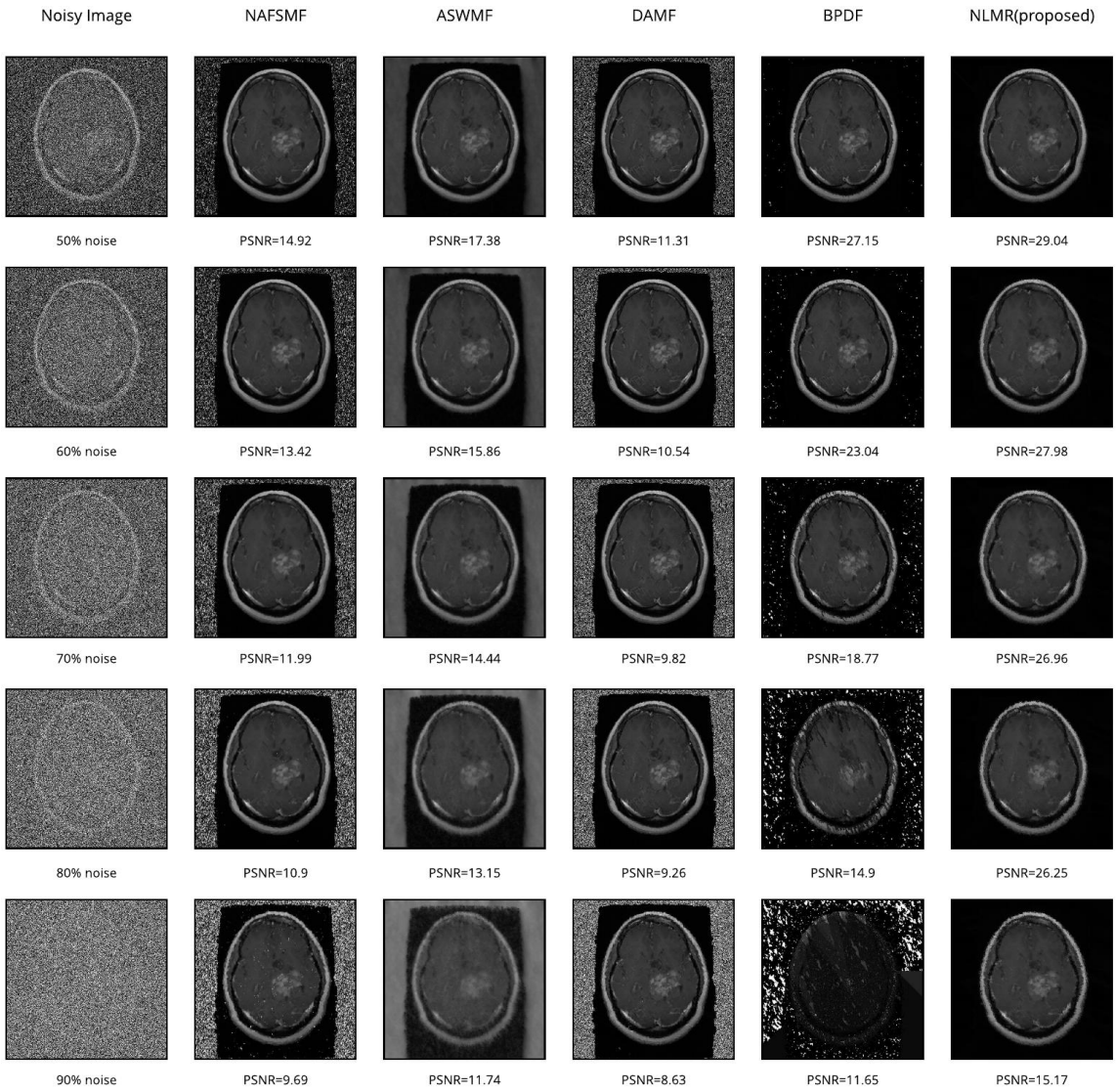


Figure 5.8: Brain MRI image De-noise comparison with PSNR

In *Figure 5.8* we presented the final result and compare other algorithms with our proposed algorithm. In that figure, we also see that some noise reduction algorithms perform well, but they are limited by the fact that they lack recursive functionality. As a result, the algorithms perform admirably on images with no clear black or white regions. However, they are unable to operate adequately with medical digital photos. To overcome this particular problem we proposed our algorithm. Which work recursively and follow a developed algorithm called Non-Local Mean filter. The result of our algorithm and other algorithms for the Brain MRI image is shown in Figure 5.8.

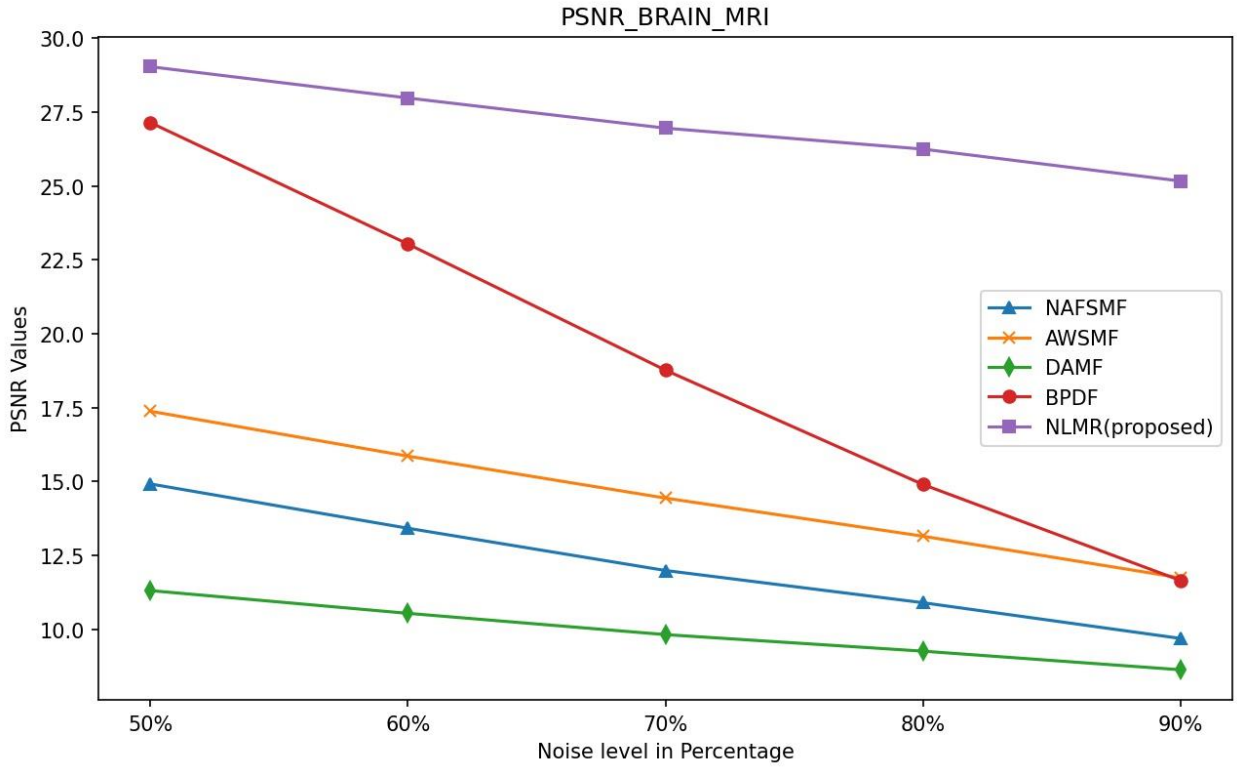


Figure 5.9: Brain MRI image PSNR line chart for all algorithms

In *Figure 5.9* we show a line chart of all algorithms including our proposed algorithm performance. This figure shows that some algorithm like the BPDF algorithm works very well at low noise and other algorithms' performance are not good at low-level noise. But when increase the noise level above 60% the performance drop instantly for the BPDF algorithm. But our proposed algorithm works perfectly at a high noise level as we can see in the chart.

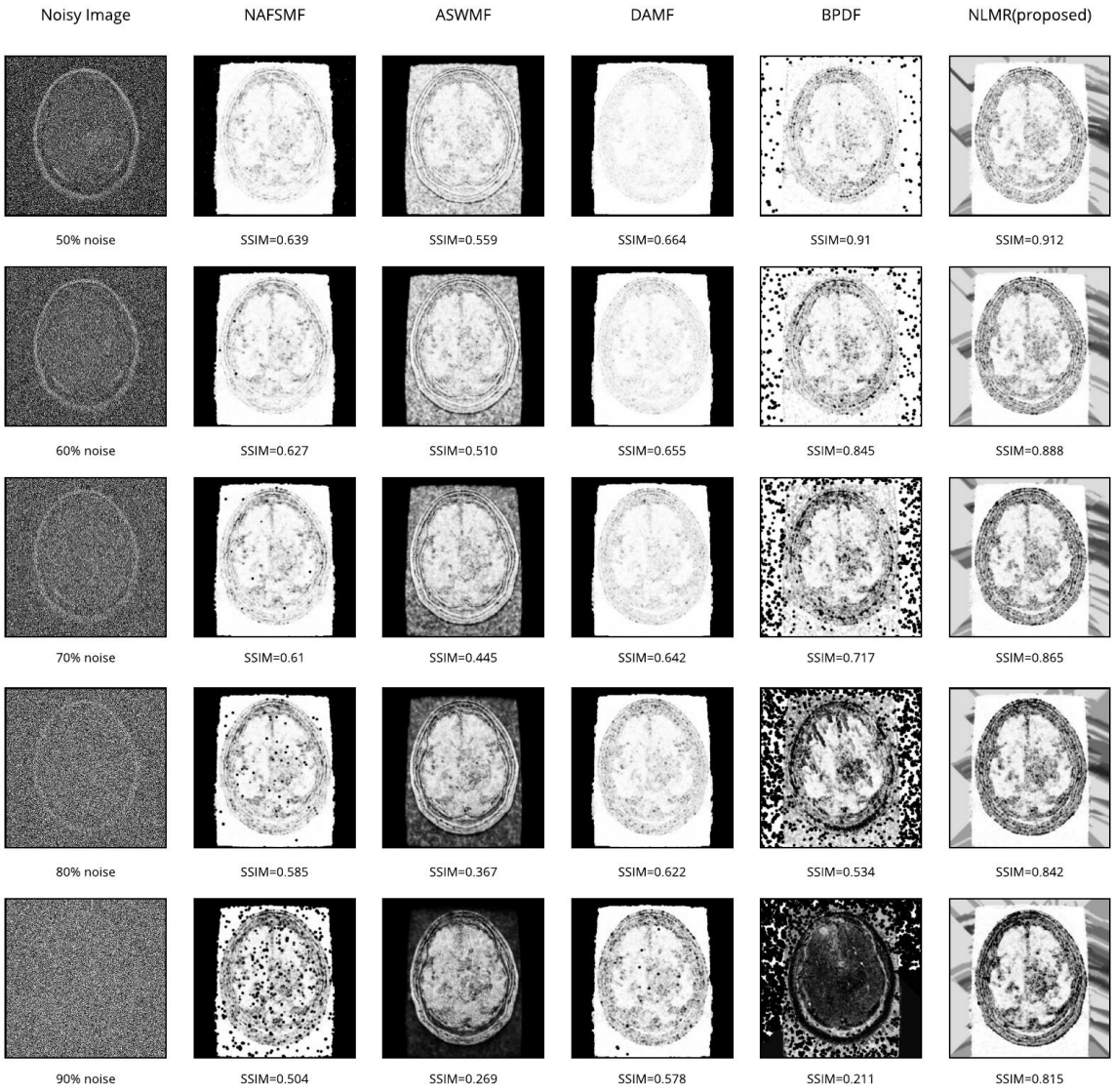


Figure 5.10: Brain MRI image De-noise comparison with SSIM

In *Figure 5.10* we presented the SSIM result of our previous brain MRI images of different types of noise removal algorithms including our proposed algorithm. In this figure first column represent the different types of noisy images. The noise level starts from 50% and ends up at 90% noise level as we working with a high-density noisy removal technique. Other columns have represented the result after nose removal of different types of noise removal algorithms. The name of the algorithms is written above each column. 1st one is NAFSMF which stands for Noise Adaptive Fuzzy Switching Median Filter. This filter is also used as the median filter to remove Salt and Pepper noise from a digital image. 2nd one is ASWMF which stands for Adaptive Switching Weight Mean Filter. The performance of the NAFSMF filter is better than the ASWMF filter. 3rd one is DAMF which is known as Different Applied

Median Filter. As the result, the DAMF filter is much better than the other two filters. 4th one is BPDF which stands for Based on Pixel Density Filter. This filter is work very well among other algorithms. All algorithms mentioned previously work for only Salt and Pepper noise. Some are good at low noise levels some are high-density noise. The last column is represented our proposed algorithm's result. The figure shows the SSIM values of every algorithm. Every SSIM value is different from the other. But our proposed algorithm contains the highest SSIM value. BPDF algorithm is very much close to our algorithm's SSIM values.

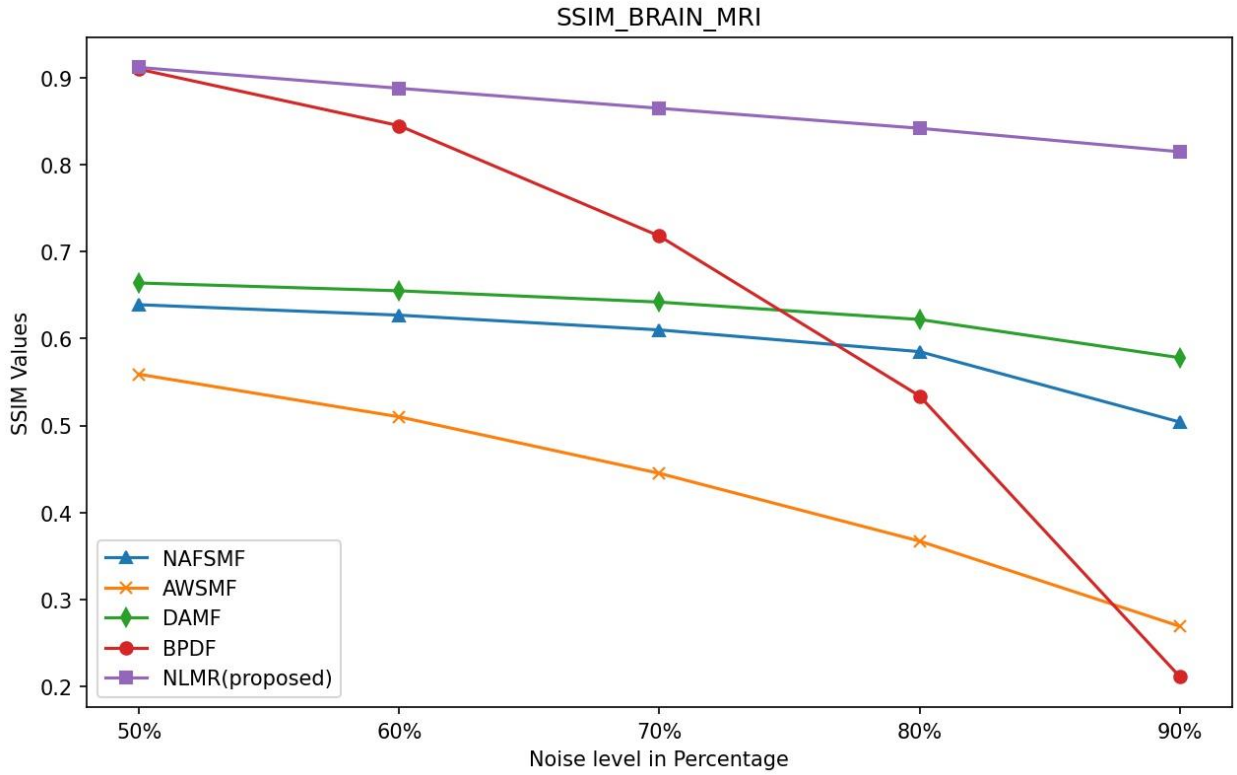
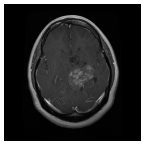


Figure 5.11: Brain MRI image SSIM line chart for all algorithms

In *Figure 5.11* we show a line chart of SSIM values of all algorithms including our proposed algorithm for the Brain MRI image. This figure shows that some algorithms like BPDF and other algorithms work very well at low noise. But when increase the noise level above 70% the performance drop immediately. At 90% of the noise level, the SSIM value of BPDF goes under all of the other algorithms. But our proposed algorithm works very well at the high noise level as we see in the figure.

Table 5.2: Brain MRI PSNR & SSIM values

MRI Images	Noise in Percentage	Result Types	NAFSMF	AWSMF	DAMF	BPDF	NLMR (proposed)
 (b)Brain MRI	50%	PSNR SSIM	14.92 0.639	17.38 0.559	11.31 0.664	27.15 0.91	**29.04** **0.912**
	60%	PSNR SSIM	13.42 0.627	15.86 0.51	10.54 0.655	23.04 0.845	**27.98** **0.888**
	70%	PSNR SSIM	11.99 0.61	14.44 0.445	9.82 0.642	18.77 0.718	**26.96** **0.865**
	80%	PSNR SSIM	10.9 0.585	13.15 0.367	9.26 0.622	14.9 0.534	**26.25** **0.842**
	90%	PSNR SSIM	9.69 0.504	11.74 0.269	8.63 0.578	11.65 0.211	**25.17** **0.815**

For different algorithms and 50% to 90% noise levels, performance measurement function PSNR and SSIM results are listed.

Figure 5.8 presents a denoise image for different algorithms including their PSNR result. In each noise level, NLMR(proposed) ensures that there has no salt and pepper noise remaining in the denoise image which is the most important feature of our algorithm. And Other algorithms don't make their image completely noise-free. Their PSNR result value is visualized in *Figure 5.9* with a line chart which clearly defines that NLMR(proposed) works well compare to other algorithms especially in 80% to 90% noise levels.

Figure 5.10 and *Figure 5.11* represents the SSIM performance measurement result.

Figure 5.10 represents the denoise and real image structural similarity index matrix. SSIM result is similar to their PSNR result. When the noise level goes up to 80%, all algorithms' performance dropped except NLMR(proposed) in SSIM as well.

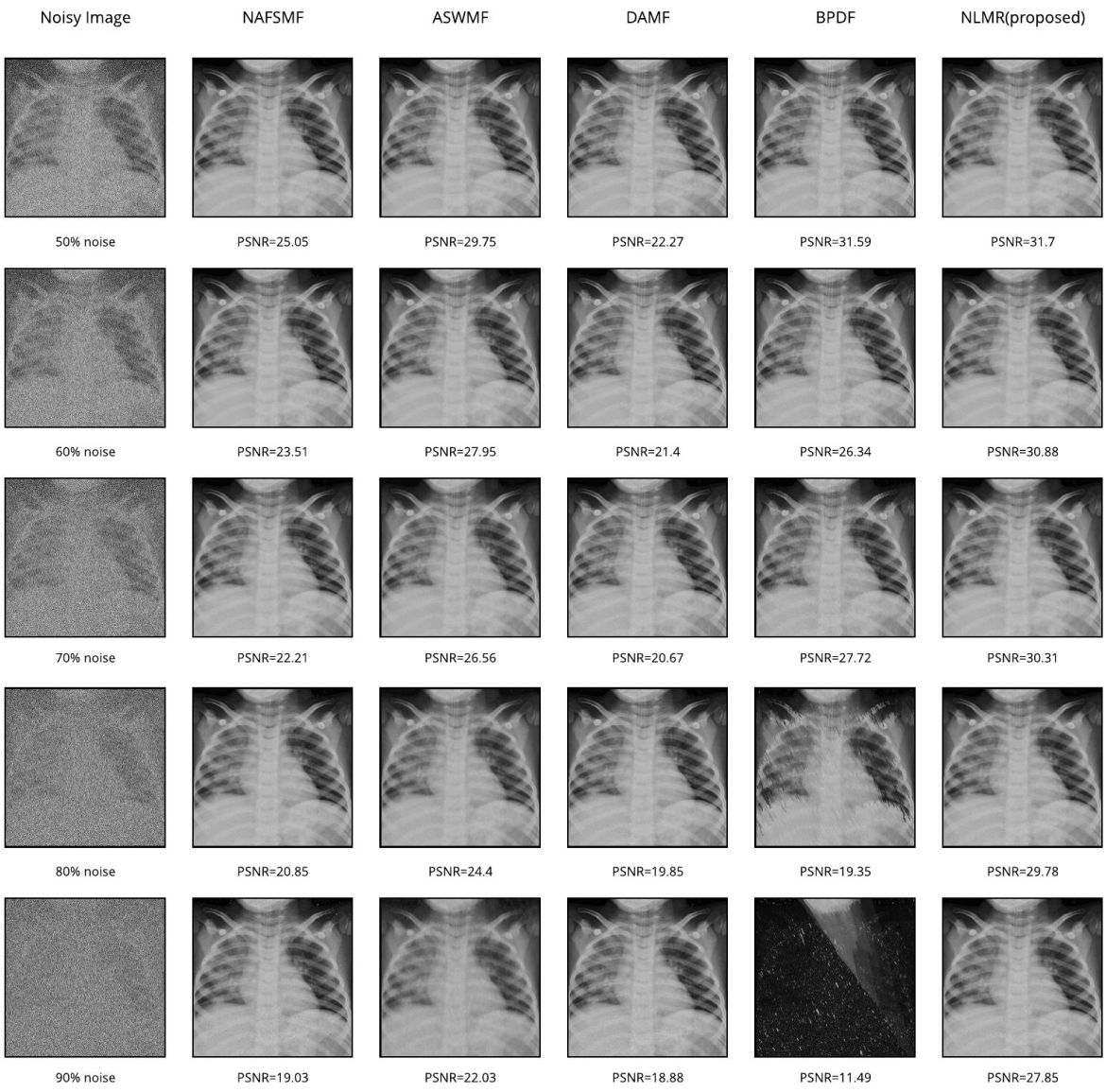


Figure 5.12: Chest image De-noise comparison with PSNR

In *Figure 5.12* we presented the final result and compare other algorithms with our proposed algorithm. In that figure, we also see that some noise reduction algorithms perform well, but they are limited by the fact that they lack recursive functionality. As a result, the algorithms perform admirably on images with no pure black or white regions. However, they are unable to operate adequately with the medical digital image. To overcome this particular problem we proposed our algorithm. Which work recursively and follow a developed algorithm called Non-Local Mean filter. The result of our algorithm and other algorithms for the chest MRI image is shown in *Figure 5.12*.

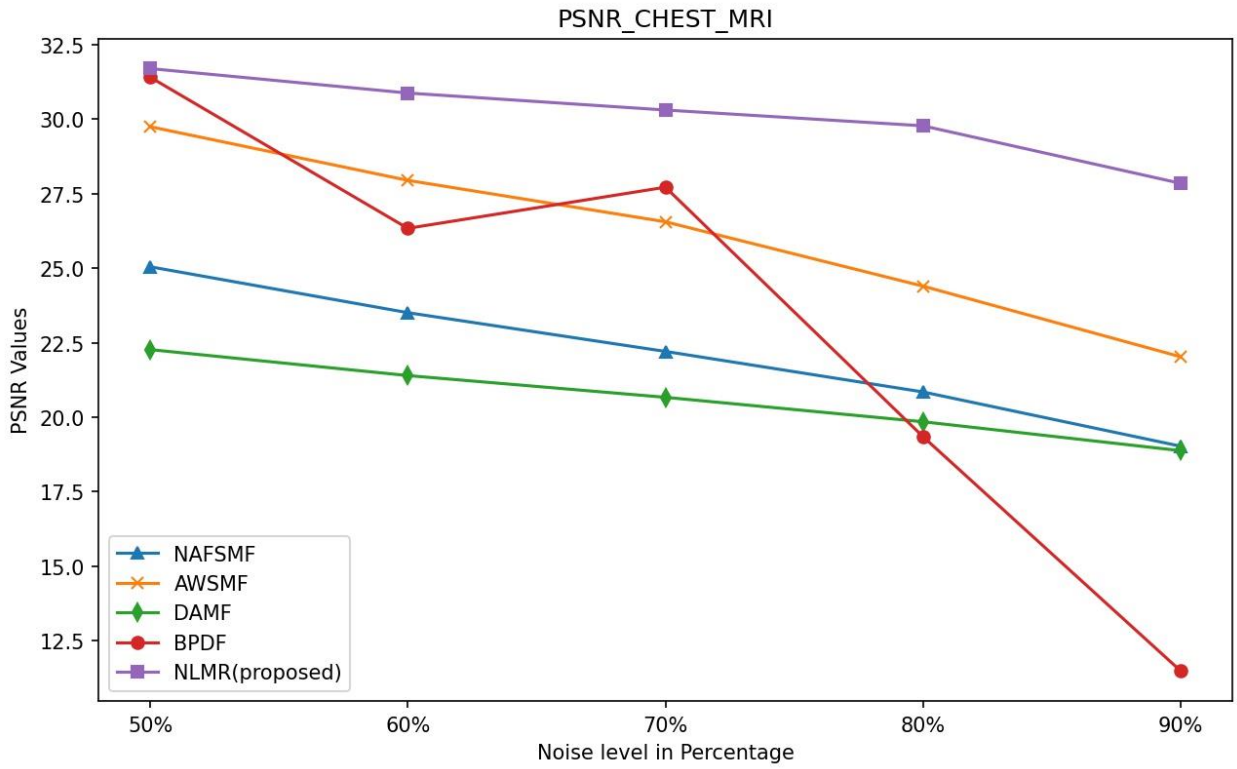


Figure 5.13: Chest MRI image PSNR line chart for all algorithms

In *Figure 5.13* we show a line chart of all algorithms including our proposed algorithm performance. This figure shows that some algorithm like the BPDF algorithm works very well at low noise and other algorithms' performance are not good at low-level noise. But when increase the noise level above 60% the performance drop instantly for the BPDF algorithm. At this noise level, we see a drop in it's performance. However, as seen in the graph, our suggested approach performs flawlessly in high-noise environments.

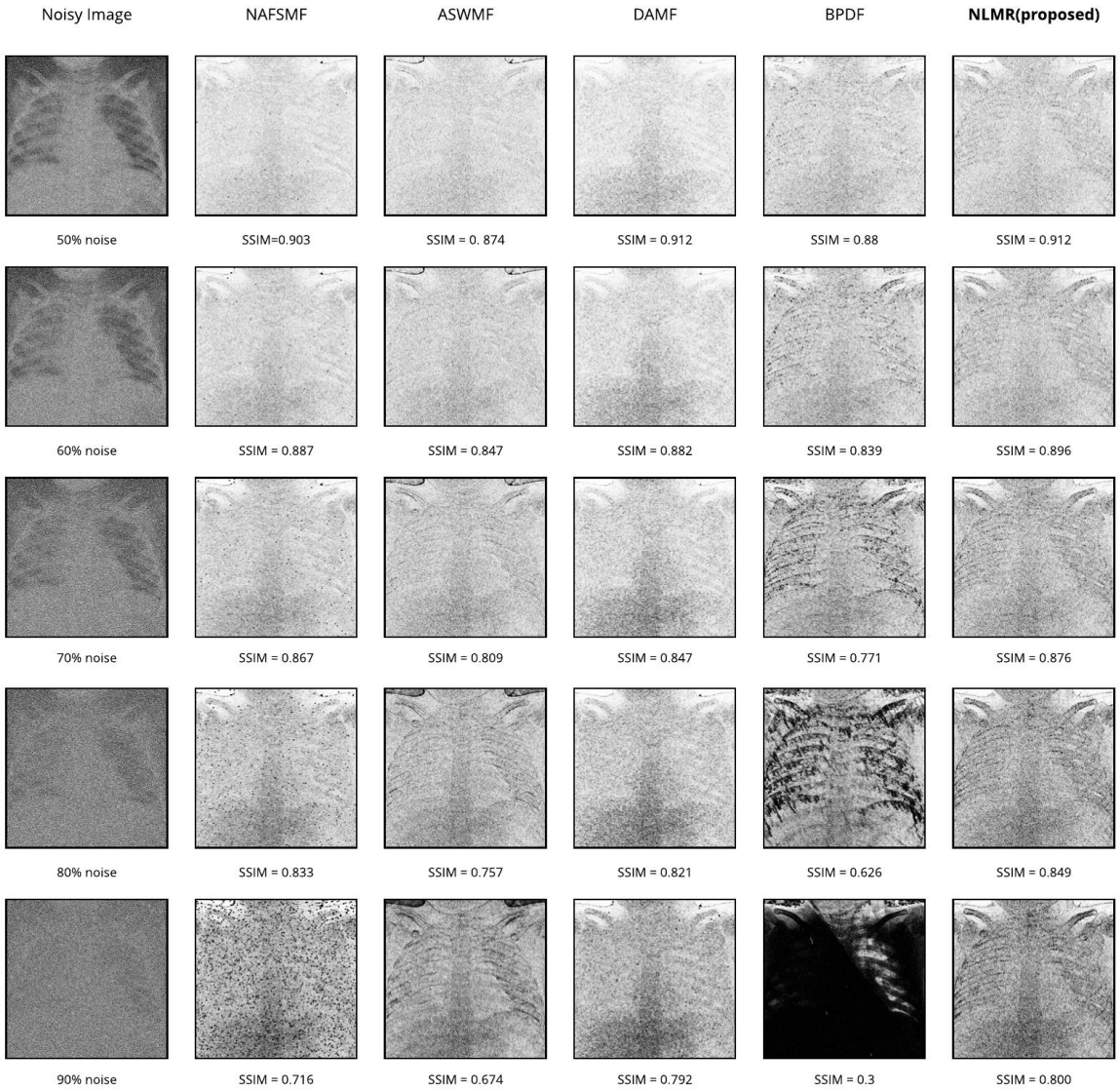


Figure 5.14: Chest MRI image De-noise comparison with SSIM

In *Figure 5.14* we presented the SSIM result of our previous brain MRI images of different types of noise removal algorithms including our proposed algorithm. In this figure first column represent the different types of noisy images. The noise level starts from 50% and ends up at 90% noise level as we working with a high-density noisy removal technique. Other columns have represented the result after nose removal of different types of noise removal algorithms. The name of the algorithms is written above each column. 1st one is NAFSMF which stands for Noise Adaptive Fuzzy Switching Median Filter. This filter is also used as the median filter to remove Salt and Pepper noise from the digital image. 2nd one is ASWMF which stands for Adaptive Switching Weight Mean Filter. The performance of the NAFSMF filter is better than the ASWMF filter. 3rd one is DAMF which is known as Different Applied

Median Filter. As the result, the DAMF filter is much better than the other two filters. 4th one is BPDF which stands for Based on Pixel Density Filter. This filter is work very well among other algorithms. All algorithms mentioned previously work for only Salt and Pepper noise. Some are good at low noise levels some are high-density noise. The last column is represented our proposed algorithm's result. The figure shows the SSIM values of every algorithm. Every SSIM value is different from the other. But our proposed algorithm contains the highest SSIM value. BPDF algorithm is very much close to our algorithm's SSIM values.

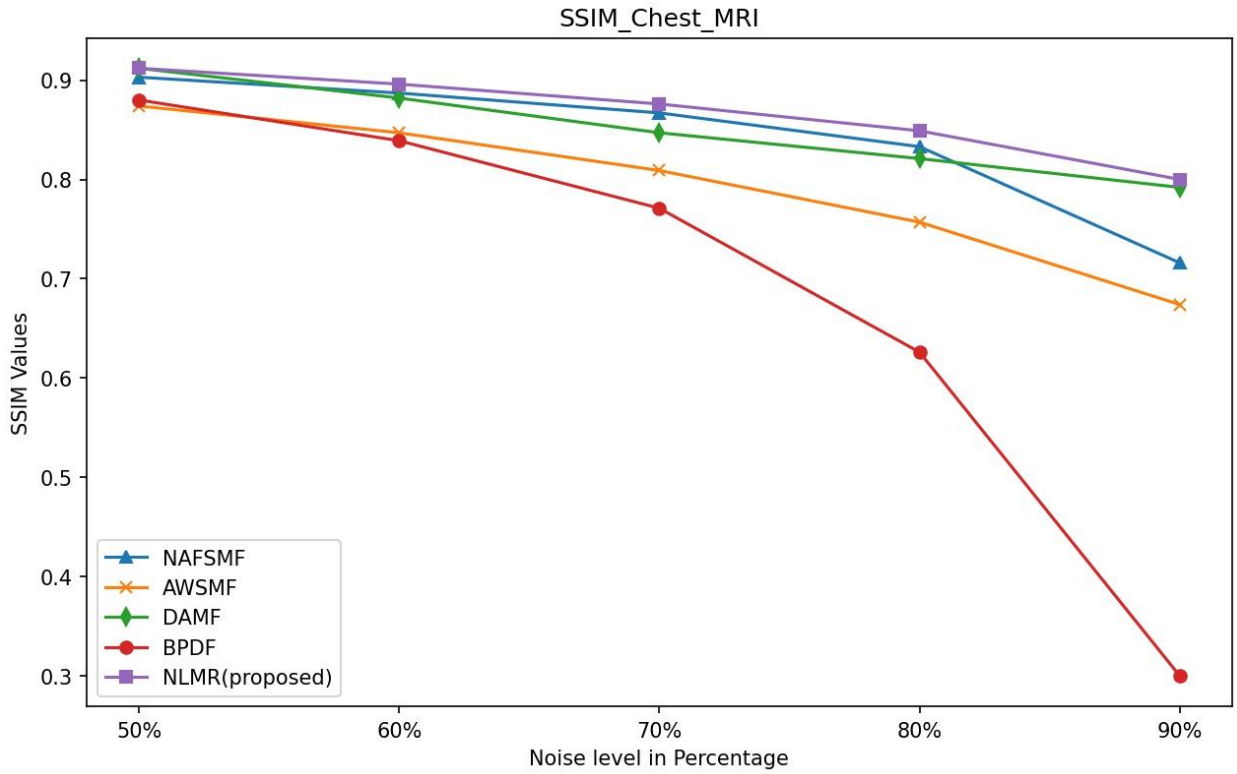


Figure 5.15: Chest MRI image SSIM line chart for all algorithms

In *Figure 5.15* we show a line chart of SSIM values of all algorithms including our proposed algorithm for the chest MRI image. This figure shows that some algorithms like BPDF and other algorithms work very well at low noise. But when increase the noise level above 70% the performance drop immediately. At 90% of the noise level, the SSIM value of BPDF and other algorithms goes under all of the other algorithms. Our proposed algorithm works very well at the high noise level as we see in the figure. For this image, other algorithms' performance is very likely to our proposed algorithm as we can see in the figure or line chart.

Table 5.3: Chest MRI PSNR &SSIM values

MRI Images	Noise in Percentage	Result Types	NAFSMF	AWSMF	DAMF	BPDF	NLMR (proposed)
 (c)Chest MRI	50%	PSNR SSIM	25.05 0.903	29.75 0.874	22.27 0.912	31.41 0.88	**31.7** **0.912**
	60%	PSNR SSIM	23.51 0.874	27.95 0.847	21.4 0.882	26.34 0.839	**30.88** **0.896**
	70%	PSNR SSIM	22.21 0.84	26.56 0.809	20.67 0.847	27.72 0.771	**30.31** **0.876**
	80%	PSNR SSIM	20.85 0.81	24.4 0.757	19.85 0.821	19.35 0.626	**29.78** **0.849**
	90%	PSNR SSIM	19.03 0.716	22.03 0.674	18.88 0.792	11.49 0.3	**27.85** **0.8**

For different algorithms and 50% to 90% noise levels, performance measurement function PSNR and SSIM results are listed.

Figure 5.12 presents a denoise image for different algorithms including their PSNR result. In each noise level, NLMR(proposed) ensures that there has no salt and pepper noise remaining in the denoise image which is the most important feature of our algorithm. And Other algorithms don't make their image completely noise-free. Their PSNR result value is visualized in *Figure 5.13* with a line chart which clearly defines that NLMR(proposed) works well compare to other algorithms especially in 80% to 90% noise levels.

Figure 5.14 and *Figure 5.15* represents the SSIM performance measurement result.

Figure 5.15 represents the denoise and real image structural similarity index matrix. SSIM result is similar to their PSNR result. When the noise level goes up to 80%, all algorithm's performance dropped except NLMR(proposed) in SSIM as well.

6 CONCLUSION & FUTURE WORK

6.1 CONCLUSION

A non-local mean recursive filtering method is proposed in this paper. The noise-removing filter technique removes high-density Salt and Pepper noise from the medical image. High-density Salt and Pepper noise mean the noise level begins at 50% and end up at 90% of the noise level. The algorithm followed a method known as recursive until there was no Salt and Pepper in the medical image. The recursive function takes input which is the output of the previous iteration. The Standard mean and Adaptive median filter algorithms cannot remove high-density noise. Most noise removal filters cannot remove all Salt and Pepper noise from the medical image because the medical image is different from the natural image captured with a digital camera. And also, medical images contain pure black and white region in them. Our Non-Local Mean Recursive Filter algorithm can remove high-density noise up to 90% of Salt and Pepper noise from the medical image. The computation time of the algorithm is defended on the noise level. More noise levels required more computation time.

6.2 FUTURE PLANS

This paper discussed how to remove Salt and Pepper noise from a medical image. There are a lot of techniques to detect Salt and Pepper noise and remove those noise from that image. For all those calculations, it took several times.

Our plan is that minimizes the computation time for removing noise. We plan to apply machine learning to predict the pixel intensity level of that noisy pixel. Machine learning results are predictions[34], [35].

To apply the machine learning method or create a machine learning model to remove Salt and Pepper noise from a medical image, first, we need to create a dataset containing 256 classes. Why 256 classes? The answer is simple: an image pixel intensity range ranges from 0 to 255, totaling 256 intensity levels. A noisy pixel can contain any intensity level of those. That's why there will be 256 classes.

The number of features can be different. It depends on how many rows and columns we selected as the window. For example, if we choose a 5×5 window, the number of features will be 25. If we decide on a 7×7 window, the number of features will be 49. If we select an 11×11 window, then the number of features will be 121, and so on.

This model will be a multi-class classification model because we denote multiple classes as the intensity level of the pixel.

The model will take input pixel intensity levels where those intensity levels will be a mixture of noisy (0 or 255) and non-noisy pixel values. And the output will be a number based on those intensity values.

If we succeed in this plan, our program will much faster remove Salt and Pepper noise from medical images.

REFERENCES

[1] D. Thanh, N. Hien, K. Palanisamy, and S. Prasath, “Adaptive Switching Weight Mean Filter for Salt and Pepper Image Denoising,” *Procedia Comput. Sci.*, vol. 171, pp. 292–301, Jun. 2020.

[2] K. Toh and N. A. Mat Isa, “Noise Adaptive Fuzzy Switching Median Filter for Salt-and-Pepper Noise Reduction,” *Signal Process. Lett. IEEE*, vol. 17, pp. 281–284, Apr. 2010.

[3] U. Erkan, L. Gökrem, and S. Enginoğlu, “Different applied median filter in salt and pepper noise,” *Comput. Electr. Eng.*, vol. 70, Aug. 2018.

[4] U. Erkan and L. Gökrem, “A new method based on pixel density in salt and pepper noise removal,” *TURKISH J. Electr. Eng. Comput. Sci.*, vol. 26, Jan. 2018.

[5] D. Thanh, L. Thanh, S. Prasath, and U. Erkan, *An Improved BPDF Filter for High Density Salt and Pepper Denoising*. 2019.

[6] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda, “Photographic tone reproduction for digital images,” *Proc. 29th Annu. Conf. Comput. Graph. Interact. Tech. SIGGRAPH ’02*, pp. 267–276, 2002.

[7] G. Katti, S. Arshiya Ara, and A. Shireen, “Magnetic Resonance Imaging (MRI) – A Review,” *Int. J. Dent. Clin.*, vol. 3, no. 1, pp. 65–70, Mar. 2011.

[8] Z. T. Al-Sharify, T. A. Al-Sharify, N. T. Al-Sharify, and H. Y. Naser, “A critical review on medical imaging techniques (CT and PET scans) in the medical field,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 870, no. 1, p. 12043, Jul. 2020.

[9] S. B. Desai, V. C. Shah, D. J. Javri, and P. Rao, “Neurocysticercosis: comparison of CT scan and MRI,” in *Proceedings of the XIV Symposium Neuroradiologicum*, 1991, pp. 578–579.

[10] J. Sander, B. D. de Vos, and I. Išgum, *Autoencoding low-resolution MRI for semantically smooth interpolation of anisotropic MRI*, vol. 78. 2022.

[11] D. B. Nissman and B. M. Dale, *Magnetic Resonance Imaging Principles and Techniques*. 2013.

[12] M. Azad, M. Hasan, and M. K, “Color Image Processing in Digital Image,” *Int. J. New Technol. Res.*, vol. 3, no. 3, p. 263334, 2017.

[13] S. Iglesias-Rey, A. Castillo-Lopez, C. Lopez-Molina, and B. De Baets, *On the Role of Context-Awareness in Binary Image Comparison*. 2022.

[14] P. Pullan, K. Mehta, M. Arora, and V. Niranjan, *Noise reduction from grayscale images*. 2020.

[15] H. Herrmann and H. Bucksch, “RGB color spectrum (US),” in *Dictionary Geotechnical Engineering/Wörterbuch GeoTechnik*, 2014, pp. 1116–1116.

[16] S. Hong, *RGB color bits*, vol. 18. 2015. doi: 10.1145/2702613.2744697.

[17] S. Alvar and I. Bajic, *Practical Noise Simulation for RGB Images*. 2022.

[18] K. Inoue, K. Hara, and K. Urahama, “RGB color cube-based histogram specification for hue-preserving color image enhancement,” *J. Imaging*, vol. 3, no. 3, p. 24, Jul. 2017.

[19] K. Inoue, M. Jiang, and K. Hara, “Hue-preserving saturation improvement in rgb color cube,” *J. Imaging*, vol. 7, no. 8, p. 150, Aug. 2021.

[20] J. Yuan, “An improved variational model for denoising magnetic resonance images,” *Comput. Math. with Appl.*, vol. 76, no. 9, pp. 2212–2222, Nov. 2018.

[21] J. V. Manjón, J. Carbonell-Caballero, J. J. Lull, G. García-Martí, L. Martí-Bonmatí, and M. Robles, “MRI denoising using Non-Local Means,” *Med. Image Anal.*, vol. 12, no. 4, pp. 514–523, Aug. 2008.

[22] S. Khandelwal, “Noise Models,” in *Advanced SPICE Model for GaN HEMTs (ASM-HEMT)*, 2022, pp. 125–130.

[23] M. Radak, “Gaussian noise removal with proper filter.” Jan. 20, 2022.

[24] J. Al-Azzeh, B. Zahran, and Z. Alqadi, “Salt and pepper noise: Effects and removal,” *Int. J. Informatics Vis.*, vol. 2, no. 4, pp. 252–256, Jul. 2018.

[25] L. Hou, H. Liu, Z. Luo, Y. Zhou, and T. K. Truong, *Image deblurring in the presence of salt-and-pepper noise*, vol. 2017-Septe. 2018.

[26] A. Buades, B. Coll, and J. M. Morel, “A Review of Image Denoising Algorithms, with a New One,” *http://dx.doi.org/10.1137/040616024*, vol. 4, no. 2, pp. 490–530, Jul. 2006.

[27] H. Zaini and Z. Alqadi, “High Salt and Pepper Noise Ratio Reduction,” *Int. J. Comput. Sci. Mob. Comput.*, vol. 10, no. 9, pp. 88–97, Sep. 2021.

[28] H. Lin, Y. Zhuang, Y. Huang, X. Ding, X. Liu, and Y. Yu, *Noise2Grad: Extract Image Noise to Denoise*. 2021.

[29] A. I. Novikov and A. V. Pronkin, “Methods for image noise level estimation,” *Comput. Opt.*, vol. 45, no. 5, pp. 713–720, Sep. 2021.

[30] B. Perumal, R. S. Devi, and M. P. Rajasekaran, “Extermination methods of image noises: a review,” *3C Tecnol. innovación Apl. a la pyme*, pp. 243–259, Nov. 2021.

[31] A. Márques and A. Pardo, *Implementation of non local means filter in GPUs*, vol. 8258 LNCS,

no. PART 1. 2013.

[32] G. Raju, F. F. Wahid, and K. P. Shareekhath, *Modified non-local means filtering*. 2015.

[33] S. Van Der Walt *et al.*, “Scikit-image: Image processing in python,” *PeerJ*, vol. 2014, no. 1, p. e453, Jun. 2014.

[34] A. M. Abou-Zamzam, “Learning from machine learning,” *J. Vasc. Surg.*, vol. 75, no. 1, p. 286, Jan. 2022.

[35] J. J. Squiers *et al.*, “Machine learning analysis of multispectral imaging and clinical risk factors to predict amputation wound healing,” *J. Vasc. Surg.*, vol. 75, no. 1, pp. 279–285, Jul. 2022.