**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**BELAGAVI, KARNATAKA-590 018**



**A PROJECT REPORT**

on

**“IMPULSE NOISE REMOVAL FROM MR IMAGES FOR RADIOSURGERY APPLICATIONS”**

***Submitted as partial fulfillment required for the award of***

**Bachelor of Engineering**

in

**ELECTRONICS & COMMUNICATION ENIGINEERING**

Submitted by

BUVAN BS 1SK17EC004

DEVIKA U 1SK17EC007

MALLIKARJUN GOKAK 1SK17EC022

UME ATHIBA 1SK17EC044

Under the guidance of

**Dr. RANGARAJU H G**

Dept. of Electronics and Communication Engineering



**Govt. Sri Krishnarajendra Silver Jubilee Technological Institute**

K. R. Circle, Bengaluru-560001

2020-2021

**GOVERNMENT SRI KRISHNARAJENDRA SILVER**

**JUBILEE TECHNOLOGICAL INSTITUTE**

K. R. Circle, Bengaluru-560001

**Department of Electronics and Communication Engineering**



**CERTIFICATE**

This is to certify that **BUVAN BS (1SK17EC004), DEVIKA U (1SK17EC007)**, **MALLIKARJUN GOKAK (1SK17EC022)** and **UME ATHIBA (1SK17EC044)** have satisfactorily completed the **PROJECT WORK** titled **“REMOVAL OF IMPULSE NOISE FROM MR IMAGES FOR RADIOSURGERY APPLICATIONS”** prescribed by the Visvesvaraya Technological University, Belagavi as partial fulfillment for the award of Bachelor of Engineering in Electronics and Communication Engineering in the Academic year 2020 – 2021.

Signature of the Guide Signature of the H.O.D.

                                                     EXAM EVALUATORS

Names of the Examiners                                                                                Signature

1. Dr. Rangaraju H G

2. Dr. Leelavathi G \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**TABLE OF CONTENTS**

**CONTENTS Page No.**

ABSTRACT…………………………………………………………………….. I

LIST OF FIGURES …………..………………………………………………... II

LIST OF TABLES…………..…………………………………………………. III

**Chapter 1: INTRODUCTION 1-5**

1.1 Domain Concept 1

1.2 Types of Noises 1

1.2.1 Gaussian Noise 2

1.2.2 Impulse Noise 2

1.2.3 Poisson Noise 2

1.2.4 Speckle Noise 3

1.2.5 Poisson Gaussian Noise 3

1.3 Impulse Noise 3

1.4 Impulse De-Noising Techniques 4

1.4.1 Median Filter 5

1.4.2 Low Complex Noise Reduction (LCNR) 5

1.4.3 Decision-Tree-based De-Noising method (DTBDN) 5

1.5 Applications 5

**Chapter 2: LITERATURE SURVEY 6-15**

2.1 Introduction 6

2.2 Review on Recent Literature 6

2.3 Summary 13

**Chapter 3: METHODOLOGY 16-23**

3.1 Introduction 16

3.2 Objectives 16

3.3 Methodology 17

3.3.1 Partitioning 17

3.3.2 Edge Detection 18

3.3.3 Disorder Analysis 19

3.3.4 Noisy Pixel Checking 20

3.3.5 Similarity Checking 20

3.3.6 Restoration 20

3.3.7 Image Formation 23

**Chapter 4: BLOCK DIAGRAM AND SOFTWARE REQUIRMENTS 24-32**

4.1 Introduction24

4.2 Block Diagram 24

4.3 Software Requirements 26

4.3.1 Introduction to MATLAB 26

4.3.2 What is MATLAB 26

4.3.3 MATLAB System parts 27

4.3.3.1 MATLAB Language 27

4.3.3.2 MATLAB Working Environment 27

4.3.3.3 Handle Graphics 27

4.3.3.4 MATLAB Mathematical Function Library 28

4.3.3.5 MATLAB Application Program Interface (API) 28

4.3.4 Starting MATLAB 28

4.3.5 To create a simple GUI using guide 29

4.3.5.1 Align the Components 31

4.3.5.2 Save the Layout 31

4.3.6 Code behavior 31

**Chapter 5: IMPLEMENTATION 33-35**

5.1 Implementation Using MATLAB 33

5.1.1 Implementation of the MATLAB code 33

5.2 Implementation of GUI 34

**Chapter 6: RESULTS AND DISCUSSION 36-43**

6.1 MATLAB Code Analysis 36

6.2 MATLAB for GUI analysis 39

6.3 Comparison with State-of-the Art Methods 41

**REFERENCES 44-45**

**ABSTRACT**

In recent years image processing techniques are used as a tool to improve detection and diagnostic capabilities in medical applications. Medical applications have been so much affected by these techniques that some of them are embedded in medical instruments such as MRI, CT and other medical devices. Among these techniques, medical image enhancement algorithms play an essential role in removal of the noise which can be produced by medical instruments and during image transfer. Impulse noise is a major type of noise, which is produced during medical operations, such as MRI, CT, and angiography, by their image capturing devices.

This project proposes an accurate algorithm for removal of impulse noise in medical images, especially for MR-Images. All image blocks are divided into three categories of edge, smooth, and disordered areas. A different reconstruction method is applied to each category of blocks for the purpose of noise removal. It preserves edges and other artifacts as well as is less complex.

**LIST OF FIGURES**

2.1 A 5×5 window is divided into 3×3overlapped 7

Sub-windows (Quadrants), arrows represent four directions

2.2 An overview of the proposed technique 11

2.3 A 3 x 3 mask centered on pi,j 12

2.4 The dataflow of DTBDM 13

3.1 (a) General structure of the proposed algorithm, 17

(b) A sample noisy image, (c) the de-noised image

3.2 Pixel numbers and edge directions in Type2-Edge-Detection window 19

3.3 Dataflow of edge-preserving image filter 21

3.4 Eight directional differences of DTBDM 22

4.1 Block Diagram for Impulse Noise removal in MR Images 24

4.2 The graphical interface to the MATLAB workspace 28

4.3 GUI guide starting page 29

4.4 Basic GUI image 29

4.5 Names in the component palette are displayed 30

4.6 Windows in GUI guide 30

5.1 GUI before execution 35

5.2 GUI after execution, for noise density 25% 35

6.1 Output for noise density, ND = 15% 36

6.2 Output for noise density, ND = 20% 37

6.3 Four sample original images and their noisy versions 38

6.4 Denoising of images using the proposed method 39

6.5 GUI before execution 39

6.6 GUI after execution, for noise density ND = 5% 40

6.7 GUI after execution, for noise density ND = 20% 41

6.8 Denoising of images of Fig. 6.3 using standard median filters 42

6.9 Visual quality comparison of the proposed method with [11] and [12] 43

**LIST OF TABLES**

Table 2.1 Comparison between existing filtering methods and 14

proposed method

Table. 6.1 PSNR and SSIM values for different noise densities 38

Table 6.2 Comparison between denoised results in terms of 42

PSNR (dB) of image size 256 x 256 for different noise densities

Table 6.3 Comparison between denoised results in terms of 42

PSNR (dB) of image size 512 x 512 for different noise densities

**CHAPTER 1**

**INTRODUCTION**

**1.1 DOMAIN CONCEPT**

**Image processing** is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image.

A **digital image** is a representation of a two-dimensional image as a finite set of digital values, called picture elements or pixels. Digital image processing is a part of digital signal processing. Digital image processing has many significant advantages over analog image processing. Image processing allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing of images.

Image de-noising is a vital image processing task i.e., as a process itself as well as a component in other processes. Medical images are affected by different types of noise such as Gaussian, Impulse, Speckle and Poisson noises. Presence of noise can produce misleading artifacts in the visual representation of the interior of human organs. Noise can be produced from different types of medical image instruments such as MR, CT, X-ray, ultrasound, etc. Imperfect instruments, problems with the data acquisition process, thermal energy of heat inside the image sensors and discrete sources of radiation can all corrupt the data of interest. Therefore, Image De-noising techniques are necessary to prevent this type of corruption from digital images especially medical images. The important property of a good image de-noising model is that it should completely remove noise as far as possible as well as preserve edges.

**1.2 Types of Noises**

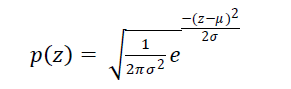
Digital images can be corrupted due to the presence of various types of noises. Filters are available to correct these errors. But to know the working of these filters it is necessary to know about different types of images, also known as noise models. Learning about the sources of these noises and methods of measuring is important too.

Noises in digital images are usually produced due to heat, amplifier gain, photon to electron conversion, pixel defects, and electronics such as Charged Coupled Devices *(CCD)* and Complementary Metal Oxide Semiconductor *(CMOS).* The noises occurred due to the above causes can be scrutinized using the techniques like Point Spreading Function *(PSF)* and Modulation Transfer Function *(MTF).* The characterization of noise can be done using Probability Density Function *(PDF)* and Histogram. Following are few noise models that were analyzed as an evolution of Digital Image processing.

* Gaussian noise
* Impulse valued noise
* Poisson noise
* Speckle noise
* Poisson- Gaussian noise

**1.2.1 Gaussian noise:**

Gaussian noise can also be called as electronic noise and thermal noise usually observed in amplifiers, detector and other electronic devices. Natural sources, discrete nature of radiation in electronic devices which heat up quickly stand as sources of this noise. The noise occurred is independent in each pixel and in terms of signal intensity. The probability density function of Gaussian noise is equal to normal distribution of the image intensity. Gaussian noise can be identified as undesirable blurring of fine scale image and pixel edges.



Where z= gray value, σ= standard deviation, μ= mean.

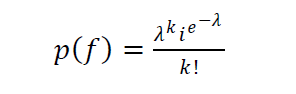
**1.2.2 Impulse noise:**

Impulse noise occurs due to the drop in original data values and hence it is called as data drop noise. It can also be named as salt and pepper noise as the corrupted images look like salt and pepper has been spilt on it. In this type of noise only a few pixels are corrupted. It can simply be explained as few of the dark pixels turn white and few of the white pixels turn dark. It is an error that occurs when data is converted from analog to digital or during the transmission, identifying it as bit error.

* + 1. **Poisson noise:**

Poisson noise is observed in pictures which usually involve electromagnetic rays such as x-rays, gamma rays, visible rays etc. Due to this the noise is statistical in nature. During the process of x-ray scanning in medical field, the rays are projected into the patient’s body. When the rays are emitted from the source of x-rays or gamma rays, there will be a fluctuation in the photons. This causes a spatial randomness in the resultant image, which we can identify as Poisson noise or photon noise or quantum noise or shot noise. Shot noise’s root-mean-square value is proportional to the square of image intensity.

Photon noise obeys the Poisson distribution function.



The changes in applied voltage and current might affect the nature of the noise drastically.

* + 1. **Speckle noise:**

Speckle noise is multiplicative in nature and is common acquisition in lasers, radars and acoustics. It exists in an image in co-ordinance with Gaussian noise. It is omnipresent, which is one of the limitations of remote sensing images. It follows Gaussian distribution function which is as follows.



* + 1. **Poisson Gaussian noise:**

Poisson- Gaussian noise is a prevalent noise in magnetic resonance imaging technique and in low-dose x-ray models. Poisson noise is added to the image from the electronic source while the other noises which corrupt the image from the sensor which results in Poisson-Gaussian noise.

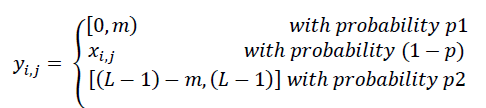
It affects the strength of the pixels in turn affecting the strength of the MRI image which might make analyzing difficult.

**1.3 IMPULSE NOISE**

**Impulse noise** occurs due to the drop in original data values and hence it is called as data drop noise. Impulse Noise can be divided into two types:

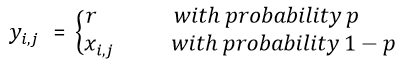
* Fixed Valued Impulse Noise.
* Random Valued Impulse noise.

**Fixed Valued Impulse** noise can also be named as **salt and pepper noise** as the corrupted images look like salt and pepper has been spilt on it. During image acquisition and transmission process noise is generated and produces effects in image quality. For 8-bit gray scale noisy image, typical value 255 assign for salt noise and 0 assign for pepper noise. This type of noise involves isolated pixels being either turned on (white, hence salt) or off (black, or pepper), as a result of impulse noise often caused by errors in data transmission.



𝑚 is a constant value, 𝑥𝑖, and 𝑦𝑖,𝑗 are original and noisy value of the pixel respectively, and 𝑝1 and 𝑝2 are the probabilities that the pixel gets the noisy value, where 𝑝= 𝑝1+ 𝑝2. Also, (𝐿−1) shows the maximum possible intensity value of a pixel. If 𝑚=0 then the induced noise is salt and pepper noise.

In **Random Valued Impulse Noise**, noise is dispersed uniformly. For gray scale images, a pixel may get noisy with a probability of 𝑝, where a value in the range of 0 to (𝐿 − 1) randomly replaces the original pixel. In, the below equation, 𝑟 is a random value, x𝑖, 𝑗 and y𝑖, are the original and new values of the pixel respectively.



Hence, we are proposing an accurate algorithm for removal of random valued impulse noise in medical images, especially for MR-Images.

**1.4 IMPULSE DE-NOISING TECHNIQUES**

Impulse noise is of interest to many researchers as a type of common noise. There are two types of filtering - spatial domain filtering and transform domain filtering. The spatial domain ﬁltering can be further divided into linear (such as Wiener ﬁlters and Mean filters) and nonlinear ﬁlters (such as median ﬁlters). Transform domain ﬁltering includes many classic methods, such as Fourier transform, wavelet transform, threshold function, curvelet, and contourlet[1]. The benefits of linear noise removing models is the speed and the limitations of the linear models is, the models are not able to preserve edges of the images in an efficient manner i.e., the edges, which are recognized as discontinuities in the image, are smeared out. On the other hand, Non-linear models can handle edges in a much better way than linear models, but are complex [2].

**1.4.1 Median filter**:

It is a non-linear digital filtering technique used to remove noise from images. It uses two different window size of 3x3 and 5x5 for computation.

The main idea of the median filter is to run through the signal entry by entry, replacing each entry with the median of the neighboring entries. The pattern of neighbors is called the ‘window’, which slides, entry by entry, over entire image.

**1.4.2 Low Complex Noise Reduction (LCNR)**:

This technique targets at removing random-valued impulse noise. It consists of two components: Noise detector and filter.

Noise detector uses two dynamically define conditional thresholds (Tlow, Thigh) to distinguish corrupted pixels from noise-free pixels. If the detector finds the value of the pixel is larger than Thigh or lower than Tlow, it sets a binary flag (S) to 1. The median filter is activated if and only if the flag (S) is set. The filter replaces the corrupted value of current processing pixel (i, j) with the median value of those pixels in W(i, j). [11].

**1.4.3 Decision-Tree-Based De-noising Method (DTBDM)**:

A decision tree based impulse noise detection is used to detect the noisy pixels and then edge-preserving filter is used to reconstruct the intensity values of noisy pixels.

The Decision-Tree-Based Impulse detector determines whether P(i, j) is a noisy pixel by using decision tree and the correlation between pixel P(i, j) and its neighboring pixels. If the result is positive, edge-preserving image filter based on direction-oriented filter generates the reconstructed value.[12].

**1.5 APPLICATIONS**

1. The proposed filtering method allows edge detection algorithms to become immune and resilient to noise.
2. It enhances image segmentation, object recognition, feature extraction and pattern classification.
3. It helps in deriving structural and functional measurements in medical imaging specially MRI images.
4. Hardware implementation of this project can be used as a part of the medical image capturing instruments for enhancement and segmentation of MR images before during surgical operation in radiography and radiosurgery.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 INTRODUCTION**

Impulse noise is of interest to many researchers as a type of common noise. Enhancing the MR image is of greater importance in segmentation of grey matter of the Brain. With the advancement of image-guided surgical approaches, segmentation of images has become an important tool. Therefore MR image enhancement and de-noising play essential roles before and during radiosurgeries. Recent works on Random Valued Impulse de-noising methods for MR images are discussed below.

**2.2 REVIEW ON RECENT LITERATURE**

**M. Mafi, et al.[13]** proposed a new filter which combines the strengths of the median filter and those of fixed mean filters to preserve image details, and overcoming the presence of impulse noise even under high density levels.

When using an adaptive median filter, all pixels with 0 and 1 values are removed from the initial sliding window. The median value of the remaining pixels, within the window with probability of (1-Pp-Ps) is used as the filtered value for the pixel being processed. If all of them are 0s, 1s or a combination of them, then the size of the window is increased by 1 and the process is repeated until the window size reaches the predefined maximum window size.

By increasing the size of the adaptive median filter window, the structural metrics will be somewhat decreased, due to a slightly blurred image. However, the edges still be sharp. Therefore, it appears that there is a tradeoff to be made between the edges extracted and the quantitative values of the structural metrics. However, the pixel being processed will remain unchanged if the maximum window size is reached and it only contains 0s and 1s, or a combination of them. When such combinations of 0s and 1s are found in several instances in the sliding window, the mean filter is applied.

**Drawback**: It assumes that boundary edges of the filter image have high correlation with the original image.

**M. Nadeem et al.[14]** proposed a filtering technique which is based on an adaptive switching mechanism that detects the noisy pixels in the degraded image and restores those pixels by using a fuzzy rule-based directional median operation leaving noiseless pixels unaltered.

The proposed iterative filter sub-divides a large-size processing window into numerous smaller overlapping windows to form a quadrant set. Selecting the appropriate size of the processing window is very critical and should be given due consideration in any impulse noise detector algorithm. More information regarding fine details like edges and textures are obtainable in processing windows having large size, whereas small size processing windows are robust to the detection of small-scale noise densities. ꞷ is a square (2n+1) x (2n+1) processing window. A mixed approach has been introduced that uses two processing (sliding) windows of different size represented by ꞷn, where n=1,2 taking into consideration the “median drifting” problem in processing window of size 5×5 and “lack of texture” issue in very small processing window of size 3×3. To achieve this, a processing window of size ꞷ2 i.e., 5×5 is apportioned into several overlapped sub windows (quadrants) of size ꞷ1 i.e., 3×3 each. An illustration is shown in Fig. 2.1. This subdivision has a special benefit that a pixel degraded with an impulse noise at the central position in a processing window ꞷ2 remains no more at that position in a processing window ꞷ1.

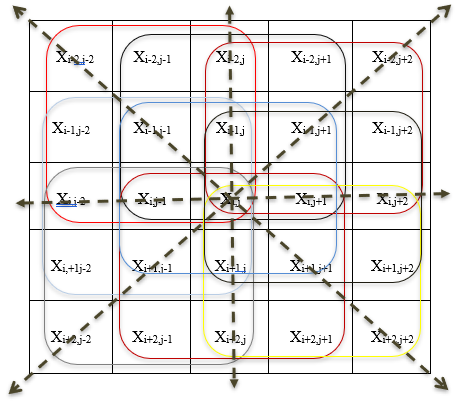
****

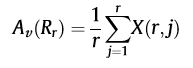
Fig. 2.1 A *5×5* window is divided into *3×3*overlapped sub-windows (*Quadrants*), *arrows* represent four directions.

A quadrant vector is computed from the sorted estimated medians of the small-size windows. This quadrant vector uses fuzzy rule-based reasoning to further explore the presence of impulse noise and the amount of texture in an image. In the restoration process, a simple switching median technique is used to restore the value of the impulsive pixel. The proposed technique successively detects and suppresses low, medium and high-density impulse noise from the deteriorated images while preserving the fine structure and details on a large scale. Experiments show that the proposed technique is equally effective for both the RVIN and SPN.

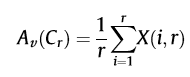
**Vikas Gupta et al.[10]** proposed de-noising method is a median ﬁlter based technique with adaptive dual threshold. In single threshold system, when a central pixel is checked for the presence of noise, any pixel value lesser than (or greater than) the given single threshold will be considered as noise. This may increase the possibility of incorrect detection. In the proposed adaptive dual threshold system, the noisy pixels are identiﬁed in a relatively narrow range and thus can reduce the probability of incorrect detection.

In the proposed method, the image to be de-noised is partitioned into sub-images (ﬁltering windows). For any given NxN gray level image which is deﬁned by f: NxN ?->I, where I =[a, b] represents the range of pixel values. Pixel value at location (i, j) is given by f(i, j). Let rxr be the size of the ﬁltering window ‘‘W’’, formed by partitioning the image. Now, the existence of noise is identiﬁed at the central pixel (CP) of distinct ﬁltering window with respect to certain deﬁned thresholds.

In images gradual changes in adjacent pixel values is more common as compared to the abrupt changes. Noise corruption at any pixel would change its gray level with respect to its surrounding pixels. Average values of a set of samples (data values) always lie in close proximity to the values under consideration. Hence any sudden change in pixel value can be easily identiﬁed by analyzing it with respect to average values. Based on this fact, averages of rows and columns of the ﬁltering window are used for threshold computation in proposed method which leads to efficient noise detection. In every ﬁltering window, minimum threshold (Thmin) and maximum threshold (Thmax) are estimated which are used to detect the abrupt changes in pixel values. In order to estimate the thresholds, ﬁrst of all, the averages of elements in individual rows (Av(R)) of the ﬁltering window are calculated.



This process will result in ‘r’ average values corresponding to every row. After that the averages of elements in individual columns (Av(C)) of the ﬁltering window are calculated.

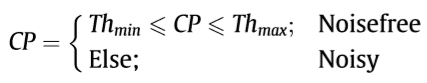


This process will also result in ‘r’ average values corresponding to every column. These ‘2r’, distinct average values will be used for ﬁnding Thmin and Thmax using following equations:

Thmin = min {Av(R1),.. Av(Rr), Av(C1),...Av(Cr) } ………(1)

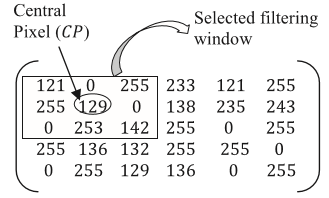
Thmax = max{Av(R1)...Av(Rr), Av(C1)...Av(Cr) } ………(2)

Now, the noise corruption at CP of the ﬁltering window is checked by comparing it with the Thmin and Thmax. If the CP value lies between thresholds computed by Eqs. (1) and (2), then it is considered as noise free otherwise noisy.

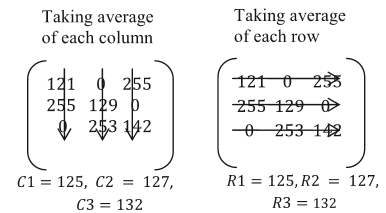


**Illustration:**

STEP-1: A ﬁltering window is selected.



STEP-2: Now dual threshold values are computed for noise detection.



Thmax = 132, Thmin = 125

STEP-3: Existence of noise is checked by comparing the Central Pixel (CP) with thresholds.

STEP-4: If the value of center pixel (CP) lies between Thmax and Thmin, the (CP) remains unchanged otherwise it is replaced by median of the filtered window.

Since the CP value lies in between the two thresholds, the central pixel (CP) is considered to be noise free and remains unchanged.

**Setu Garg et al.[8]** Discussed the performance of all the filters are compared on the basis of filter parameters root mean square error (RMSE) and peak signal to noise ratio (PSNR).

**RMSE** is basically the average square of the error between original image and filtered image. Low value should be obtained for better filtering.



**PSNR** is the ratio of signal power to noise power. High value of PSNR indicates better filtering of the image.



These two measures can measure the quality of the image. The combination of higher PSNR and RMSE indicates the best suitable filter for de-noising.

**Matsubara et al.[11]** proposed technique targets at removing random-valued impulse noise. It consists of two components: noise detector and filter. The **noise detector** determines whether the pixels are corrupted by the random-valued impulse noise or not. Each noisy pixel is processed by **median filtering** for reconstruction. Otherwise, there is no fundamental reason to modify the value of a non-noisy pixel, so the median filtering is skipped. The processing flowchart is shown in Fig. 2.3

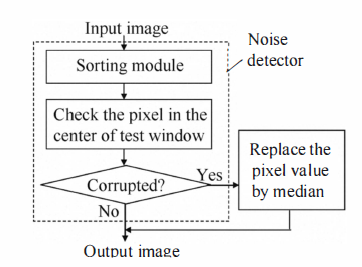
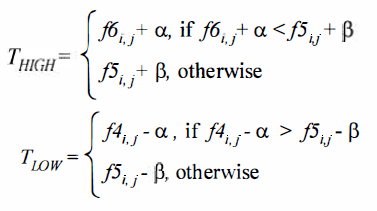


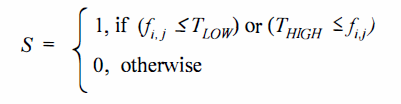
Fig. 2.2. An overview of the proposed technique.

The **noise detector** uses two dynamically defined conditional thresholds (TLOW, THIGH) to distinguish corrupted pixels from the noise-free pixels. To compute the thresholds, the noise detector sorts the values of pixels within W(i,j) in ascending order. The values of three pixels in the middle be f4i,j,f5i,j,f6i,j+1, and f4i,j f5i,j f6i,j. That is f5i,j is the median value of N2 values in W(i,j). Where W(i,j) is the set of pixels that surround the (i,j) within the test window of (N x N) pixels in size and fi,j is the value of pixel with coordinates (i,j).

The thresholds TLOW and THIGH are defined as:



If the detector finds that the value fi,j of pixel (i,j) is larger than THIGH or lower than TLOW, it sets a binary flag S to one. Otherwise, the S is zero. That is,



Thus, the flag (S=1) points out that the pixel (i,j) is corrupted. If S=0, the pixel has a noise-free value.

The **median filter** is activated if and only if the flag S is set. The filter replaces the corrupted value of current processing pixel (i,j) with the median value of those pixels in W(i,j).

The noisy image was then processed by the proposed technique (3 x3 window) with α varying from 0 to 60, while for each value of α, the value of β was changing from 0 to 100. Those values of α and β, which maximized PSNR were selected.

**C. Lien et al.[12]** proposed a novel adaptive **decision-tree-based de-noising method (DTBDM)** and its VLSI architecture for removing random-valued impulse noise. The noise considered in this paper is random-valued impulse noise with uniform distribution. Here, they have adopted a 3 x 3 mask for image de-noising. The pixel to be de-noised is located at coordinate (i, j) and denoted as pi,j, and its luminance value is named as fi,j, as shown in Fig. 2.4. According to the input sequence of image de-noising process, we can divide other eight pixel values into two sets: WTopHalf and WBottomHalf. They are given as

WTopHalf= {a, b, c, d}

WBottomHalf = {e, f, g, h}

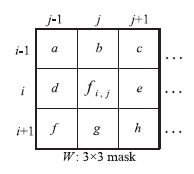


Fig. 2.3. A 3 x 3 mask centered on pi,j

DTBDM consists of two components: decision-tree-based impulse detector and edge-preserving image filter. The detector determines whether pi,j is a noisy pixel by using the decision tree and the correlation between pixel pi,j and its neighboring pixels. If the result is positive, edge-preserving image filter based on direction-oriented filter generates the reconstructed value. Otherwise, the value will be kept unchanged. The design concept of the DTBDM is displayed in Fig. 2.5.

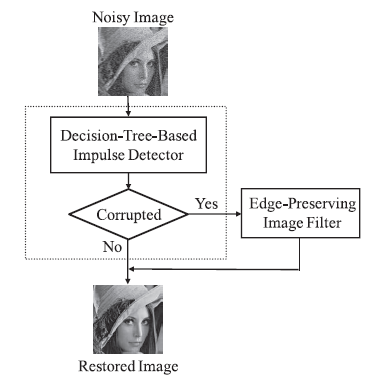


Fig. 2.4. The dataflow of DTBDM.

The decision-tree-based impulse detector, consist of three modules—isolation module (IM), fringe module (FM), and similarity module (SM). Three concatenating decisions of these modules build a decision tree. The decision tree is a binary tree and can determine the status of pi,j by using the different equations in different modules.

**Isolation module** is used to decide whether the pixel value is in a smooth region. If the result is negative, it is concluded that the current pixel belongs to noisy free. Otherwise, if the result is positive, it means that the current pixel might be a noisy pixel or just situated on an edge.

The **fringe module** is used to confirm the result. If the current pixel is situated on an edge, the result of fringe module will be negative (noisy free), otherwise, the result will be positive. If isolation module and fringe module cannot determine whether current pixel belongs to noisy free.

The **similarity module** is used to decide the result. It compares the similarity between current pixel and its neighboring pixels. If the result is positive, pi,j is a noisy pixel; otherwise, it is noise free.

**2.3 SUMMARY**

In de-noising processes, different factors such as accuracy, scalability and complexity must be taken into account. All of the mentioned factors are important, but in some applications some of these properties become critical. In real-time applications and for embedding an algorithm in a medical imaging instrument, it is essential to decrease complexity and increase speed of the operation.

Table 2.1: Comparison between existing Filtering methods and the proposed method

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Author** | **Title** | **Algorithm/Concept** | **Advantages** | **Disadvantages** |
| M. Mafi, H. Martin and M. Adjouadi | High Impulse Noise Intensity Removal In MRI Images, 2017. | This paper proposes a new filter which combines the strengths of the median filter and those of fixed mean filters to preserve image details, and overcoming the presence of impulse noise even under high density levels. | It is computationally simple and does better filtering than individual mean or median filter. | It assumes that boundary edges of the filter image have high correlation with the original image. |
| M. Nadeem, Ayyaz Hussain, Asim Munir, M. Habib and M. Tahir Naseem | Removal of Random Valued Impulse Noise from Grayscale images using Quadrant based Spatially Adaptive Fuzzy Filter, Signal Processing, 2019. | This paper proposes a filtering technique which is based on an adaptive switching mechanism that detects the noisy pixels in the degraded image and restores those pixels by using a fuzzy rule-based directional median operation leaving noiseless pixels unaltered. | A switching technique based fuzzified degree identifies a certain pixel of an image as a noise free, noisy or edge pixel in the filtering phase. | Computational complexity is higher since a processing window of size *n×n* is subdivided into an overlapped set of sub-windows of reduced size. |
| Vikas Gupta, Vijayshri Chaurasia and Madhu Shandilya | Random-valued impulse noise removal using adaptive dual threshold median ﬁlter, 2014. | This paper proposes a median filter with adaptive dual threshold. | It works to eliminate the disadvantages of a single threshold system. | Pixels at the edges may also be identified as noisy pixels. |
| T. Matsubara, V.G. Moshnyaga, K. Hashimoto. | A FPGA implementation of low-complexity noise removal, 2010. | This paper proposes a noise detector which uses dynamically defined thresholds (Tlow and Thigh) to identify corrupted pixels which sets the binary flag ‘S’. Median filter is activated only if the binary flag is ‘1’. | Low computational complexity and hence the cost of the de-noising hardware is less. | As other adaptive median filters this filter also identifies pixels at edges as noisy pixels. |
| C. Lien, C. Huang, P. Chen and Y. Lin | An Efficient De-noising Architecture for Removal of Impulse Noise in Images, 2013. | This paper proposes a decision-tree-based impulse detector and an edge-preserving image filter. The detector determines whether pi,j is a noisy pixel by using the decision tree and the correlation between pixel pi,j and its neighboring pixels. If the result is positive, edge-preserving image filter based on direction-oriented filter generates the reconstructed value. | Edge-preserving filter takes eight directional differences to reconstruct the noisy pixel. | It cannot improve image quality because propagation can cause blurring. |
| Setu Garg, Ritu Vijay, Shabana Urooj, | Statistical Approach to Compare Image De-noising Techniques in Medical Images and MR Images, 2019. | Image de-noising techniques in medical images are compared using statistical parameters such as Peak signal to Noise Ratio (PSNR) and Root mean square error (RMSE). | MSE-based metrics are frequently used as a metric for de-noising of an image. | MSE-based metrics fail to measure the structural improvement or degradation in an image after denoising. |
| Proposed Method | Impulse noise removal from MR Images for Radiosurgery Applications. | Image is divided into local image blocks, such as edges, smooth, and disordered areas. Different reconstruction methods are applied for each block depending on the noisy pixels. | Due to efficient detection and adaptive reconstruction method, noisy pixels are removed while image structures are preserved accurately. | It is necessary to use variable block size for a better detection in different image blocks which in turn increases computational complexity. |

**CHAPTER 3**

**METHODOLOGY**

**3.1 INTRODUCTION**

This chapter discusses the objectives and methodologies of this project. In this project we are considering Random Valued Impulse Noise, which is more common noise and its removal is more challenging. For random value impulse noise it is not easy to label a pixel as being noisy because it could have any grayscale value. To overcome this issue in the proposed method we categorize all image blocks in five block types. These categories are called smooth, noisy-smooth, edge, noisy-edge, and disordered blocks.

**3.2 OBJECTIVES**

The main objectives of proposed work are:

* To remove Random valued Impulse Noise from MRI Images.
* To calculate PSNR (Peak Signal to Noise Ratio) and SSIM (Structural Similarity Index) for different noise densities ranging from 5% to 40% to assess the quality of restored images.
* To achieve the better accuracy in restoring images compare to the available literature.

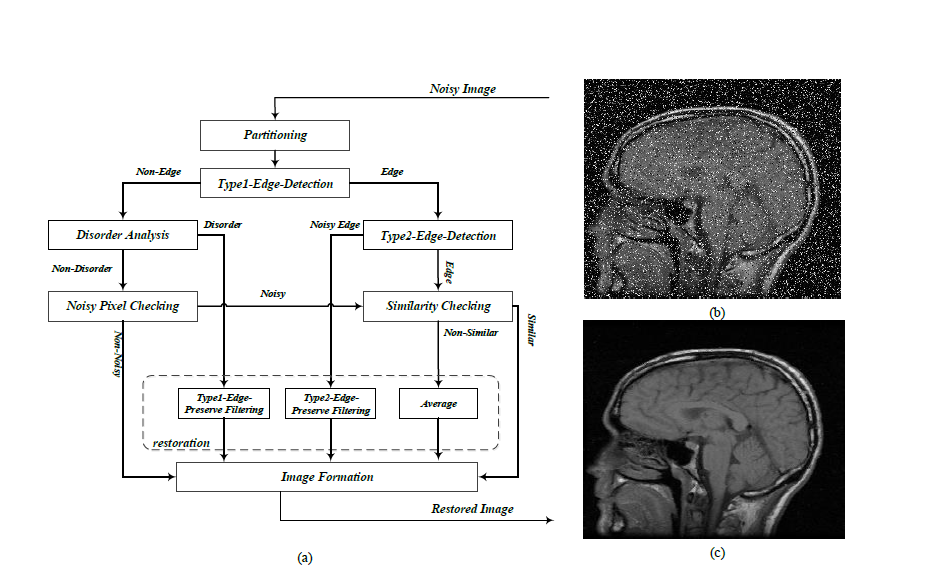
**3.3 METHODOLOGY**

Fig. 3.1. (a) General structure of the proposed algorithm, (b) a sample noisy image, (c) the de-noised image.

The general structure of the proposed algorithm is shown in Fig. 3.1.

**3.3.1Partitioning:**

Normally, neighboring pixels of an image block have similarities. Such characteristic diminishes in the presence of noise. To detect abnormal variations in pixel values, the neighborhood of each pixel should be analyzed to find out if the pixel is a normal part of that neighborhood or not. Hence, the input image is **partitioned** into 3×3 and 5×5 blocks depending on the local structure of the image.

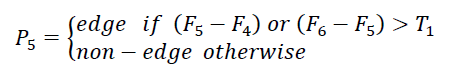
It is necessary to use variable block size for a better detection in different image blocks. For example, in edge blocks, the larger block size leads to better edge detection and better image restoration. In this paper the 9 pixels of the 3×3 block are called 𝑃1, 𝑃2, …, 𝑃9, where 𝑃5 is the central pixel. Also, in 5×5 blocks the pixels are called as 𝑃1, 𝑃2, …, 𝑃25, where 𝑃13 is the central pixel. In order to find a better detection of block categories, pixel intensities are sorted. The sorted pixel values are called 𝐹1, 𝐹2, …, 𝐹9. Partitioning of pixels in different size blocks can be useful for the next processing stages.

**3.3.2 Edge Detection:**

In noisy conditions, edges blocks can be affected by noise, hence it is necessary to find out they are noisy or non-noisy. Here edge detection is done in two steps including **Type1-Edge-detection** and **Type2-Edge-detection**.

* In the first step using **Type1-Edge-detection** it is determined that a block is containing an edge or not.

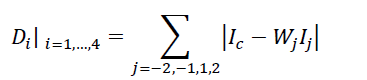
A 3×3 block around each pixel is considered and elements of the block are sorted. The differences between the 5th and 4th sorted elements, (𝐹5−𝐹4), or the 5th and 6th elements, (𝐹6−𝐹5), represent the edge strength in the block. Then using a threshold (𝑇1) the central pixel is labeled as edge based on the below equation.



For blocks that the condition is true, Type2-Edge-Detectionis also performed and for blocks that Type1-Edge-Detection is false, Disorder-Analysis is performed.

* For all pixels that are labeled as edge by the Type1-edge-detection, the second edge detection criterion is also checked. In Type1-Edge-Detection all edges are detected but it is not known whether these edges are noisy or not. Hence in **Type2-Edge-detection** step, noisy pixels are detected. Using a 5×5 block with respect to the edge directions; rough and non-noisy edges are separated from noisy edges.

In this step, noisy pixels are detected by considering major edge directions in a 5×5 block. To do so, as shown in Fig.3.2, four main directions of horizontal, vertical, diagonal, and anti-diagonal, are considered. In each of the four directions the weighted sum of absolute differences, 𝐷𝑖|𝑖=1,2,3,4, between the central pixel and the other pixels located on a particular direction, is calculated based on equation given below.



where 𝐼𝑐 is the central pixel, 𝐼±1are the two pixels that are closest to the central pixel in each direction. Also, 𝐼±2 are the two pixels that are farthest from the central pixel, in each direction. For the two farthest pixels of 𝐼±2 a weight coefficient of (1/2) is considered, which means 𝑊𝑗=½|𝑗=±2. For the two nearest pixels of 𝐼±1 a weight coefficient of 1 is considered, which means 𝑊𝑗=1|𝑗=±1.

This operation is done in all four main directions. The minimum value, in all four directions, 𝐷𝑚𝑖𝑛=(𝐷𝑖|𝑖=1,…,4), shows the most probable edge direction.

If 𝐷𝑚𝑖𝑛 were to be less than a threshold (𝑇2) there is a high similarity between the central and the edge pixels, and the central pixel is considered as an edge pixel. However, if 𝐷𝑚𝑖𝑛>𝑇2, it would be labeled as a noisy edge pixel.

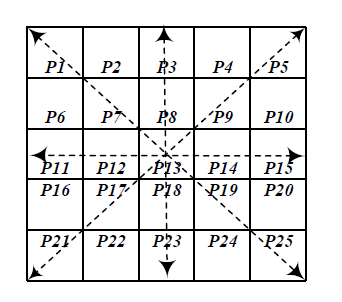
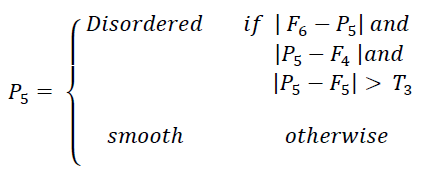


Fig.3.2.Pixel numbers and edge directions in Type2-Edge-Detectionwindow.

**3.3.3Disorder Analysis:**

When a pixels labeled as non-edge by Type-1 edge detection undergo **Disorder Analysis** to know whether the pixel is in a smooth or in a disordered area. Non-smooth blocks, and blocks that their central pixels have different values from their surrounding pixels are considered as disordered blocks.

Considering the sorted pixels of the 3 × 3 Type1-Edge- Detection window we need to work with the three median ones. Absolute difference between the central pixel and three median sorted pixels including 𝐹4, 𝐹5 and 𝐹6 is a measure of disorder within the 3 × 3 neighborhood.

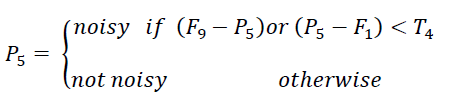


Where 𝑇3 is a threshold value. The 4th, 5th, and 6th sorted elements are considered in and their differences with the central pixel is compared with threshold 𝑇3. If all three absolute differences are greater than 𝑇3, the image block is considered to be disordered. Since the disordered blocks are potentially expected to be noisy, the Noisy-Pixel-Checking procedure is applied as the next step.

**3.3.4 Noisy Pixel Checking:**

In disorder analysis, blocks which were detected as smooth, are checked for **noisy pixels**. In fact, these blocks are included noisy pixel surrounded by smooth pixels. In smooth area, those pixels which have different intensity values from their background are determined as noisy pixels.

As equation given below the differences between the central pixel 𝑃5 and the maximum or minimum pixels inside the 3×3 window are used for detection of a noisy pixel.



If either (𝐹9 − 𝑃5) or (𝑃5 − 𝐹1) is less than a threshold 𝑇4 then the pixel is considered as noisy pixel in a smooth area. In some conditions all non-noisy block pixels may have nearly maximum or minimum values. In such a case they are wrongly considered as noisy pixels. To prevent this wrong decision, similarity between a noisy pixel and its neighbors is checked by the Similarity-Checking step.

**3.3.5 Similarity Checking:**

**Similarity Checking** is done when Type2-Edge-Detection and Noisy-Pixel-Checking have true conditions. Checking out the block similarity is necessary in two conditions. First when we want to leave the pixel without any modification. Second when by only using the pixel’s intensity the noisy pixel is detected.

Absolute differences between the central pixel and its eight neighbors are calculated to determine the similarity amount. Using threshold 𝑇4, these absolute differences determine similarity or non-similarity among these 8 pairs. If the number of the similar pixel around a pixel becomes less than a threshold (𝑇5), then it is considered to be a noisy pixel.

**3.3.6 Restoration:**

**Restoration** mechanisms are different in different blocks. Type of the restoration method depends on the block in which the pixel is detected in.

* In smooth blocks as well as in blocks that a pixel has similar value to its neighbors, restoration is performed by **averaging** on fourth, fifth and sixth elements of sorted 3×3 block.
* In the **Type1-Edge-Preserve Filtering** step, the noisy pixels are restored using the direction of the edge.

The dataflow of our edge-preserving image filter are shown in Fig. 3.3. Here, we consider eight directional differences, from D1 to D8, to reconstruct the noisy pixel value, as shown in Fig. 3.4. If the pixel is likely being corrupted by noise, we don’t consider the direction including the suspected pixel. In the second block, if d, e, f, g, and h are all suspected to be noisy pixels, and no edge can be processed, so i,j(the estimated value of pi,j) is equal to the weighted average of luminance values of three previously de-noised pixels and calculated as (a + b x 2 + c)/4.

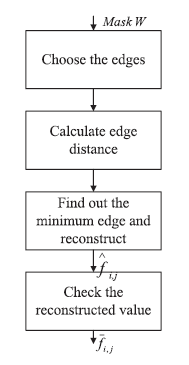


Fig. 3.3. Dataflow of edge-preserving image filter

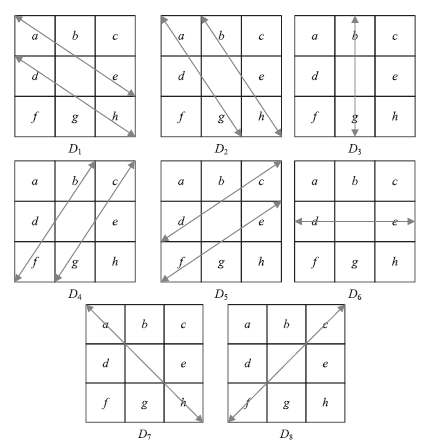
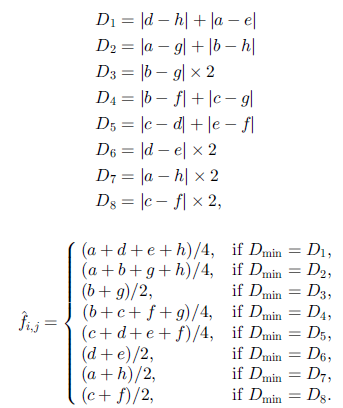


Fig. 3.4. Eight directional differences of DTBDM.

In other conditions, the edge filter calculates the directional differences of the chosen

directions and locates the smallest one (Dmin) among them in the third block. The equations are as follows:



In the last block of Fig. 3.3, the smallest directional difference implies that it has the strongest spatial relation with pi,j, and probably there exists an edge in its direction. Hence, the mean of luminance values of the pixels which possess the smallest directional difference is treated asi,j. After i,j is determined, a tuning skill is used to filter the bias edge. Ifi,j obtain the correct edge, it will situate at the median of b, d, e, and g because of the spatial relation and the characteristic of edge preserving. Otherwise, the values ofi,j will be replaced by the median of four neighboring pixels (b, d, e, and g). We can express i,j as :

i,j = Median (i,j, b, d, e, g)

* Noisy pixels which are detected in edge areas are restored with **Type2-Edge-Preserve Filtering**. In a 5×5 block, four main directions including horizontal, vertical, diagonal, and anti-diagonal, are considered. All pixels corresponding to each direction are considered. Sum of absolute differences between each pixel and their corresponding average is calculated. In this step central pixel isn’t considered in final results because this pixel is a noisy pixel. Next to determine the possible direction of the edge, the minimum value in four directions is computed. Finally restoring is performed by taking a median operation on directions which is determined in the previous step.

**3.3.7 Image Formation:**

Noise-free pixels detected in the previous steps as well as restored pixels are placed back in order to form the **noise-free image**.

A Graphical User Interface (GUI) is created, so that the user can load any image file type and then generate a de-noised image.

**CHAPTER 4**

**BLOCK DIAGRAM AND SOFTWARE REQUIREMENTS**

**4.1 INTRODUCTION**

This chapter explains the step-by-step procedure and the software requirements of the project.

**4.2 BLOCK DIAGRAM**

The below fig. 4.1, illustrates the steps followed in Impulse Noise removal of MR Images. First we take a MR image as input, we insert the noise to this MR image and calculate the noise component of the noisy image is done. After the calculations we remove the noise with the proposed methodology and again the noise component of the de-noised image is calculated.

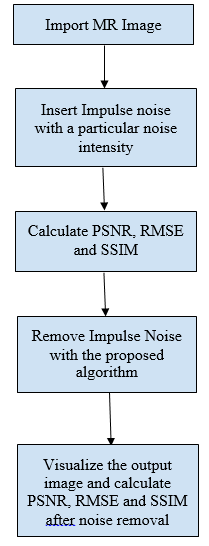
****

Fig. 4.1. Block Diagram for Impulse Noise removal in MR Images

1. This project starts with importing the MR Images from the available database.
2. Impulse noise of noise densities between 5% and 40% are uniformly injected.

Calculate RMSE, PSNR and SSIM values for different noise densities added.

**Root mean squared error (RMSE)** is basically the average of square of the error between original image and filtered image. Low value should be obtained for better filtering.

****

**Peak signal noise ratio (PSNR)** is ratio of signal power to noise power. High value of PSNR indicates better filtering of the image.

****

**Structural Similarity Index (SSIM)** is a recently proposed image fidelity measure which has proved highly effective in measuring the fidelity of signals. An image quality metric that assesses the visual impact of three characteristics of an image: luminance, contrast and structure.

Let **x** = *{xi|i* = 1*,* 2*, ...,N}* and **y** = *{yi|i* = 1*,* 2*, ...,N}* be the original and the test image signals respectively. Then, the SSIM is given by

…(1)

The above equation can be rewritten as

…(2)

The SSIM measures distortions as a combination of three factors: loss of correlation, luminance distortion and contrast distortion. The first component in (2) is the correlation coefficient between **x** and **y**. It measures the degree of correlation between **x** and **y**. Its dynamic range is [*−*1*,* 1] and the best value 1 is obtained when *yi* is linear with respect to *xi* for all *i* = 1*,* 2*, ...,N* i.e. *yi* = *axi* + *b*. The second component has a value range of [0*,* 1]. It measures the mean luminance between **x**. It equals 1 if and only if = . The third component measures the similarity of the contrast between **x** and **y**. Its range is also [0*,* 1], where the best value is 1. This occurs only when *σx* = *σy*.

1. After calculating the RMSE, PSNR and SSIM, remove the impulse noise with the proposed algorithm.
2. To check the performance of the algorithm, we calculate PSNR, RMSE and SSIM values for the de-noised MR Image.

**4.3 SOFTWARE REQUIREMENTS**

We use MATLAB R2015a software for image acquisition, noise removal, and for visualizing the de-noised image. The brief explanation of the MATLAB software is given as follows.

**4.3.1 Introduction to MATLAB:**

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages.

* + 1. **What Is MATLAB?**

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include:

* Math and computation
* Algorithm development
* Modeling, simulation, and prototyping
* Data analysis, exploration, and visualization
* Scientific and engineering graphics
* Application development, including Graphical User Interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows you to solve many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar non interactive language such as C or Fortran.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects, which together represent the state-of-the-art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

* + 1. **MATLAB system consists of five main parts:**

**4.3.3.1 MATLAB language.**

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

* + - 1. **MATLAB working environment.**

This is the set of tools and facilities that you work with as the MATLAB user or programmer. It includes facilities for managing the variables in your workspace and importing and exporting data. It also includes tools for developing, managing, debugging, and profiling M-files, MATLAB's applications.

* + - 1. **Handle Graphics.**

This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete Graphical User Interfaces on your MATLAB applications.

* + - 1. **MATLAB mathematical function library.**

This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix Eigen values, Bessel functions, and fast Fourier transforms.

* + - 1. **MATLAB Application Program Interface (API).**

This is a library that allows you to write C and Fortran programs that interact with MATLAB. It include facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

* + 1. **Starting MATLAB:**

After logging into your account, you can enter MATLAB by double-clicking on the MATLAB Shortcut icon (MATLAB 7.0.4) on your system. When you start MATLAB, a special window called the MATLAB desktop appears. The desktop is a window that contains other windows. The major tools within or accessible from the desktop are:

* The Command Window
* The Command History
* The Workspace
* The Current Directory
* The Help Browser
* The Start button

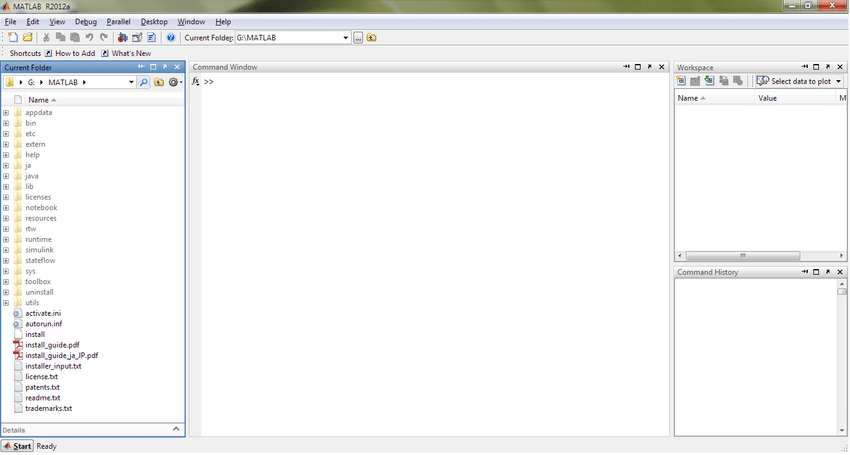


Fig. 4.2 The graphical interface to the MATLAB workspace

* + 1. **To create a simple GUI using guide:**

Open a New UI in the GUIDE Layout Editor.

1. Start GUIDE by typing **guide** at the MATLAB prompt.

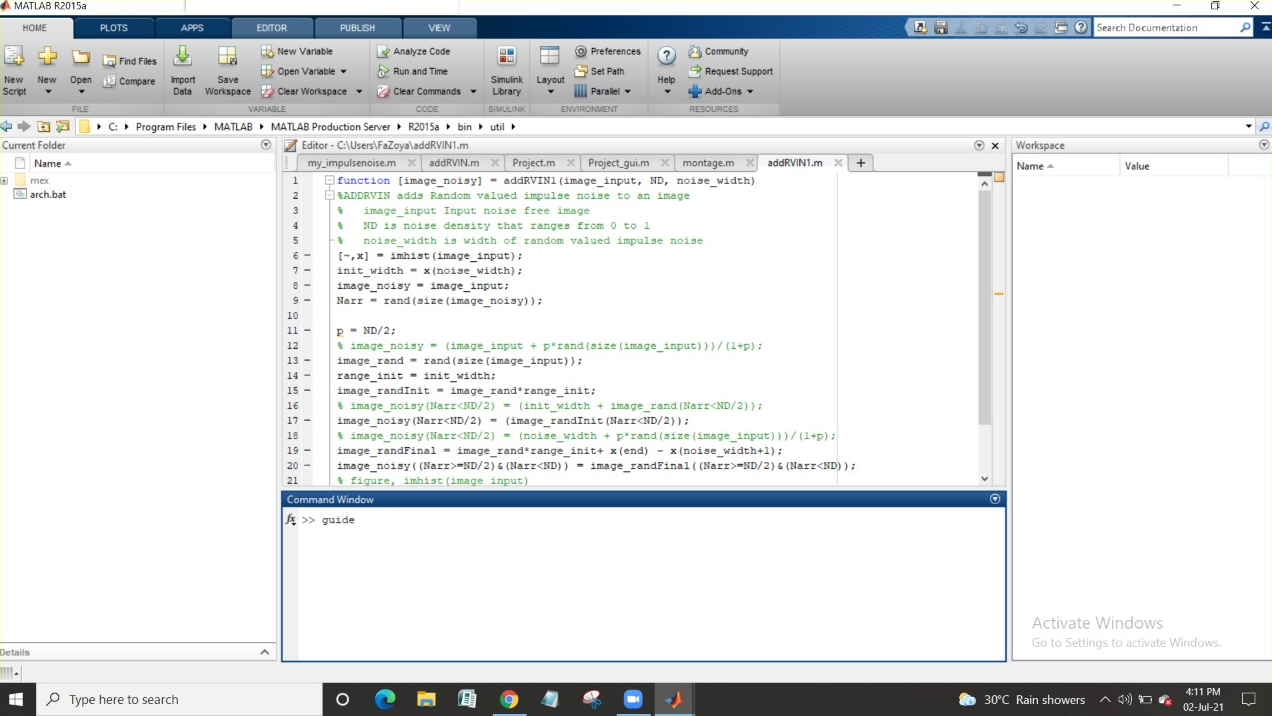


Fig. 4.3 GUI guide starting page

1. In the GUIDE Quick Start dialog box, select the **Blank GUI (Default)** template, and then click **OK**.

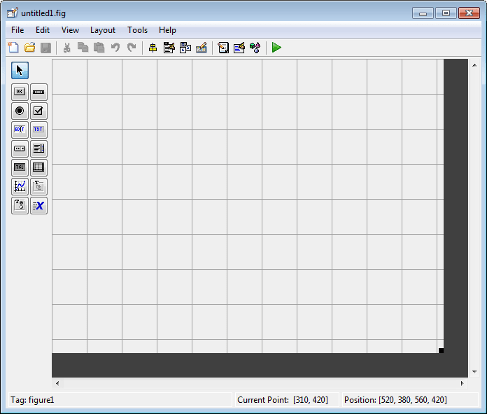


Fig. 4.4 Basic GUI image.

1. Display the names of the components in the component palette:
2. Select **File** > **Preferences** > **GUIDE**.
3. Select Show names in the component palette.
4. Click **OK**.

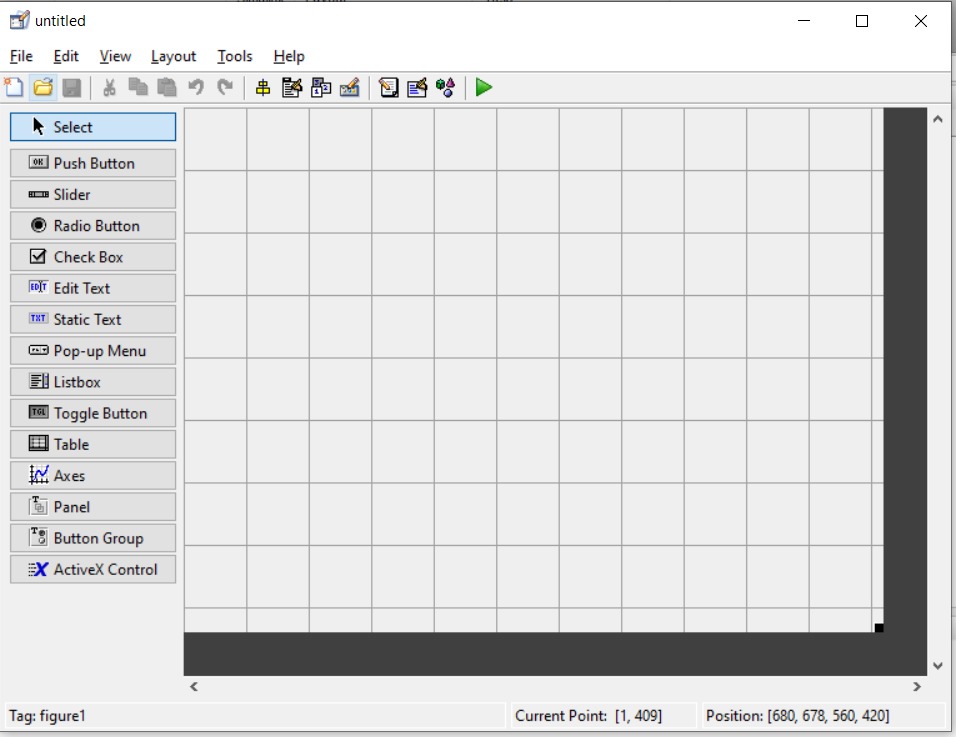
****

Fig. 4.5 Names in the component palette are displayed.

Set the size of the window by resizing the grid area in the Layout Editor. Click the lower-right corner and drag it until the canvas size is sufficient for the GUI to be designed.

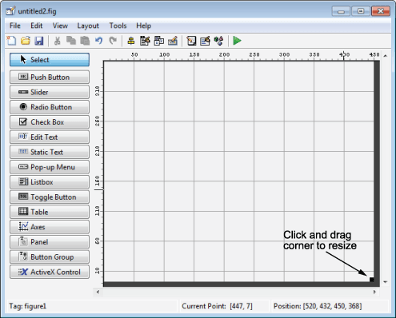


Fig. 4.6 Windows in GUI guide

Drag and drop the required toolbar from the left-side of the panel, and place it as required by you.

**4.3.5.1 Align the Components.**

If several components have the same parent, you can use the Alignment Tool to align them to one another. To align the three push buttons:

* Select all three push buttons by pressing **Ctrl** and clicking them.
* Select **Tools** > **Align Objects**.
* Make these settings in the Alignment Tool:
  + Left-aligned in the horizontal direction.
  + 20 pixels spacing between push buttons in the vertical direction.
    - 1. **Save the Layout.**

When you save a layout, GUIDE creates two files, a fig-file and a code file. The fig-file, with extension .fig, is a binary file that contains a description of the layout. The code file, with extension .m, contains MATLAB functions that control the app’s behavior.

1. Save and run your program by selecting **Tools** > **Run**.
2. GUIDE displays a dialog box displaying: “Activating will save changes to your figure file and MATLAB code. Do you wish to continue? Click Yes.
3. GUIDE opens a **Save As** dialog box in your current folder and prompts you for a fig-file name.
4. Browse to any folder for which you have write privileges, and then enter the file name simple\_gui for the fig-file. GUIDE saves both the fig-file and the code file using this name.
5. If the folder in which you save the files is not on the MATLAB path, GUIDE opens a dialog box that allows you to change the current folder.

**4.3.6 Code behavior:**

The opening function generates this data by calling MATLAB functions. The opening function initializes the UI when it opens, and it is the first callback in every GUIDE-generated code file. In this example, you add code that creates three data sets to the opening function. The code uses the MATLAB functions, peaks, membrane, and sinc.

1. Display the opening function in the MATLAB Editor. If the file simple\_gui.m is not already open in the editor, open it from the Layout Editor by selecting **View** > **Editor**.
2. On the **EDITOR** tab, in the **NAVIGATE** section, click **Go To**, and then select simple\_gui\_OpeningFcn. The cursor moves to the opening function, which contains this code:

% --- Executes just before simple\_gui is made visible.

function simple\_gui\_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.

% hObject - handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% varargin command line arguments to simple\_gui (see VARARGIN)

Run your program from the Layout Editor by selecting **Tools** > **Run**.

**CHAPTER 5**

**IMPLEMENTATION**

**5.1 Implementation Using MATLAB:**

MATLAB is an abbreviation for "matrix laboratory." While other programming languages usually work with numbers one at a time, MATLAB® operates on whole matrices and arrays. Language fundamentals include basic operations, such as creating variables, array indexing, arithmetic, and data types. Our project is implemented using MATLAB R2015a.

**5.1.1 Implementation of the MATLAB code:**

In our project we are implementing the code to remove impulse noise from MR images on the basis of the block diagram shown in Fig. 4.1.

Functions that are used from the Image Processing toolbox:

* imread(filename) : reads the image from the file specified by filename.
* imshow(I) : displays the grayscale image I in a figure.

Flow of the code:

1. Using imread, the input image is read.
2. A user defined function (addRVIN1.m) is written in order to add Random valued impulse noise of specific noise density.
   1. Input arguments : image\_input, ND and noise\_width
      1. image\_input : Input noise free image.
      2. ND : noise density that ranges from 0 to 1 (0% to 100%).
      3. noise\_width : width of random valued impulse noise (in our case 0 to 255).
   2. Output variable : image\_noisy (stores the noisy image)
3. User-defined function addRVIN is called and the output is stored in variable I1 (noisy image).
4. The noisy image is divided into smooth, disorder and edge regions and different filters are used for each region.
5. Objective testing of peak signal-to-noise ratio (PSNR) and subjective testing of structural Similarity index (SSIM) is used to assess the quality and structural fidelity of the restored images respectively.
6. The input and output image is displayed using imshow.
7. Local SSIM map of noisy and denoised images is displayed with input image taken as reference.

**5.2 Implementation of GUI:**

STEP 1: Open the gui model by using guide in the command window and press Enter to load the default gui plane.

STEP2: Default gui plane opens.

STEP 3: Create a model from the blocks.

Create a Model by dragging and dropping the tools from the toolbar at the left side of the plane.

Go to Property Inspector→Create CallBack, CreateFcn wherever required.

To build a model, we have used the following Tools from Toolbar:

* To display the name of the project, static textbox is used.
* To browse input file from the system, pushbutton is used and the tag is given as browse.
* Edit textbox is used to display the path of the selected input image.
* Edit textbox is used to accept noise density from the user and the tag is given as noise\_density.
* Axes are used to display noisy image and de-noised image and the tags noisy\_image and denoised image are given.
* The no. of times the program is repeated is selected by the user through a pop-up menu. Tag of the pop-up menu is no\_of\_iterations which is stored in a variable and later used in the program.
* Once the noisy image is loaded, DENOISE push button is used to run the denoising algorithm.
* Once Denoised image is displayed, using the edit textbox PSNR and SSIM values of both noisy and de-noised MR image is displayed, tags are psnr1, ssim1, psnr2, ssim2 respectively.

Coding Methodology:

1. User can browse the image from the system and give a specific noise density of random-valued impulse noise with which the image is affected.
2. Noisy image is displayed, the user can select the number of times the program is run (iterations) and once the denoise pushbutton is pressed it displays the denoised MR image.
3. Simultaneously PSNR and SSIM values of both noisy and de-noised MR images are displayed.

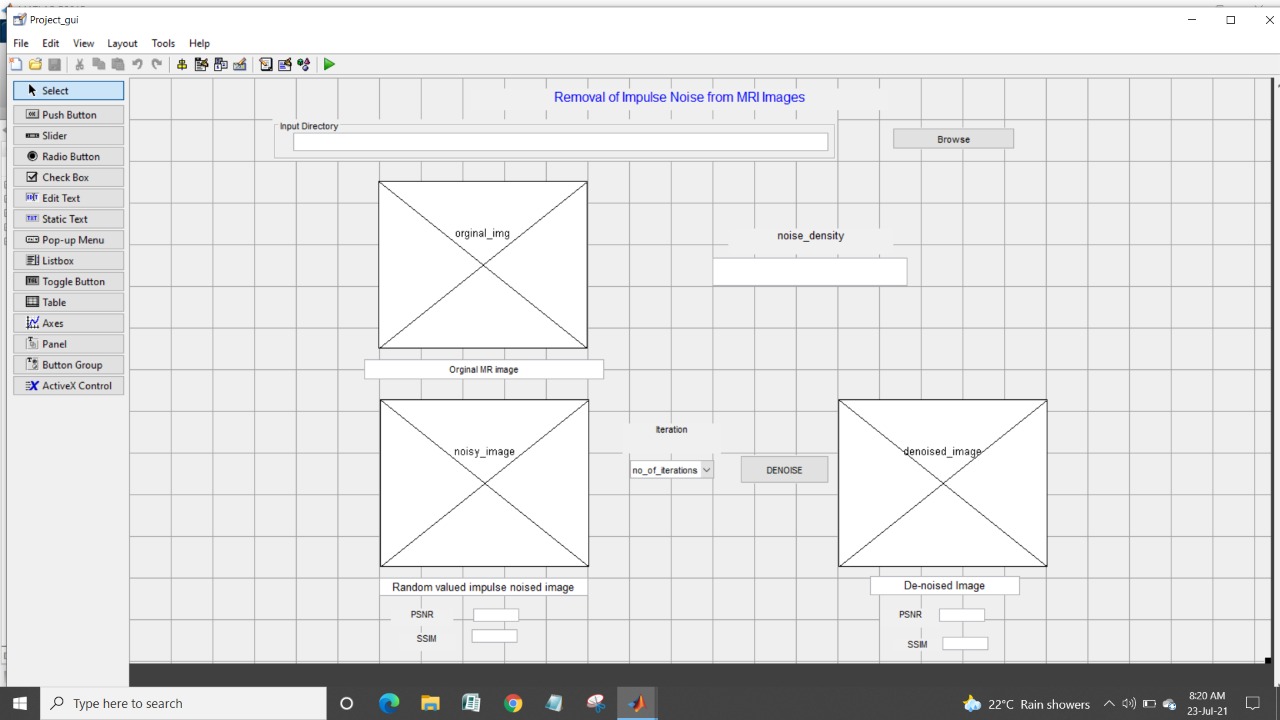


Fig. 5.1 GUI before execution

Once the model is created and the program is loaded to it, the next step is to run it.

Click on the Run button to run the model.

After pressing the Run button and giving the appropriate inputs it produces simulation result shown in Fig. 5.2.

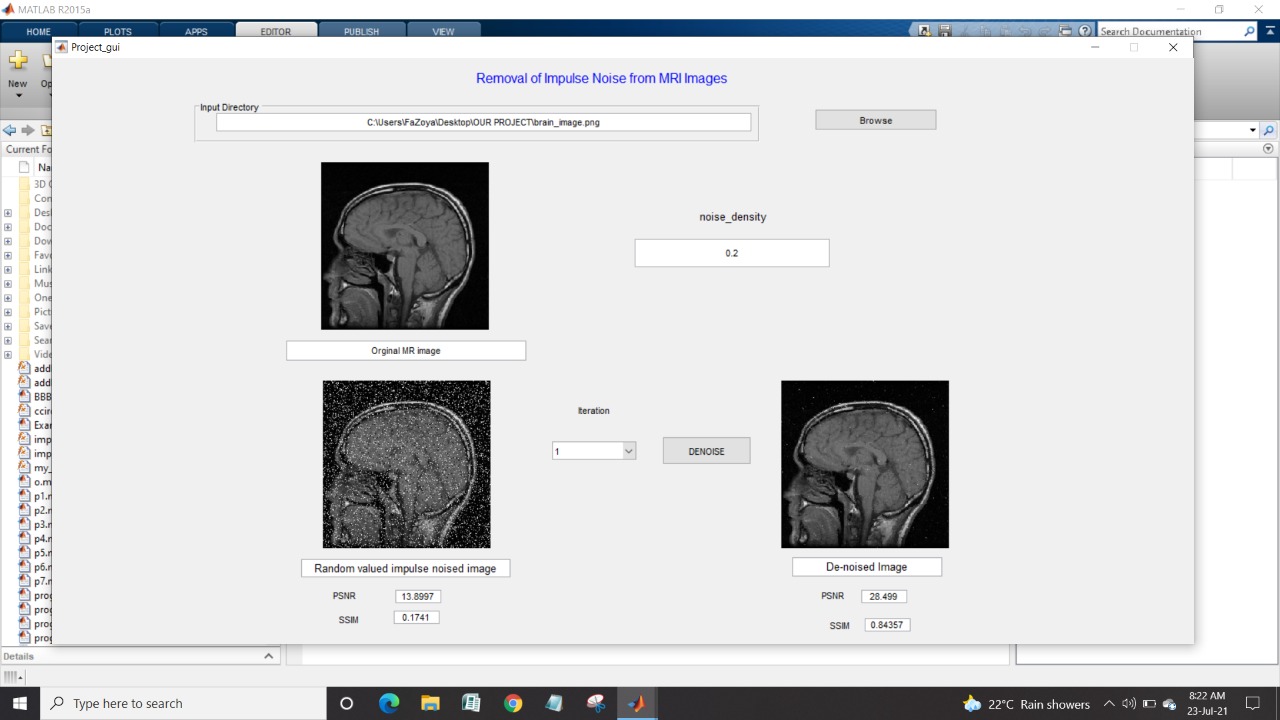


Fig. 5.2 GUI after execution, for noise density 20%.

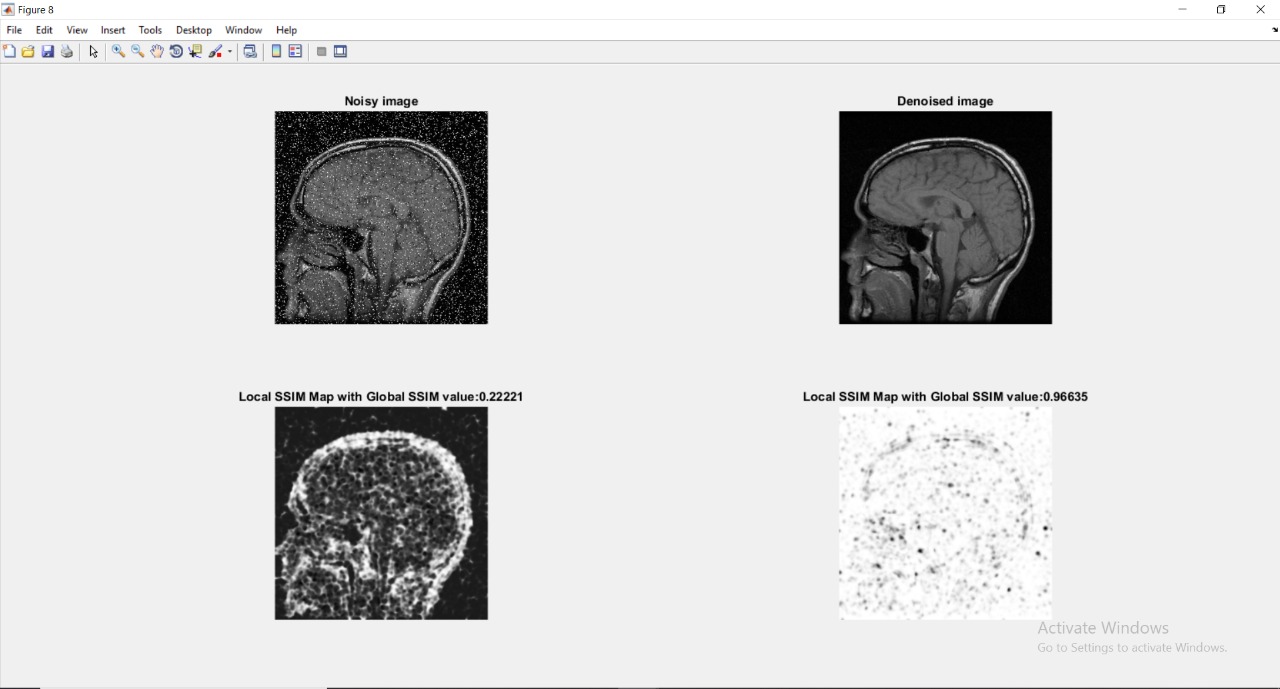
**CHAPTER 6**

**RESULTS AND DISCUSSION**

**6.1 MATLAB Code Analysis:**

MATLAB code is written for removing Random valued impulse noise (RVIN) using the proposed algorithm. Input MRI images can be of any size. We have used MR images of size 256x256 and 512x512 for demonstration.

* Input:
  + MR image from the system.
  + Noise density of RVIN.
* Output:
  + De-noised MR image.
  + PSNR and SSIM values before the removal of noise.
  + PSNR and SSIM values after the removal of noise.
  + SSIM map of MR image before and after the removal of noise.



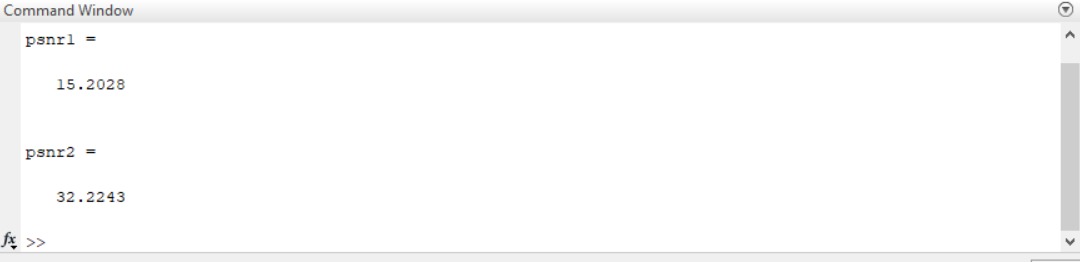
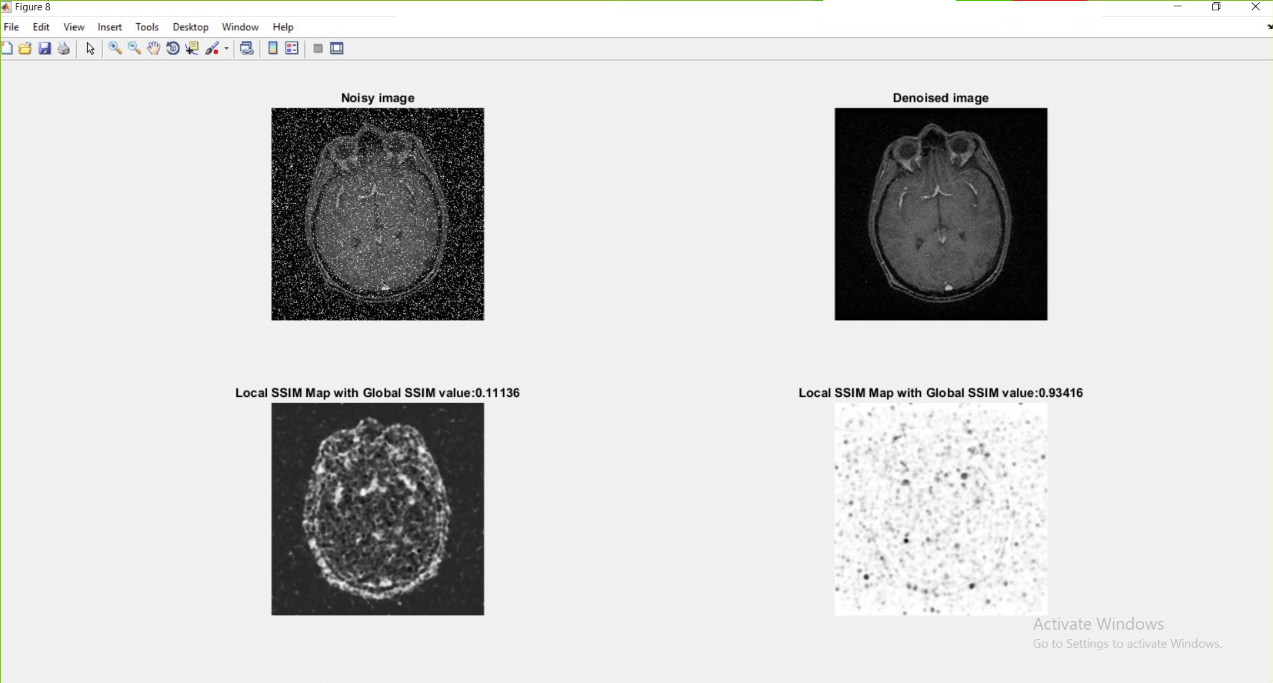


Fig. 6.1 Output for noise density, ND = 15%

* Fig 1 shows the noisy image after addition of specific noise density.
* Fig 2 shows the de-noised image.
* Fig 3 and 4 shows the local SSIM map which gives the local SSIM of noisy and de-noised image compared with the original image respectively.
* Small values of local SSIM appear as dark pixels in the local SSIM map. Regions with small local SSIM value correspond to areas where the De-noised image noticeably differs from the reference image (Original image). Large values of local SSIM value appear as bright pixels. Regions with large local SSIM correspond to uniform regions of the reference image, where the image was de-noised properly.
* psnr1 is the PSNR value of the Noisy image and psnr2 is the PSNR value of the De-Noised image.



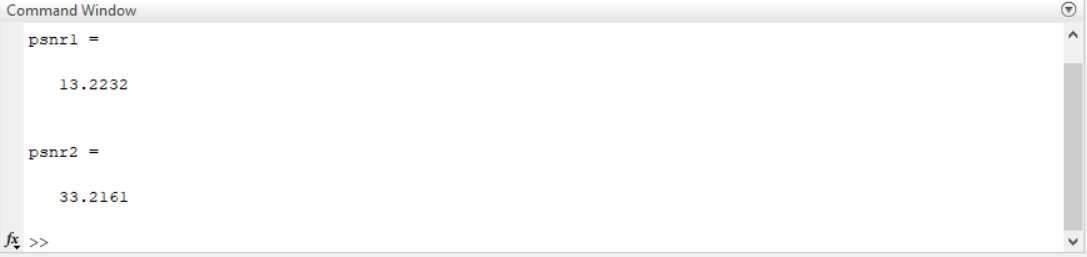


Fig. 6.2 Output for noise density, ND = 20%

PSNR and SSIM values for different noise densities ranging from 5% to 40% are recorded and shown in the table. 6.1.

Table. 6.1. PSNR and SSIM values for different noise densities.

|  |  |  |
| --- | --- | --- |
| **Noise Density** | **PSNR** | **SSIM** |
| 5% | 34.4652 | 0.98679 |
| 10% | 33.1445 | 0.97773 |
| 15% | 32.2243 | 0.96635 |
| 20% | 30.7072 | 0.95098 |
| 30% | 28.1288 | 0.88391 |
| 40% | 25.2303 | 0.76120 |

To show some visual results of the proposed method, in Fig. 6.3, four original MR images and their noisy versions with the presence of 20% impulse noise, are shown. In Fig. 6.4, Simulation results of the proposed method are shown. PSNR (dB) values are reported for each image.

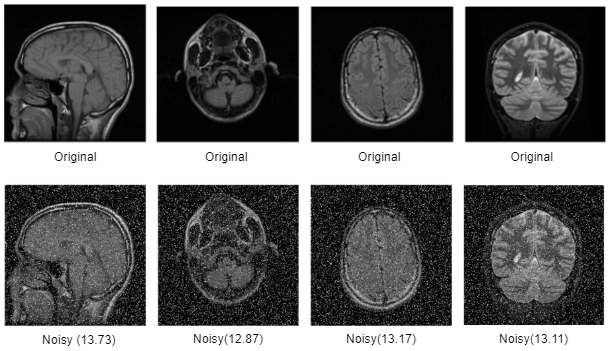


Fig. 6.3 Four sample original images and their noisy versions. Peak signal‐to‐noise ratio (dB) values mentioned for noisy images.

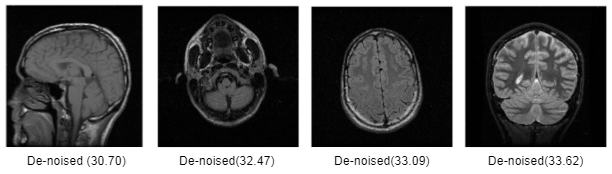


Fig. 6.4 Denoising of images of Fig. 6.3 using the proposed method. Peak signal‐to‐noise ratio (dB) values are indicated for denoised images.

**6.2 MATLAB for GUI analysis:**

This section explains about the MATLAB GUI for removal of impulse noise from MRI images.

* Input to the GUI
  + Using **Browse** Push Button an image can be selected from any folder of the system.
  + Noise density of RVIN to be added to the input image.
  + No. of iterations.
* Output
  + De-noised MRI image.
  + PSNR and SSIM values before the removal of noise.
  + PSNR and SSIM values after the removal of noise.

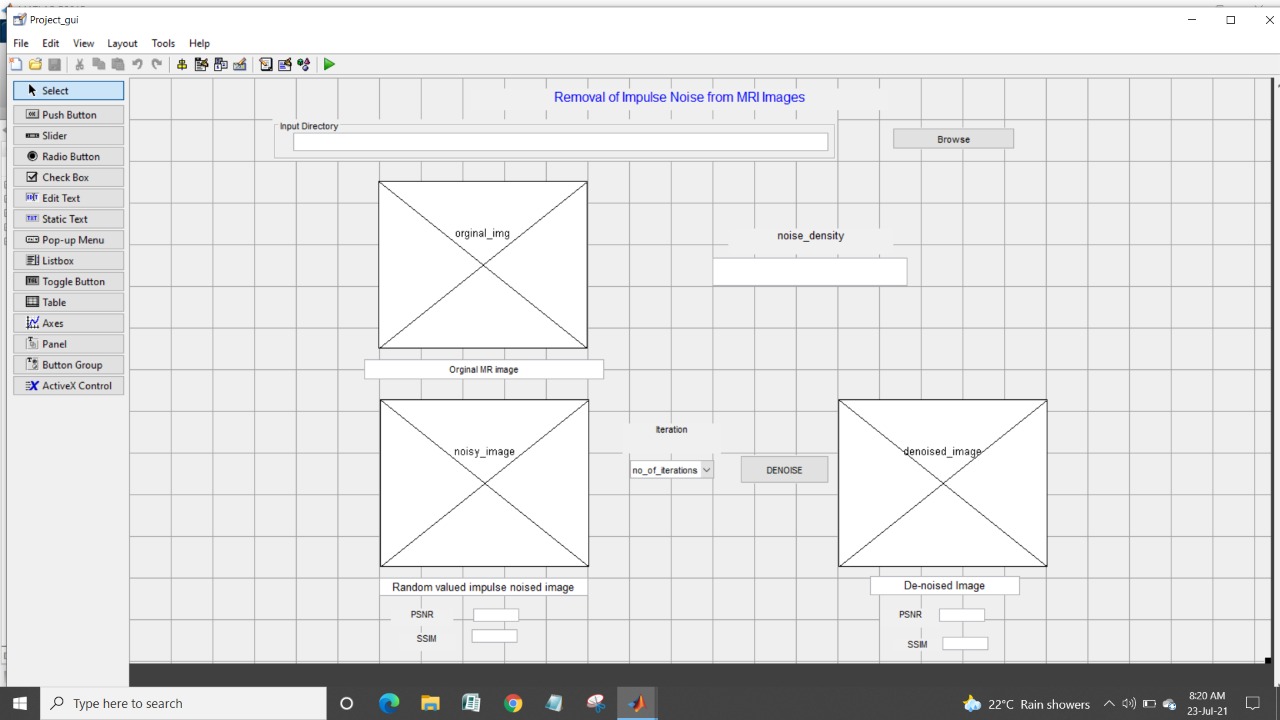


Fig. 6.5 GUI before execution

1. **Simulation-1:**
   * Input:
     1. Browse and select the image file that should be given as input to the algorithm.
     2. ND = 5%
     3. No. of iterations = 1
   * Output:
     1. De-noised image.
     2. PSNR of the image before noise removal = 19.1464 dB.
     3. SSIM of the image before noise removal = 0.38994
     4. PSNR of the image after noise removal = 38.1996 dB.
     5. SSIM of the image after noise removal = 0.97349

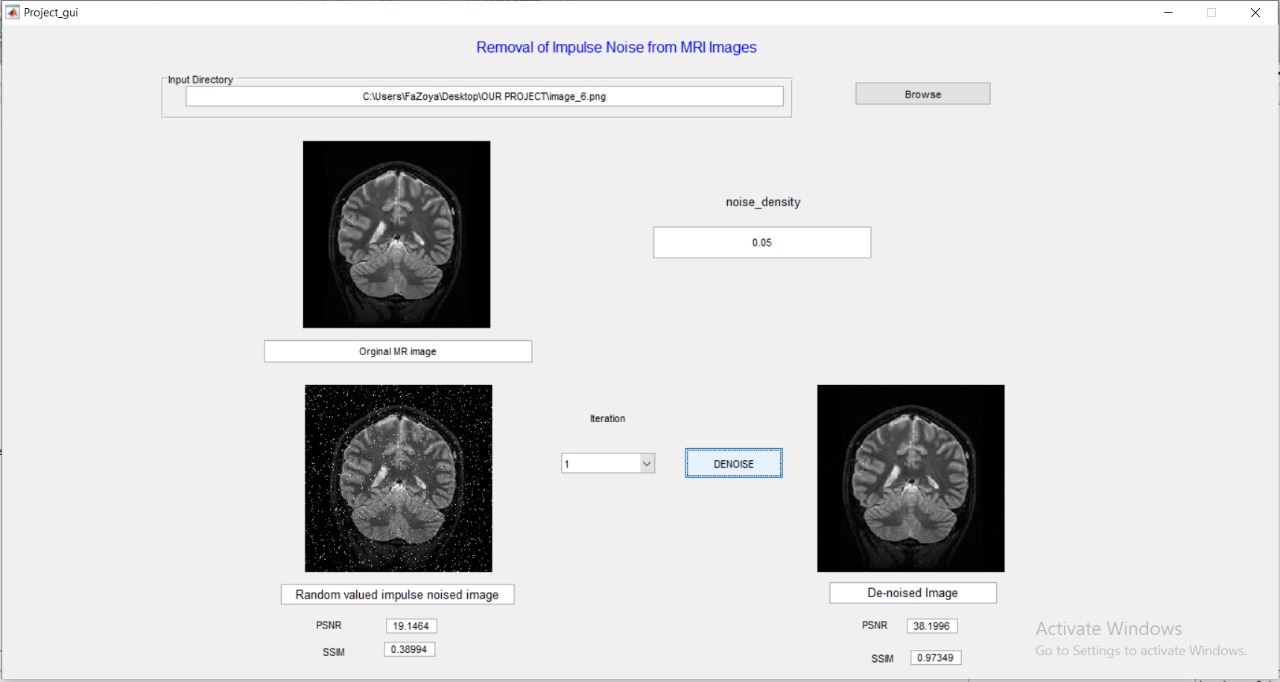


Fig. 6.6 GUI after execution, for noise density ND = 5%

1. **Simulation-2:**
   * Input:
     1. Browse and select the image file that should be given as input to the algorithm.
     2. ND = 20%
     3. No. of iterations = 1
   * Output:
     1. De-noised image.
     2. PSNR of the image before noise removal = 13.8997 dB.
     3. SSIM of the image before noise removal = 0.1741
     4. PSNR of the image after noise removal = 28.499 dB.
     5. SSIM of the image after noise removal = 0.84357

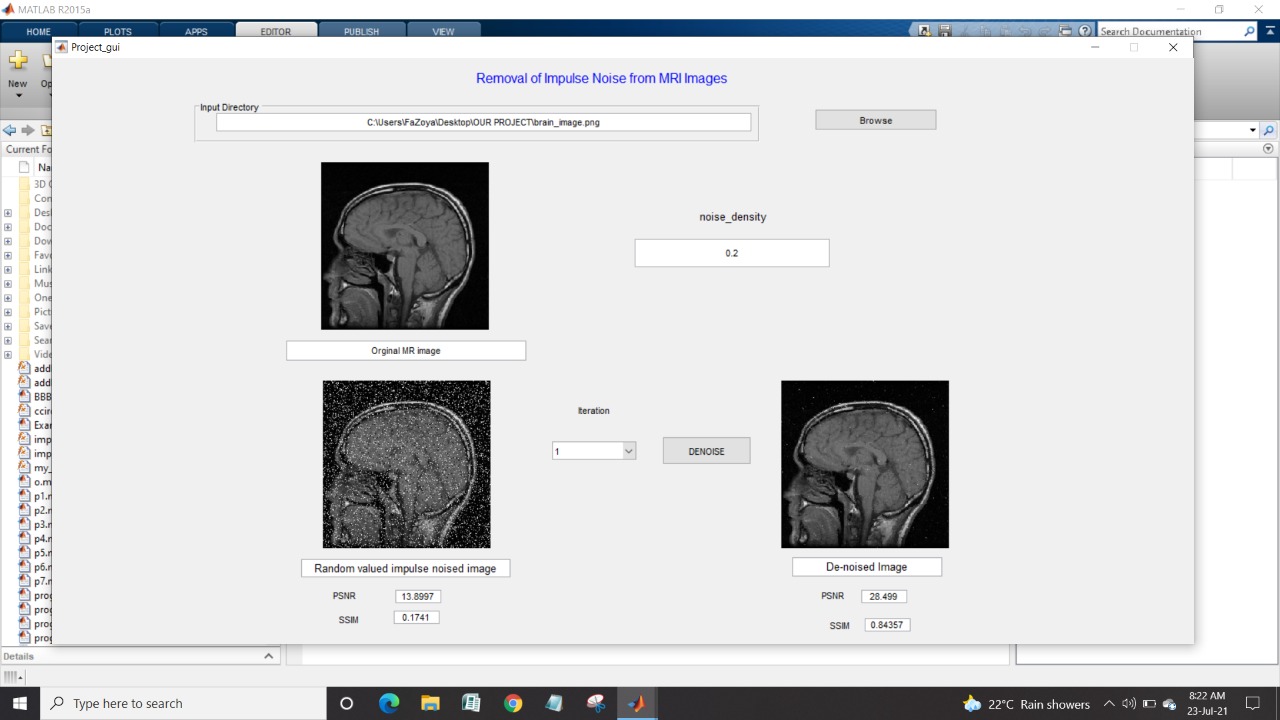


Fig. 6.7 GUI after execution, for noise density ND = 20%

**6.3 Comparison with State-of-the Art Methods:**

There are quite a few studies in the literature about Impulse Noise removal from MR Images. Two hardware architectures, proposed in references [11, 12], are used for removal of the impulse noises. Also, 3 × 3 and 5 × 5 median filters are used for comparison. As shown in the Table 6.2, the proposed method has better results than the compared methods for all noise densities.

Table 6.2: Comparison between denoised results in terms of PSNR (dB) of image size 256 x 256 for different noise densities.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Method** | **5%** | **10%** | **15%** | **20%** | **30%** | **40%** |
| Median 3×3 | 34.27 | 33.17 | 31.14 | 28.40 | 22.89 | 18.41 |
| Median 5×5 | 30.18 | 29.97 | 29.66 | 29.28 | 27.76 | 23.77 |
| LCNR[11] | 38.27 | 35.65 | 32.18 | 28.65 | 22.67 | 18.13 |
| Proposed Method | 34.47 | 33.14 | 32.22 | 30.71 | 28.13 | 25.23 |

Table 6.3: Comparison between denoised results in terms of PSNR (dB) of image size 512 x 512 for different noise densities.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Method** | **5%** | **10%** | **15%** | **20%** | **30%** | **40%** |
| DTBDM[12] | 36.18 | 34.93 | 33.78 | 32.48 | 29.62 | 26.18 |
| Proposed Method | 46.83 | 43.31 | 40.43 | 38.74 | 34.05 | 28.75 |

To show some visual results of the proposed method, in Fig. 6.3 four original MR images and their noisy versions with the presence of 20% impulse noise are shown. In Fig. 6.8 median filter is used for noise removal and PSNR (dB) values are reported for each image. In Fig. 6.9 comparison of the proposed method with [11] and [12] are shown. Simulation results, as shown in Fig. 6.9, indicate that the proposed method produces better image qualities in terms of PSNR values.

Median 3×3 (27.72) Median 3×3 (27.07) Median 3×3 (27.86) Median 3×3 (28.44)

Median 5×5 (26.07) Median 5×5 (26.78) Median 5×5 (27.25) Median 5×5 (28.97)

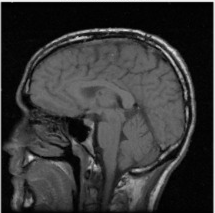
Fig. 6.8. Denoising of images of Fig. 6.3 using standard median filters. PSNR (dB) values are indicated for denoised images.

LCNR[11] (28.69) LCNR[11] (27.76) LCNR[11] (28.26) LCNR[11] (28.12)

DTBDM[12] (28.86) DTBDM[12] (29.30) DTBDM[12] (30.18) DTBDM[12] (32.48)

De-noised (30.70) De-noised (32.47) De-noised (33.09) De-noised (33.62)

Fig. 6.9 Visual quality comparison of the proposed method with [11] and [12]. PSNR (dB) values are also shown.

**REFERENCES**

1. Yanqiu Zeng, Baocan Zhang, Wei Zhao, Shixiao Xiao, Guokai Zhang, HaipingRen, Wenbing Zhao, Yonghong Peng,Yutian Xiao, Yiwen Lu, YongshuoZong, and Yimin Ding, **“Magnetic Resonance Image Denoising Algorithm Based on Cartoon, Texture, and Residual Parts"**, *Hindawi Computational and Mathematical Methods in Medicine*, 2020.
2. Alisha P B, Gnana Sheela K, **“Image Denoising Techniques-An Overview”**, *IOSR Journal of Electronics and Communication Engineering (IOSR-JECE)*, Volume 11, pp. 78-84, 2016.
3. Zohreh HosseinKhani, Mohsen Hajabdollahi, Nader Karimi, S.M. Reza Soroushmehr, Shahram Shirani, ShadrokhSamavi and KayvanNajarian, **“Real-Time Impulse Noise Removal from MR Images for Radiosurgery Applications”**, *International Journal of Circuit Theory and Applications*, 2019.
4. R. S. Vidhya, P. Ashritha, M. A. Kumar and R. N, **"Image Noise Declining Approaches by Adopting Effective Filters”,** *International Conference on Emerging Trends in Science and Engineering (ICESE)*, pp. 1-5, 2019.
5. B. Deepa and M. G. Sumithra, **"Comparative analysis of noise removal techniques in MRI brain images”**, *IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, pp. 1-4, 2015.
6. S. Gupta and R. K. Sunkaria, **"Real-time salt and pepper noise removal from medical images using a modified weighted average filtering"**, *Fourth International Conference on Image Information Processing (ICIIP)*, pp. 1-6, 2017.
7. Hanafy M. Ali. **“MRI Medical Image Denoising by Fundamental Filters”**, High-Resolution Neuroimaging - Basic Physical Principles and Clinical Applications, Ahmet MesrurHalefoğlu, IntechOpen, 2018.
8. Setu Garg, Ritu Vijay, Shabana Urooj, **“Statistical Approach to Compare Image Denoising Techniques in Medical MR Images”**, *Procedia Computer Science*, Volume 152, pp. 367-374, 2019
9. D. Chowdhury, S. Panda and S. Dutta, **"Eradication of Salt and Pepper Noise from a Tumorous MRI image using SNPRB Filter"**, *International Conference on Opto-Electronics and Applied Optics (Optronix)*, pp. 1-6, 2019.
10. Vikas Gupta, Vijayshri Chaurasia, Madhu Shandilya, **“Random-valued impulse noise removal using adaptive dual threshold median filter”**, *Journal of Visual Communication and Image Representation*, [Volume 26](https://www.sciencedirect.com/science/journal/10473203/26/supp/C), pp. 296-304, 2015.
11. T. Matsubara, V.G. Moshnyaga, K. Hashimoto. **“A FPGA implementation of low-complexity noise removal”**, *17th IEEE International Conference on Electronics, Circuits and Systems*, pp. 255-258, 2010.
12. C. Lien, C. Huang, P. Chen and Y. Lin, **"An Efficient Denoising Architecture for Removal of Impulse Noise in Images"**,*IEEE Transactions on Computers*, vol. 62, pp. 631-643, 2013.
13. M. Mafi, H. Martin, M. Adjouadi, **“High Impulse Noise Intensity Removal in MRI images”**, *IEEE Signal Processing in Medicine and Biology Syaposium (SPMB),* pp. 1-6, 2017.
14. M. Nadeem, Ayyaz Hussain, Asim Munir, M. Habib, M. Tahir Naseem, **“Removal of Random Valued Impulse Noise from Grayscale images using Quadrant based Spatially Adaptive Fuzzy Filter”**, *Signal Processing*, 2019.
15. Rafel C. Gonzalez, Richard E. Woods and Steven L. Eddins, **“Digital image processing using MATLAB”**. *McGraw Hill Education*, 2010.
16. Ndajah, Peter & Kikuchi, Hisakazu & Yukawa, Masahiro & Watanabe, Hidenori & Muramatsu, Shogo, **“SSIM image quality metric for denoised images”**, *International Conference on Visualization, Imaging and Simulation – Proceedings*, pp53-57, 2010.