IMPROVED MEDIAN FILTERING ALGORITHM FOR THE REDUCTION OF IMPULSE NOISE IN CORRUPTED 2D GREYSCALE IMAGES.

BY

WEYORI, BENJAMIN ASUBAM

(BSC. COMPUTER SCI.)

A THESIS SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF PHILOSOPHY

IN

COMPUTER ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

COLLEGE OF ENGINEERING

June, 2011

DECLARATION

This thesis is my original research work undertaken at the Department of Computer Engineering, Faculty of Computer and Electrical Engineering, College of Engineering, Kwame Nkrumah University of Science and Technology, Kumasi, Ghana under the supervision of the undersigned.

All works consulted have been duly acknowledged in the references.

BENJAMIN ASUBAM WEYORI	----------	----------
(PG2654608)	DATE	SIGNATURE
STUDENT		
DR. K. O. BOATENG	----------	----------
(SUPERVISOR)	DATE	SIGNATURE

CERTIFICATION

I certify that this thesis has been assessed and all correction has been made in accordance with the comments made by the Examiners.

DR. K. O. BOATENG	----------	----------
HEAD OF DEPARTMENT	DATE	SIGNATURE

DEDICATION

This work is dedicated to my Late Parents Mr. and Mrs. John C. Weyori and my family for the diverse support they gave me throughout my post-graduate study.

ABSTRACT

Digital images are often corrupted by Impulse noise due to errors generated in noisy sensor, errors that occur in the process of converting signals from analog-to-digital and also errors that are generated in the communication channels. This error that occurs inevitably alters some of the pixels intensity while some of the pixels remain unchanged. In order to remove impulse noise and enhance the affected image quality, the median filter has been studied and a method based on an improved median filtering algorithm has been proposed. This method removes or effectively suppresses the impulse noise in the image whiles preserving the image edges information and enhancing the image quality.

The proposed method is a spatial domain approach and uses the overlapping window to filter the signal based on the selection of an effective median per window. The approach chosen in this work is based on a functional level 2n +1 window that makes the selection of the normal median easier, since the number of elements in the window is odd. The median so chosen is confirmed as the effective median or, where the median is an impulse a more representative value is sought and used as the effective median. The improved median filtering algorithms uses the median switching technique to compute an effective median when the active median of the window is an impulse.

The performance of the proposed effective median filter has been evaluated in MATLAB using a 3 × 3 fixed window for simulations on an image that has been subjected to various degrees of corruption with impulse noise. The results demonstrate the effectiveness of the proposed algorithm vis-à-vis the standard and adaptive median filtering algorithms, and others.

The peak signal-to-noise ratios of the filtered image using the various filtering techniques are computed quantitatively, to show the effectiveness and efficiency of the method of this

thesis. The peak signal-to-noise ratio has been used to compare the performance of the proposed median filtering algorithm with other digital median filtering algorithms.

For example, the improved median filtering algorithm, when applied to the cameraman image after adding 20% impulse noise resulted in a filtered image with a peak signal to noise ratio of 38.0638. Applied to the same corrupted image, the adaptive median filter, the standard median filter, the maximum filter and the minimum filter yielded a filtered image with a peak signal to noise ratio of 33.9901dB, 32.1349dB, 25.9170dB and 30.3778dB, respectively.

ACKNOWLEDGEMENT

To Him who sits on the throne and unto the Lamb, be all glory and honor and power forever. My sincere gratitude goes to my supervisor; Dr. K. O Boateng, for his guidance, encouragement and contributions in diverse ways. I am deeply indebted to him for agreeing to work with me and making his expertise readily available to me at all times throughout the research. In fact, this work is a product of his constant devotion of time from his busy schedule to read through the scripts.

My sincere gratitude also goes to the University for Development Studies for granting me study leave and financial assistance to pursue this course. The assistance has gone a long way to improve upon my academic outlook. I would like to express my sincere thanks to the Head of Department, Computer Science Department, University for Development Studies, Dr. Gbolagade K. A. for his encouragement, invaluable advice, and patient guidance throughout this research work.

I am deeply indebted to my late parents Mr. and Mrs. John Chongati Weyori for all they sacrificed to see me this far. May your souls rest in peace. My thanks also go to Mr. Teddy Surya Gunawan of the Image signal and Information Processing Group for your technical assistance in the development of the code for the proposed algorithm.

My sincere gratitude goes to my elder brother Mr. Rowell Kutie Weyori whose encouragement and counseling like a father gave me hope when I was down. I also appreciate the patience of my wife, having given to me all that pertain to life and sharing in my dream, I love you. My special thanks goes to my beloved daughter Princess Kalisha Wematu Weyori. I finally thank all friends and relatives, the Lord bless and keep you all.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

1.	SMF	Standard Median Filter
2.	VR	Ultraviolet Radiation
3.	2D	Two Dimensional
4.	AD-Converter	Analog-to-Digital Converters
5.	CT	Computed Tomography
6.	PMC	Pulse Code Modulation
7.	IIR	Infinite Impulse Response
8.	FIR	Finite Impulse Response
9.	MRI	Magnetic Resonance Imaging
10.	IMF	Improved Median Filter
11.	AMF	Adaptive Median Filter

CHAPTER ONE

1.0 INTRODUCTION

1.1BACKGROUND

An image may be defined as a two-dimensional function, f(x,y), where x and y are spatial (plane) coordinates and the amplitude of f at any coordinate (x,y) is called the intensity or gray level of the image at that point. When x, y and the amplitude values are all finite discrete quantities we call the image a digital image. The field of digital image processing refers to processing digital images by means of a computer. A digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are referred to as picture elements, image elements, pels or pixels. Pixel is the term most widely used to denote the element of a digital image [1, 2, 3].

Image file formats are standardized means of organizing and storing digital images. Image files are composed of either pixel or vector (geometric) data that are rasterized to pixels when displayed (with few exceptions) in a vector graphic display. The pixels that constitute an image are ordered as a grid (columns and rows); each pixel consists of numbers representing magnitudes of brightness and color. Image file size expressed as the number of bytes increases with the number of pixels composing an image, and the color depth of the pixels. Larger number of rows and columns leads to better image resolution, even though the image file becomes bigger. Also, each pixel of an image increases in size when its color depth increases—an 8-bit pixel (1 byte) stores 256 colors, a 24-bit pixel (3 bytes) stores 16 million colors, the latter being known as true-color [4]. Figure 1.1 shown below is an example of the digital 2D grayscale image obtained using photo-graphic reproduction made from tapes perforated at the telegraph receiving terminal.

Figure 1.1: Digital picture produced from a coded tape by Telegraph printer

Images can be formed from many kinds of objects using differing mechanisms of formation, and, consequently, imaging systems can be classified according to several different criteria. Table 1.1 classifies systems according to the type of radiation or field used to form an image. Electromagnetic radiation is used most often in imaging systems. The radiofrequency band is used in astronomy and in magnetic resonance imaging (MRI). Microwaves are used in radar imaging, since they can penetrate clouds and other atmospheric conditions that interfere with imaging using visible light [5]. A vast number of systems use visible light and infrared radiation, including microscopy, remote sensing and industrial inspection. Ultraviolet radiation is used in fluorescence microscopy, for example, and x-rays are used in medical diagnostic work.

Table 1.1: Classification of imaging systems by type of radiation or fields used

Types of Radiation or Field	Examples
Electromagnetic Waves	Radio, Microwaves, Infrared, Vision light, Ultraviolet, (Soft) X-Ray
Other Waves	Water, Sonar, Seismic, Ultrasound, Gravity
Particles	Neutrons, Protons, Electrons, Heavy ions, (Hard) X-Ray, γ-Ray
Quasistatic Fields	Geomagnetic, Biomagnetic, Bioelectric, Electrical Impedance

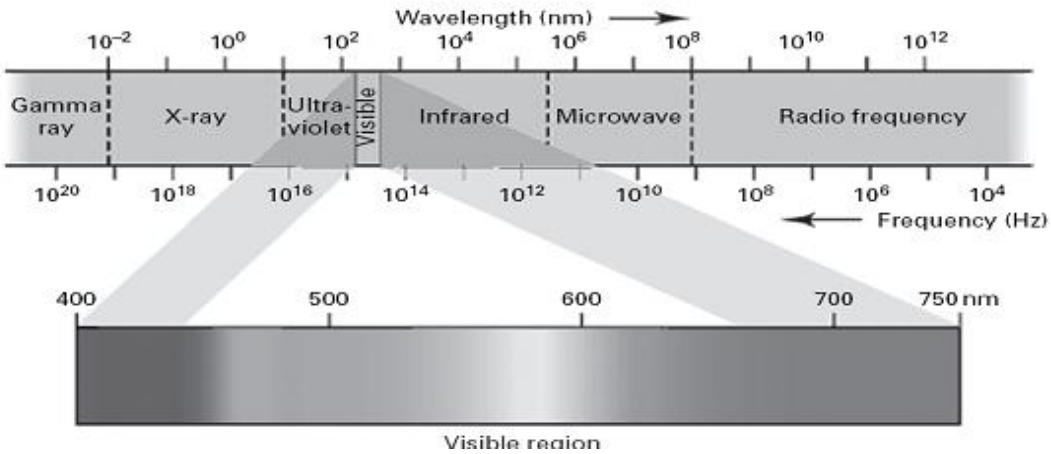


Figure 1.2: The Electromagnetic Spectrum arrangement according to the Energy of the photons or the frequency waves

An imaging system senses or responds to an input signal, such as reflected or transmitted electromagnetic radiation from an object, and produces an output signal or image. When this radiation is focused and then sensed by a photographic film, for example, it gives rise to an image that is recognized as analog, comprising continuously varying shades or colors. A grayscale photographic image is a two-dimensional function of optical density or brightness and position. If an object can move, its image is an average over the exposure time. A color image is represented by three two-dimensional functions, each corresponding to the density of one of the three color emulsions, red, green and blue, on the film. It might be argued that these images are not continuous (i.e., analog) at the level of the silver halide particles of the photographic emulsion, which are the sensors; but the scale of these is considerably below the level of perception of the human eye [6].

More recently, with the advent of small solid-state electronic detectors in digital still and video cameras, the option exists to capture the radiation using sensors organized in a two dimensional array. This sensor array, placed at the focal plane, produces outputs proportional to the integral of the radiation received at each sensor during the exposure time, and these

values become the terms in a two-dimensional matrix, which represents the scene; this is called a sampled image. It is not yet a digital image. The physical disposition of sensors facilitates the collection of data into an array, but the values themselves are still integrals and hence continuous; they need to be quantized to a discrete scale before the image is a digital image. Digital images can be represented by an array of discrete values, which makes them amenable to storage and manipulation within a computer.

An imaging system can either be a continuous-to-continuous system, responding to a continuous input signal and producing a continuous or analog output image, or it can be a continuous-to discrete system, responding to the continuous input signal by producing a discrete, digital output image. Tomographic images are reconstructed from many, one dimensional, view or projections collected over the exposure time. X-ray computed tomography (CT) imaging is an example of a continuous-to-discrete imaging system, using computer reconstruction to produce a digital image from a set of projection data collected by discrete sensors [5].

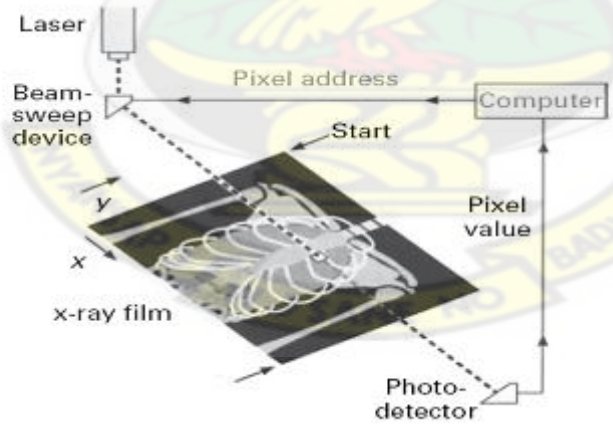


Figure 1.3: Analog image scanned

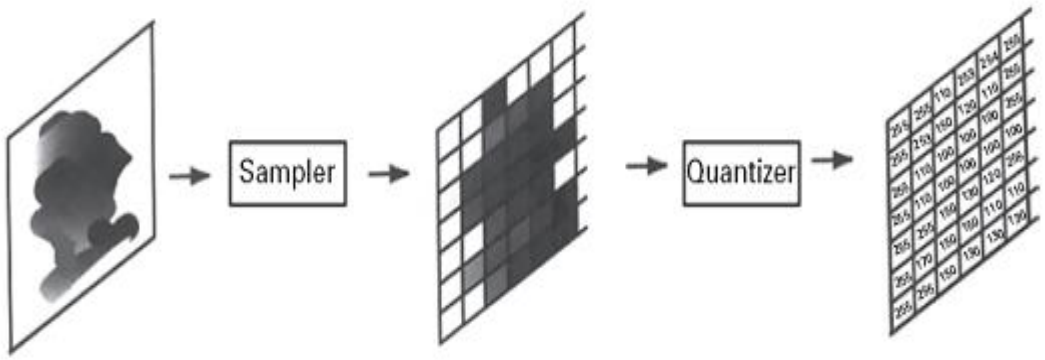


Figure 1.4: Relationship between Analog and Digital image

There are no clear cut boundaries in the continuum from processing at one end to computer vision at the other. Computer image processing is divided in to three categories low-level, medium-level and high-level processing.

Low-level processing involves primitive operation such as image preprocessing to reduce noise, contrast enhancement and image sharpening. A low-level process is characterized by the fact that both its input and output are images. Medium-level processing of images involves tasks such as segmentation (partitioning an image into regions or objects), description of those objects to reduce them to form suitable for computer processing and classification (recognition) of individual objects. A medium-level process is characterized by the fact that its input generally is images but its outputs are attributes extracted from those images (eg. edges, contours and the identity of individual objects). Finally, higher-level processing involves "making sense" of an ensemble of recognized objects, as in image analysis, and at the far end of the continuum, performing the cognitive function normally associated with vision [7, 8, 9].

Image Processing is one of the rapidly and fast growing fields in the area of computer science and engineering. The growth of this field has been improved by the technological advances in digital computing, computer processors, digital signal processing and mass storage devices.

All fields which were operating on the traditionally analog imaging are now gradually switching to the digital systems for their ease of use, affordability and flexibility. Image processing is very useful and has been extensively used in the area of medicine, film and video production, photography, remote sensing, desktop publishing, military target analysis, and manufacturing automation and control [7, 10]. The various applications, such as those mentioned, usually require bright and clear images or pictures, hence corrupted or degraded images need to be processed to enhance easy identification and further works on the image. Image processing techniques such as image enhancement, object detection and image filtering are used to process the image depending on the type of interference that has caused the degradation factor.

1.2 APPLICATION AREAS OF IMAGE PROCESSING

In this era of technology, there is no area of technical endeavor that is not impacted in some way by digital image processing. This research will only highlight on a few areas where image processing is used.

The fields that use digital image processing techniques can be divided into criminology, microscopy, photography, remote sensing, medical imaging, forensics, transportation and military applications.

1.2.1 CRIMINOLOGY / FORENSICS

Few types of evidence are more incriminating than an image evidence or video tape that places a suspect at a crime sense. Ideally, the images will be clear, with all persons setting and objects reliably identifiable. Unfortunately, the image maybe grainy, blurred, of poor contrast or damaged by noise. More commonly the images are a polluted with noise. In this

case, only a computerized technology that enables digital image processing is required to recover the distorted or noisy image for identification.

1.2.2 MEDICAL IMAGE

This is a technology that can be used to generate images of a human body (or part of it). These images are then processed or analyzed by experts, who provide clinical prescription based on their observation. These images such as the X ray image and Ultrasonic images are very much used in the hospitals all over the world. In Ghana most of the hospital that used this X ray is the teaching hospital like OKomfo Anokye Teaching Hospital, Tamale Teaching hospitals. This image generated from the X ray and Ultrasonic need to be clear for analysis. When the image generated is distorted or corrupted then image processing is used.

1.2.3 REMOTE SENSING

This is a technology of employing remote sensors to gather information about the earth. Usually the techniques used to obtain the information depend on the electromagnetic radiation, force fields or acoustic energy that can be detected by cameras, radiometers, lasers, radar systems. For clear picturing of the earth details, image need to be processed when the earth image goes through a medium and gets distorted or corrupted.

1.2.4 MILITARY

This area has been overwhelmingly studied recently. Existing applications consist of detection, tracking and three dimensional reconstructions of territory. For example, a human body or any subject producing heat can be detected in the night time using infrared imaging sensor. This technique has been commonly used in the battle fields. Another example is that

three dimensional recovery of a target is used to find its correspondence the template stored in the database before this target is destroyed by a missile.

1.2.5 TRANSPORTATION

This is a new area that has just been developed in recent years. One of the key technological processes is the design of automatically driven vehicles, where imaging systems play a vital role in path planning, obstacle avoidance and serve control.

1.3 PROBLEM STATEMENT

Digital image signals can be corrupted by impulse noise (salt and pepper) in a variety of ways from errors in collection sensors to noisy transmission channels to faulty storage devices. The problem of extraction of one signal from another arises quite often. In many applications the desired image signal is not available or observed directly. Instead, the desired image signal is corrupted by impulse noise. In some simple situation, it may be possible to design a filter to recover the desired signal from the data. However these filters are rarely optimum in the sense of producing the best extraction of signals. The standard median filter is the best ordered statistics filter in the removal of impulse noise from a noisy image. Though the standard median filter produces good result but has some deficiencies. They are

- Signal weakening (objects counters and edges blurred)
- Affecting non-corrupted (good) image pixels [11].

The adaptive median filter is developed to handle the problems of the standard median filter. Hence the adaptive median filter performs better than the standard median filter in the filtering out of impulse noise. This algorithm in literature of noise reduction yields good results depending on the density of the impulse noise in the image. The adaptive median filter

has some drawbacks when the density of the impulse noise in the image is “great” or is “less” [12];

- Detects noise and replaces the noise pixel with noise
- Affecting some non-corrupted (good) pixels when the density of the noise is less.
- Increase processing time when the density of the impulse noise in the image is great
- Difficult to detect and preserve edge details.

1.4 OBJECTIVE OF STUDY

The main objective of the research is to design and improve upon the median filtering algorithms so as:

- To avoid signal weakening (objects counters and edges blurred)
- To make sure non-corrupted (good) image pixels are left intact, irrespective of density of noise in the image
- To avoid the situation where detected noise pixel is replaced with another noise pixel
- To reduce time complexity of the algorithm
- To make it easier for median filtering to detect and preserve edge details

1.4.1 SPECIFIC OBJECTIVES

The ultimate goal of this research thesis is to attempt to reconstruct or recover an image that has been degraded by using prior knowledge of the type of impulse noise (salt and pepper). To achieve this goal the set specific objectives of this research thesis are:

- To derive an improved algorithm that is effective, efficient and very simple in the impulse noise reduction in the image.

- To implement the algorithm in computer software for simulation purpose To apply the proposed algorithm in filtering noise from noisy images to which the standard median filter and/or the adaptive median filter have been applied
- To compute representative signal-to-noise ratios.

1.5 JUSTIFICATION

Digital images are often corrupted by Impulse noise due to errors generated in noisy sensor, errors that occur in the process of converting signals from analog-to-digital error, dead pixels and also errors that are generated in the communication channels. This error that occurs inevitably alters some of the pixels intensity while some of the pixels remain unchanged and causing the image to be corrupted by impulse noise (salt and pepper noise).

The project is justified because impulse noise is the most common image noise. Impulse noise is introduced into an image data when there is a failure in any electrical system used for storage, in transmission and/or in processing.

The median filter is considered because it is the best known nonlinear ordered statistic filter for filtering of impulse noise but has some computational deficiencies. The adaptive median filter was developed to solve the problems of the median filter and it achieves better results, however, it also still faces some deficiencies. This project is aimed at improving median filtering that achieves good results through simple manipulations.

1.6 SCOPE OF STUDY

The scope of this research work is to carry out a study on the standard median filtering algorithms and adaptive median filtering algorithm, carefully outline the strengths and weaknesses of these algorithms. A decision base nonlinear filtering algorithm will be derived

to eliminate some of the shortcomings whiles drastically reducing some others. The derived algorithm will be implemented in MATLAB and tested on some standard images. Also, the outcome of the implementation will be compared with the result of application of the standard median filtering algorithm and the adaptive median filtering algorithm to show the effectiveness and efficiency of the derived algorithm. Finally, the drawbacks of the newly derived median filtering algorithm will be outlined for further studies.

1.7 ORGANIZATION OF THESIS

The rest of the work is organized as follows.

In chapter 2, a literature review of signal processing and terminology is presented. Also an overview of filters is presented. The overview includes the definition of a filter, types of filters and a brief explanation of linear and non-linear filters. Finally in this chapter, an in-depth review and explanation of the mode of operation of the standard median filter and the adaptive median filter are outlined.

In the third chapter, the limitations of the standard median filtering algorithm and the adaptive median filtering algorithm are explained. The design of the proposed median filtering algorithm is explained with the aid of flowcharts. Further, an illustration of the operation of the algorithm is presented.

In chapter four, the algorithm implementation in the MATLAB Programming Language and mode of operation are explained. The developed code is tested on some standard images, corrupted with different percentages of impulse noise. The results are also presented in this chapter with comparisons with the results of the standard median filter and the adaptive median filter.

Finally, in chapter 5, the conclusion and recommendation are presented.

CHAPTER TWO

1.0 REVIEW OF LITERATURE

This chapter summarizes all the relevant literature researched during the course of this thesis. It presents certain approaches used by many

2.1 SIGNAL PROCESSING AND TERMINOLOGIES

In this section, we will be looking at the meanings of many terms that are commonly used in signal processing and define and explain the terminologies with relation to signals or waveform. Analog and digital signals as well as continuous and discrete signals will be discussed into details. Some of the digital signal processing techniques will also be reviewed in this work.

Analog signals are a representation of time varying quantities in a continuous signal. Basically, a time variance is presented in a manner in which some sort of information is passed using various types of methods. These methods can include electrical, mechanical, hydraulic or pneumatic systems. Unlike digital signals, which use a numeric method of transmitting information, analog signals use small fluctuations in the signal itself to pass information. Analog signals act essentially like simulations of a continuous time varying quantity. They duplicate the features of the actual quantity by presenting a different quantity. In other words, they use one method of recording information and transfer it to a different format that, in turn, presents the information in that medium. Each analog signal uses a property of the final medium to convey the information for the signal. For example, a thermometer will utilize the heat of a particular object to determine its temperature. The heat is then transferred to mercury, which changes its position to display the temperature information on the gauge.

The most common form of analog signal transmission occurs electrically. In order for this to happen, a voltage must be sent at a specific frequency. The flow of this electrical charge is

known as current. By controlling the frequency of the current, information can be transmitted to another medium and presented on that medium. For example, magnetic tape on a cassette conveys information to the stereo which transmits it into electrical signals of specific frequencies which in turn tell the speakers what noise to make.

Disadvantages of the system include the tendency to create unwanted variations in the information transmission such as noise. These unwanted variations can occur in random patterns. When an analog signal is copied and potentially re-copied, each subsequent version exhibits more of the random patterns, making information transmission harder and ultimately causes signal loss [13, 14, 15].

Analog signals have a great advantage over digital signals in that they have a much higher density which can present more refined information. Essentially, there is the potential for the signal resolution to be infinite. In addition, the process to create an analog signal is achieved much more simply. By merely adjusting the time quantities, information can be presented. In order to avoid these disadvantages, or at least mitigate their effects, the concept of modulation can be used. The base analog signal is modified in some way to help retain the information as it is transmitted. An example of this is when the amplitude of a waveform is altered. This is known as amplitude modulation. Other options for retaining an electric signal over different generations by using increased shielding or different cable types twisted together [16].

A continuous signal is a mathematical function of an independent variable $t \in \beta$, where β represents a set of real numbers. It is required that signals are *uniquely* defined in t except for a finite number of points. For example, the function $f(t) = \sqrt{t}$ does not qualify for a signal even for t > 0since the square root of t has two values for any non negative t. A continuous

signal is represented in Figure 2.1. Very often, especially in the study of dynamic systems, the independent variable t represents time. In such cases f(t) is a time function.

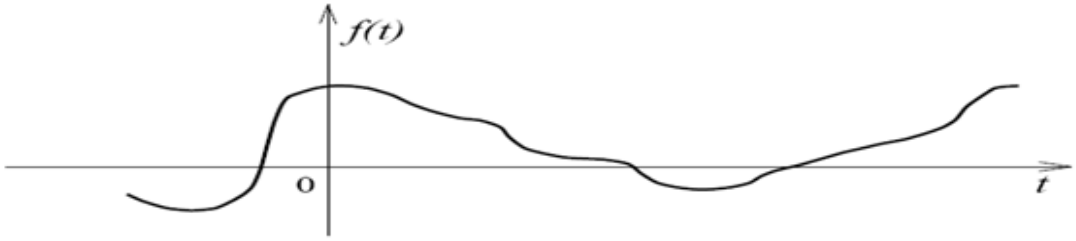


Figure 2.1: A Continuous Signal

Note that signals are real mathematical functions, but some transforms applied on signals can produce complex signals that have both real and imaginary parts. For example, in analysis of alternating current electrical circuits we use phasors, rotating vectors in the complex plane $I(j\omega) = |I(j\omega)| \angle I(j\omega)$, where ω represents the angular frequency of rotation, $\angle I(j\omega)$ denotes the phase, and $|I(j\omega)|$ is the amplitude of the alternating current. The complex plane representation is useful to simplify circuit analysis; however, the above defined complex signal represents in fact a real sinusoidal signal, oscillating with the corresponding amplitude, frequency, and phase, represented by $|I(j\omega)| \sin(\omega t + \angle I(j\omega)$ [17, 18].

A digital signal is a physical signal that is a representation of a sequence of discrete values (a quantified discrete-time signal), for example of arbitrary bit stream, or of a digitized (sampled and analog-to-digital converted) analog signal. The term digital signal can refer to

1. a continuous-time waveform signal used in any form of digital communication.
2. a pulse train signal that switches between a discrete number of voltage levels or levels of light intensity, also known as a line coded signal, for example a signal found in

digital electronics or in serial communications using digital baseband transmission in, or a pulse code modulation (PCM) representation of a digitized analog signal.

A signal that is generated by means of a digital modulation method (digital pass band transmission), produced by a modem, is in the first case considered as a digital signal, and in the second case as converted to an analog signal [19].

A discrete-time signal is a sequence or a series of signal values defined in discrete points of time. It is also uniquely defined as a mathematical function (single-valued function) of an independent variable *K€Z*, where *Z* denotes a set of integers. Such a signal is represented in Figure 1.2. In order to clearly distinguish between continuous and discrete signals, we will use in this book parentheses for arguments of continuous signals and square brackets for arguments of discrete signals, as demonstrated in Figures 1.1 and 1.2. If *K* represents discrete time (counted in the number of seconds, minutes, hours, days, ...) then *g[K]*defines a discrete-time signal [20].

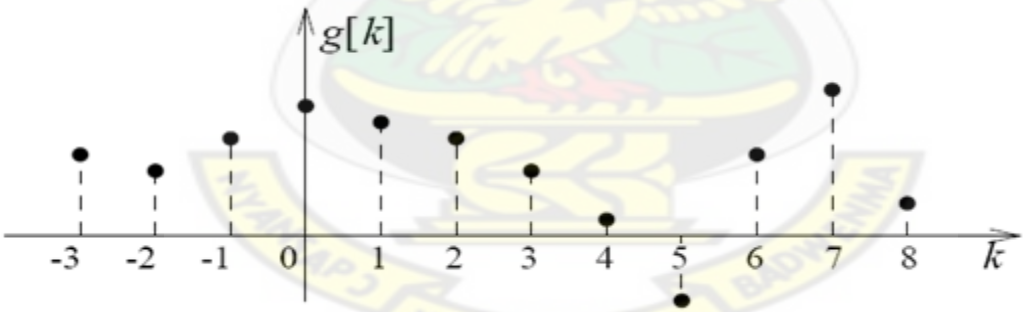


Figure 2.2: A Discrete Signal

Continuous (analog) and discrete (digital) signals can be related through sampling operation in the sense that a discrete signal can be obtained by performing sampling on a continuous-time signal. The AD converter (analog-digital) converts a continuous signal $y_{\alpha}(t)$, which can be a voltage

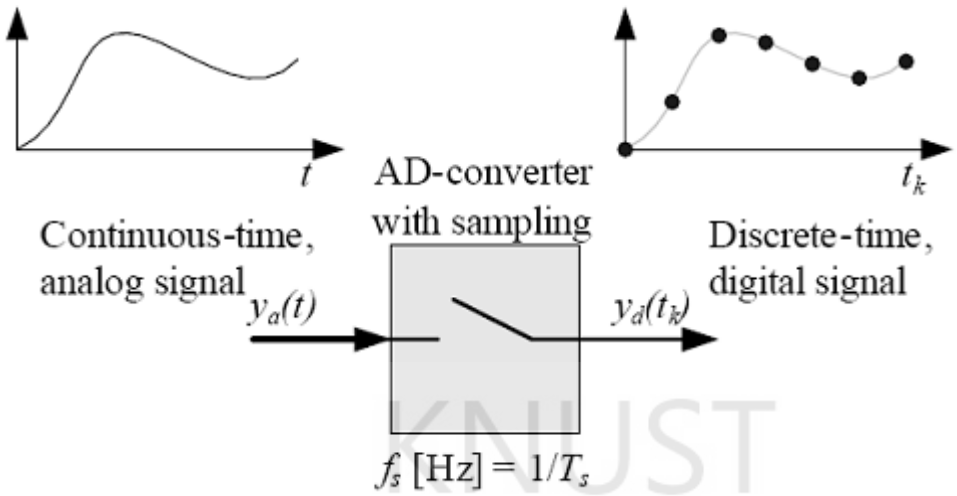


Figure 2.3: Sampling

The digital signal is represented internally in the computer by a number of bits. One bit is the smallest information storage in a computer. A bit has two possible values which typically are denoted 0 (zero) and 1 (one). Assume that $y_\alpha(t)$ has values in the range $[Y_{\min}, Y_{\max}]$ and that the AD-converter represents

y_α in the given range using n bits. Then y_α is converted to a digital signal in the form of a set of bits:

$$y_d \approx b_{n-1}b_{n-2}...b_1b_0 \quad \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Where each bit b_1 has value 0 or 1. These bits are interpreted as weights or coefficients in a number with base 2:

$$y_d = b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + ... + b_1 2^1 + b_0 2^0 \quad \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

b_0 is the LSB (least significant bit), while b_{n-1} is the MSB (most significant bit) [21, 22].

2.2 FILTERS

This sections considers filters and their types, emphasis will be on non-linear ordered statistics filters like the median filter and the adaptive median filter.

In signal processing, a filter is a device or process that removes from a signal some unwanted component or feature. Filtering is a class of signal processing, the defining feature of filters being the complete or partial suppression of some aspect of the signal. Most often, this means removing some frequencies and not others in order to suppress interfering signals and reduce background noise. However, filters do not exclusively act in the frequency domain; especially in the field of image processing many other targets for filtering exist. There are many different bases of classifying filters and these overlap in many different ways; there is no simple hierarchical classification. Filters may be:

- analog or digital
- discrete-time (sampled) or continuous-time
- linear or non-linear
- passive or active type of continuous-time filter
- Infinite impulse response (IIR) or finite impulse response (FIR) type of discrete-time or digital filter.

We will focus or direct our attention to non-linear filters since our work is geared towards is a filtering our impulse noise from a 2-dimensional grayscale image.

A nonlinear filter is a signal-processing device whose output is not a linear function of its input. Terminology concerning the filtering problem may refer to the time domain (state space) showing of the signal or to the frequency domain representation of the signal. When

referring to filters with adjectives such as "bandpass"," highpass", and "low pass" one has in mind the frequency domain. When resorting to terms like "additive noise", one has in mind the time domain, since the noise that is to be added to the signal is added in the state space representation of the signal. The state space representation is more general and is used for the advanced formulation of the filtering problem as a mathematical problem in probability and statistics of stochastic processes [23, 24].

In signal processing one often deals with obtaining an input signal and processing it into an output signal. At times the signal may be transmitted through a channel which corrupts it, resulting in a noisy output. As a consequence, the user at the output end has to attempt to reconstruct the original signal given the noisy one. When the noise is additive, i.e. it is added to the signal (rather than multiplied for example) and the statistics of the noise process are known to follow the gaussian statistical law, then a linear filter is known to be optimal under a number of possible criteria (for example the mean square error criterion, aiming at minimizing the variance of the error). This optimality is one of the main reasons why linear filters are so important in the history of signal processing [25].

However, in several cases one cannot find an acceptable linear filter, either because the noise is non-additive or non-gaussian. For example, linear filters can remove additive high frequency noise if the signal and the noise do not overlap in the frequency domain. Still, in two-dimensional signal processing the signal may have important and structured high frequency components, like edges and small details in image processing. In this case a linear lowpass filter would blur sharp edges and yield bad results. Nonlinear filters should be used instead.

Nonlinear filters locate and remove data that is recognised as noise. The algorithm is 'nonlinear' because it looks at each data point and decides if that data is noise or valid signal. If the point is noise, it is simply removed and replaced by an estimate based on surrounding data points, and parts of the data that are not considered noise are not modified at all. Linear filters, such as those used in bandpass, highpass, and lowpass, lack such a decision capability and therefore modify all data. Nonlinear filters are sometimes used also for removing very short wavelength, but high amplitude features from data. Such a filter can be thought of as a noise spike-rejection filter, but it can also be effective for removing short wavelength geological features, such as signals from surgical features [26].

2.3 MEDIAN FILTERS

In image processing, several filtering algorithms belong to a category called windowing operators. Windowing operators use a window, or neighborhood of pixels, to calculate their output [27]. For example, windowing operator may perform an operation like finding the average of all pixels in the neighborhood of a pixel. The pixel around which the window is found is called the *origin*. Figure 2.4, below, shows a 3 by 3 pixel window and the corresponding origin.

_	_	_
_	origin	_
_	_	_

Figure 2.4: Pixel window and Origin

The work for this project is based on the usage of image filtering algorithms (Median filters) using these pixel windows to calculate their output. Although a pixel window may be of any size (mostly odd number) and shape, a square 3x3 size was chosen for this application because it is large enough to work properly and small enough to implement efficiently on an image.

All the Median Filters are rank order filter, which are usually common filtering algorithm in image processing systems. Median Filters are nonlinear filter, so while it is easy to develop, it is difficult to understand its properties. It offers several useful effects, such as smoothing and noise removal [28]. Median filters are very useful in salt -and-pepper noise filtering. Since the rank order filter uses no arithmetic, a mathematical description is difficult to represent efficiently [29].

2.3.1 THE STANDARD MEDIAN FILTER

The best-known order-statistics filter is the median filter, which as its name implies, replaces the value of a pixel by the median of the gray levels in the neighborhood of that pixel:

$$\bar{f}(x,y) = \underset{(s,t)\varepsilon S_{xy}}{median}\{g(s,t)\} \ldots\ldots\ldots\ldots (3)$$

The original value of the pixel is included in the computation of the median. Median filters are quite popular because, for certain types of random noise, they provide excellent noise-reduction capabilities, with considerably less blurring than linear smoothing filters of similar size. Median filters are particularly effective in the presence of both bipolar and unipolar impulse noise.

The median filter is a non-linear ordered statistic digital filtering technique which is normally used to reduce noise drastically in an image. It is one of the best windowing operators out of the many windowing operators like the mean filter, min and max filter and the mode filter.

The simple idea is to examine a sample value of the input signal and decide if it is representative of the signal. Due to this, the median filter often does a better job than the boxcar filtering technique with regard to preserving useful detail in an image [30, 31].

The median filter filters each pixel in the image in turn and its nearby neighbours are used to decide whether or not it is representative of its surroundings. Normally, instead of replacing the pixel value with the mean of neighboring pixel values, median filter replaces it with the median of those values. That is, the values from the surrounding neighbourhood are first sorted into numerical order, and then the value of the pixel in question is replaced with the middle (median) pixel value. The neighbourhood is referred to as the *window*. The window can have various shapes centered on the target pixel. The square is a typical shape chosen for windows defined for 2D images. It should be noted that under normal circumstances the median filter, is performed using a window containing an odd number of pixels. If the neighbourhood under consideration consists of an even number of pixels, the median value selected as the output is the average of the two middle pixel values. The figure below illustrates an example of how the median filter calculation is performed in the window.

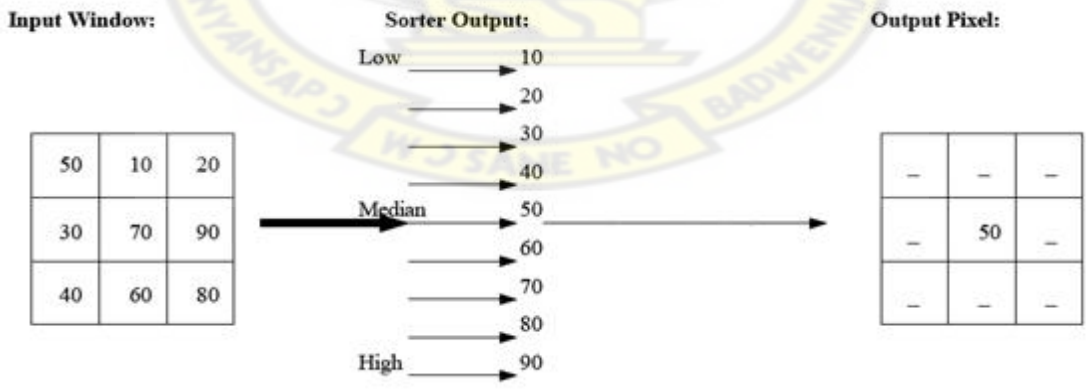


Figure 2.5: A graphical depiction of the median filter operation

This filter works by analyzing the neighborhood of pixels around an origin pixel like in the diagram above, for every valid pixel in an image. For this case, a 3×3 window, of pixels is used to calculate the output. For every pixel in the image, the window of neighboring pixels is found. As shown in the example above, the pixel values in the window are sorted in ascending order and the median value is chosen, in this case the median value is 50. Next, the pixel in the output image corresponding to the origin pixel in the input image is replaced with the value specified by the filter order. The value in the origin which is 70 is replaced by 50.

One of the advantages of median filter over the other rank order filters especially the mean filter, is that the median value is a more robust average than the mean value; the median value will not be affected significantly by one very unrepresentative pixel in neighbourhood. The median value of the surrounding pixels is most likely to be the value of one of the pixels in the neighbourhood within the window. Thus the median filter is least likely to create new unrealistic pixel values especially when the filter is working in transition zones. For this reason, the median filtering technique is much better than the mean filtering technique in terms of preserving sharp edges [32].

2.3.2 THE ADAPTIVE MEDIAN FILTER

The median filter discussed performs well as long as the spatial density of the impulse noise is not large (as a rule of thumb, P_a and P_b less than 20%). It is shown in this section that adaptive median filtering can handle impulse noise with probabilities even larger than these. An additional benefit of the adaptive median filter is that it seeks to preserve detail while smoothing non-impulse noise, something that the "standard" median filter does not do. As in all the nonlinear ordered statistics filters in literature, the adaptive median filter also works in a rectangular window area S_{xy}, unlike those filters, however, the adaptive median filter

changes (increases) the size of S_{xy}, during filter operation, depending on certain conditions listed in this section. Keep in mind that the output of the filter is a single value used to replace the value of the pixel at (x, y), the particular point on which the window S_{xy}, is centered at a given time. Consider the following notation [33]:

$z_{\min}$ = minimum gray level value in S_{xy}
$z_{\max}$ = maximum gray level value in S_{xy}
z_{med} = median of gray levels in S_{xy}
z_{xy} = gray level at coordinates (x, y)
$S_{\max}$ = maximum allowed size of S_{xy}.

The adaptive median filtering algorithm works in two levels, denoted level A and level B, as follows:

Level A: $A1 = z_{\text{med}} - z_{\min}$
$A2 = z_{\text{med}} - z_{\max}$
If $A1 > 0$ AND $A2 < 0$, Go to level B
Else increase the window size
If window size $\leq S_{\max}$ repeat level A
Else output z_{xy}.

Level B: $B1 = z_{xy} - z_{\min}$
$B2 = z_{xy} - z_{\max}$
If $B1 > 0$ AND $B2 < 0$, output z_{xy}
Else output z_{med}.

The key to understanding the mechanics of this algorithm is to keep in mind that it has three main purposes: to remove salt-and-pepper (impulse) noise, to provide smoothing of other noise that may not be impulsive, and to reduce distortion, such as excessive thinning or thickening of object boundaries. The values $Z_{\max}$ and $Z_{\min}$ are considered statistically by the algorithm to be "impulselike" noise components even if these are not the lowest and highest possible pixel values in the image. With these observations in mind, we see that the purpose of level ***A*** is to determine if the median filter output Z_{med}, is an impulse (black or white) or

not. If the condition, $Z_{\min} < Z_{med} < Z_{\max}$ holds, then Z_{med} cannot be all impulse for the reason mentioned in the previous paragraph. In this case, we go to level *B* and test to see if the point in the center of the window Z_{xy}, is itself an impulse (recall that Z_{xy} is the point being processed). If the condition $B1 > 0$ AND $B2 < 0$ is true, then $Z_{\min} < Z_{med} < Z_{\max}$ and Z_{xy} cannot be an impulse for the same reason that Z_{med} was not. In this case, the algorithm outputs the unchanged pixel value Z_{xy} [34]. By not changing these "intermediate-level" points, distortion is reduced in the image. If the condition $B1 > 0$ AND $B2 < 0$ is false, then either $Z_{xy} = Z_{\min}$ or $Z_{xy} = Z_{\max}$. In either case, the value of the pixel is an extreme value and the algorithm outputs the median value Z_{med}, which we know from level A that is not an impulse. The last step is what the standard median filter does. The problem is that the standard median filter replaces every point in the image by the median of the corresponding neighborhood. This causes unnecessary loss of detail. Continuing with the explanation, suppose that level A does find an impulse (i.e. it fails the test that would cause it to branch to level B).The algorithm then increases the size of the window and repeats level A. This looping continues until the algorithm either finds a median value that is not an impulse (and branches to level B), or the maximum window size is reached. If the maximum window size is reached, the algorithm returns the value of Z_{xy}. Note that there is no guarantee that this value is not an impulse. The smaller the noise probabilities P_a and/or P_b are, or the larger $S_{\max}$ is allowed to be, the less likely it is that a premature exit condition will occur. This is plausib1e. As the density of the impulses increases, it stands to reason that we would need a larger window to "clean up" the noise spikes. Every time the algorithm outputs a value, the window $S_{\max}$ is moved to the next location in the image. The algorithm then is reinitialized

and applied to the pixels in the new location. As indicated in the median filtering algorithm, the median value can be updated iteratively using only the new pixels, thus reducing computational overhead [35, 36].

2.4 SHORTCOMINGS OF THE STANDARD AND ADAPTIVE MEDIAN FILTERS

The standard median filter is one of the best nonlinear ordered statistics filters. The standard median (SM) filter is a simple nonlinear smoother that can suppress noise while retaining sharp sustained changes (edges) in signal values. It is particularly effective in reducing impulsive-type noise [1]. The output of SM filter at a point is the median value of the input data inside the window centered at the point. Although the standard median filters performs well in the presence of impulse noise (salt and pepper) it has some important shortcomings that has brought about the modification of various types of median filters.

An important shortcoming of the median filter is that its output is always constrained, by definition, to be the median value in the window. In a k×k window, if the number of polluted pixels is greater than the value($\frac{k\times k}{2}$), then the median computed will be an impulse and the noise will not be removed from the window. One of the remedies to this shortcoming is to increase the size of the window to improve the filtering efficiency [37]. The figure below explains the situation when the impulse noise pollutes more than half the pixels in the window.

0	50	0
30	0	40
0	60	0

Figure 2.6: 3×3 Window polluted by impulse noise (pepper noise)

In the 3×3 window, the impulse noise present in the window is greater than the value($\frac{k\times k}{2}$). Thus sorting the pixel values in the window will result in zero being the median value.

0	0	0	0	0	30	40	50	60

Figure 2.7: showing the sorted list of the 3×3 window

The 5th value which is the median of the nine elements in the window is zero. This implies that the noise in the window still remains. The median is the 50th percentile hence the median value from the sorted list will be an impulse. Since the median replaces the center value of the active window. This implies that the noise will not be filtered out of the window. This happens if and only if the density of the noise signal is greater than or equal to 50 percent of the pixel values in the window [38, 39].

On the other hand, since the centre value replaced by the median is not tested to find out if it is an impulse or not. A clean pixel value or a fine pixel value can be replaced with an impulse in the window. All this accounts for the occasional poor performance of the median filter. This is usually the case when the intensity of the impulse noise is very high in the window. This can be illustrated in the figure below.

255	65	255
255	50	75
255	55	255

Figure 2.8: 3×3 Window polluted by impulse noise (salt noise)

50	55	65	75	255	255	255	255	255

Figure 2.9: showing the sorted list of the 3×3 window

Another important shortcoming of the standard median filter is when the number of the pixels of edge features is very small (that is, when the value is far less than($\frac{k\times k}{2}$)), the edge pixels in this case is mostly replaced by the median value, which changes the resolution of the image. From literature the solution to this problem is by reducing the size of the window depending on the window size chosen.

0	75	85
110	200	100
210	95	255

Figure 2.10: 3×3 Window polluted by impulse noise (salt and pepper noise)

In this situation the bigger values in the 3×3 window represents the image edge features. It can be seen from the window that the edge feature present in the window is less that the value($\frac{k\times k}{2}$). The replacement of all the image edge features introduces another form of distortion into the image.

0	75	85	95	100	110	200	210	255

Figure 2.11: showing the sorted list of the 3×3 window

These are the major deficiencies of the standard median filter that will be considered in this work. Though the standard median filter has deficiencies it outperforms most of the nonlinear

order rank filters like the mean filter, the maximum and minimum filter and the mode filter in the reduction of impulse noise (salt and pepper noise) in an image.

The basic difference between the standard median filter and the adaptive median filter is that the size of the window in the standard median filter is fixed whiles the window size of the adaptive median filter varies depending on whether the computed median for the window is an impulse or not. Thus, if the median is an impulse the window is extended and all the values in the larger window are resorted afresh.

There are some deficiencies in the mode of operation of the algorithm of the adaptive median filter. It is known that the condition $Z_{\min} < Z_{med} < Z_{\max}$ is always satisfied. When the condition $Z_{\min} < Z_{med} < Z_{\max}$ is satisfied, then go to level B. Otherwise, when $Z_{\min} = Z_{med} < Z_{\max}$ or $Z_{\max} = Z_{med} > Z_{\min}$ the algorithm is kept in level A and the window size is increased. When $S_{\max}$ is reached, the algorithm outputs the value of Z_{xy}.

However, it has some deficiencies. When an imposed limit to the window size, S_{xy}, is reached and while the selected median is still an impulse, then the impulsive noise remains in that window of the image. The adaptive median filter achieves good results in most cases, but even so, computation time is proportional to the degree of corruption of the image being filtered.

0	130	0	155	0
125	0	165	0	135
0	255	0	145	0
255	0	140	0	125
0	156	0	160	0

Figure 2.12: 5×5 Window polluted by impulse noise (salt and pepper noise)

In this situation, the adaptive median filter algorithm increases the window size until it reaches the $S_{\max}$.

Negative impulse appears as black point (pepper noise) in an image and positive impulse appears as white point (salt noise). When the black image in a window is corrupted by impulse noise, there are black and white pixels. For an 8-bit image, it means the pixel values are 0 or 255. But most of the pixel values are 0, so $Z_{\min} = Z_{med} = 0,\ Z_{\max} = 255$. Even though the window size is increased until the maximum window size is reached noise is still present in the window, meaning $Z_{\min} = Z_{med} = 0,\ Z_{\max} = 255$. In this case, the output value will be 0. But, according to the algorithm, the output value is equal to Z_{xy}. The output Z_{xy} may be any value between 0 and 255. So the salt noise cannot be filtered in the black background of corrupted image. For the same reason, the pepper noise cannot be removed in the white background. If the output value is changed from Z_{xy} to Z_{med} the noise still remains in the window [40, 41].

2.5 STRATEGIES TO OVERCOME THE SHORTCOMINGS

The improved median filter is developed and implemented to handle the problems faced by the standard median filter and to improve the mode of operation of the adaptive median filter to achieve effective results. When the number of corrupted pixels by impulse noise is greater than $\frac{k \times k}{2}$, in a $k \times k$ window, the standard median filter outputs an impulse pixel value as the median. The adaptive median filter increases the size of the window, if the impulse noise still present is greater than or equal to the 50^{th} of the pixels in the window when $S_{\max}$ is reached, then Z_{med} is an impulse and hence if Z_{med} is compared to Z_{xy} the algorithms terminates,

meaning the noise is not remove from the window. The improved median filter algorithm uses the switching median technique, where the median from the window is tested to verify if it satisfies the condition $X_{\min} < X_{med} < X_{\max}$. The switching median technique is applied when the median chosen from the sorted list is an impulse. The median switches until an effective median if found that satisfies the condition $X_{\min} < X_{med} < X_{\max}$. The switching technique is more explained in section 3.3.

The other situation is when all the pixels values are corrupted by impulse noise. The standard median filter has no mechanism put in place to check whether the median is an impulse or not. The adaptive median filter equally does not generate any good results since it output is dependent on the pixels values of the window. The window size is increased until the maximum window size is reached. Since all the pixels are corrupted the noise still remains. The improved median filter applies the switching technique and if an effective median is not found, the algorithm computes $X_{med} = \frac{K1 + K2}{4}$ as the effective median. The optimal substructure property is applied to obtain an optimal median value.

The improved median filtering algorithm also uses the 'divide and conquer' programming technique to break the algorithm into levels, so that each level can produce an efficient solution. The improved median filter also carries a test on every pixel value in the window and replaces the polluted pixel value with the median/effective median.

The adaptive median filter is complex and time consuming. The time spend in the selection of the median from an image polluted with impulse noise is dependent on the intensity of the noise present in each window selected. The improved median filtering algorithm is simple and produces efficient result. The greedy algorithm is applied in the selection of the effective median by switching until a value is selected that is not an impulse.

CHAPTER THREE

3.0 DESIGN AND DEVELOPMENT OF THE IMPROVED MEDIAN FILTERING ALGORITHM

3.1 PREPROCESSING DEVELOPMENT PHASE

In the proposed method the size of the window is fixed, however, the (effective) median may be different from the value at the middle of the sorted pixel values. The proposed effective median filter is designed to diminish the problem faced by the standard median filter and reduce that of the adaptive median filter. As with the standard median technique, the window is chosen to cover a $k \times k$ array of pixels such that

$$k^2 = 2n + 1 \Longrightarrow n = \frac{k^2 - 1}{2}$$

Where for integer$n > 0$, $k = 3, 5, 7, \ldots$. In the proposed technique of filtering, as in standard median filter, the pixels are sorted and the median is selected from a sorted list of the current window.

The improved median filter algorithm is partitioned into three stages, which we call levels A, B and C processing, respectively. The algorithm starts from a preprocessing step before the process continues to the three levels. In the preprocessing stage the window is selected and the values in the window are sorted and save in the sorted list. The minimum pixel value, X_{min}, and the maximum pixel value, X_{max}, are compare with impulse values K1 and K2, respectively, where K1=0 and K2=255. If X_{min} = K1 or if X_{max} = K2 then the window has impulse noise and processing proceeds through all levels A, B, and C. Otherwise processing proceeds through only level C.

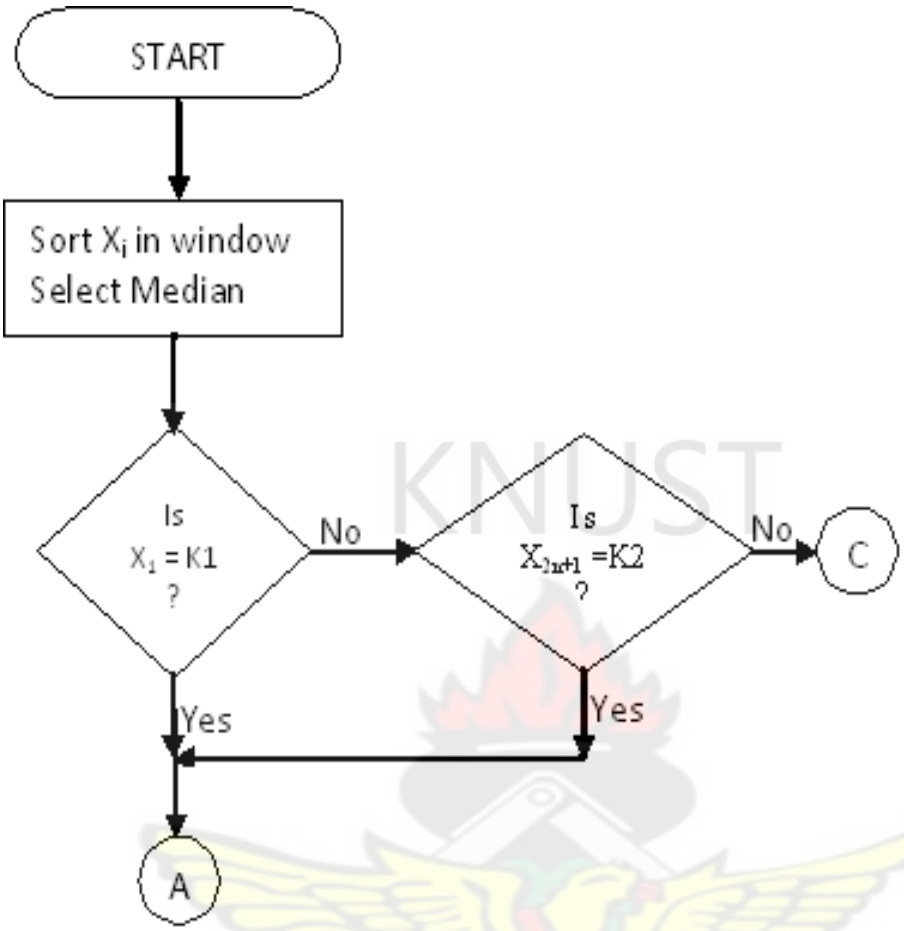


Figure 3.1: Checking window for presence of impulse noise (Preprocessing stage).

3.2 LEVEL A DEVELOPMENT PHASE

The improved median filtering algorithm adapts the local processing technique from the standard median filtering method and the adaptive median filtering method, where the entire image data is divided into small segments called windows. This is done to enable easy and efficient processing in each window. The Overlapping-moving window technique is applied in the dividing of the image data into window for processing. This process begins from the first pixel of the image and applies the technique by dividing the entire image in a 3×3, or

5×5 or 7×7. The improved median technique is applied to a fixed 3×3 window for the processing of the two dimensional images.

The improved median filter selects the median from the window by first sorting the pixels in ascending order or descending order. This process of choosing the median form each window and using it as a robust value and fair representation of the pixels in the window is a greedy technique that is employed. The greedy algorithm is an optimal problem solving technique that makes the choice that looks best at the moment. That is, it makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution. The greedy technique is applied in the choosing of the median value in the window [43, 44].

The algorithm proceeds to carry out a series of test after the median value is chosen to verify if it is polluted by noise or it is noise free. When the test carried out on the median proves that it is not an impulse or not polluted then the algorithm proceeds to level B and the algorithm continues. On the other hand, the median value is deemed to be an impulse, if and only if the median value is equal to the maximum value or the minimum value.

Impulse noise (salt and pepper) appears as very dark shots in bright regions and very bright shots in dark regions of the image. The very bright shot appears in an image by converting some of the pixels in a dark region as 255, which is the maximum value of an 8-bit 2D image and also converting some of the pixels to be very dark shots having the value of 0 (which is the minimum value) in the bright regions. The median computed, if however is deemed to be an impulse then the value is equal to the maximum or minimum value.

In this first situation, when the median value is equal to the minimum value of the same window, then the switching median technique is applied in the computation of the effective median. The process selects the next higher ranked value from the sorted list. A test is carried on the value selected to verify if it is an impulse or not. When the value selected satisfies the condition $X_{\min} < Select\_value < X_{\max}$ then the value is used as the effective median else the

condition is not met then, the next higher ranked value after the value that is immediate after the actual median to its right is selected. The value selected is also test to find out if it satisfies the condition. The value is used as the effective median if it satisfies the condition else the process moves on to select the next higher ranked value. This process continues to choose values between the actual median value and the maximum until a value is picked that satisfies the condition $X_{\min} < Select\_value < X_{\max}$. Otherwise if the switching median technique proceeds until the maximum is reached and the condition is not satisfied then a value $X_{med} = \frac{K1 + K2}{4}$ is computed and used as the effective median for that particular window.

On the other hand, when the median value is equal to the maximum value, the switching median technique is applied in the reverse direction in selecting the effective median value. This time around the next lower ranked value before the median, towards the left hand side is selected in the sorted list. The value is equally tested to verify if it satisfies the condition $X_{\min} < Select\_value < X_{\max}$. When the condition is met then the value is used as the effective median value else the next lower ranked value is selected and the same condition is carried out on the value selected to also verify if it satisfies the condition. This process continues to select values between the actual median and the minimum value. The condition is carried out on each value selected that is between the median and the minimum value. If the minimum value is reached and the condition still is not met then the value $X_{med} = \frac{K1 + K2}{4}$ is computed and used as the effective median.

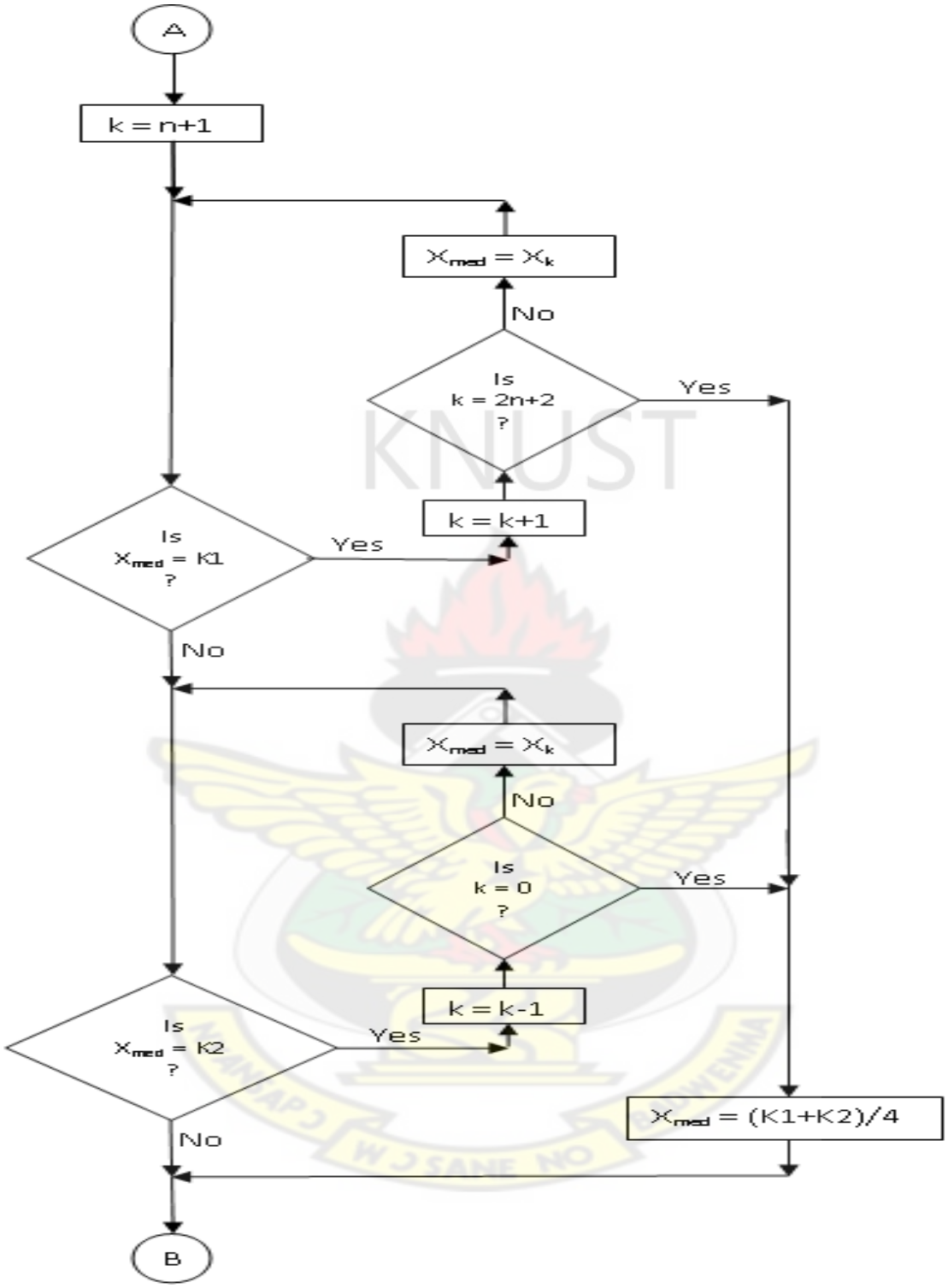


Figure 3.2: Level A Processing

3.2 LEVEL B DEVELOPMENT PHASE

The algorithm continues in level B after the computation of the median/effective median. The main task of the improved median filtering algorithm operation takes place in this section. The divide and conquer programming technique adapted in the design of the algorithm. Divide and conquer algorithm was employed to divide the algorithm into levels for easy and efficient solution to be developed.

The Pixels that are polluted by impulse noise (salt and pepper) in the window are detected and removed in this section. A recursive algorithm is design in this section, the program begins by selecting the first value in the window and assessing the value with a simple detection technique to verify if any of the pixels is an impulse or not. At this stage of the algorithm, two decisions are considered when the pixel value is an impulse and when the pixel value is not an impulse. In the first case if the pixel value is an impulse, then the value is removed and replaced with the median/effective median value computed for in level A. The process continues to the next pixel value and the same test is carried on the value, this continues recursively until the last value has been tested. On the other hand, when the pixel value is tested and the result indicates that the value is not an impulse then the value is left untouched and the process continues.

While the improved method uses the recursive process to scan through all the pixels in the window, in the standard median filter only the middle value of the window is replace, however a recursive function is not used. The effectiveness and efficiency of the algorithm to generate an optimal noise free image or output image is as a result of level B processing. The time spend in a window is relative to the number of polluted pixels, hence if the number of polluted pixels by impulse noise in the window is more than half the total number of pixels (that is in a K×K window) then much time will be spend in level B replacing the noisy pixels.

The optimal substructure property of the greedy algorithm is employed to obtain an optimal and efficient solution. The property is a strategy that is used by the greedy algorithm and dynamic programming to achieve an optimal solution to the problem which contains optimal solutions to the sub-problems. The main differences in the implementation of the standard median filtering algorithm and the improved median filtering algorithm is that the standard median filter replaces only the center pixel of the window but the improve median filter replaces any pixel that is polluted by impulse noise.

After all the polluted pixels have been replaced, the window pixels are resorted and the maximum and minimum values are selected. The process then continues to level C and further processing takes place there.

The flowchart below shows the pictorial and systematic application of level B of the improved median filtering algorithm.

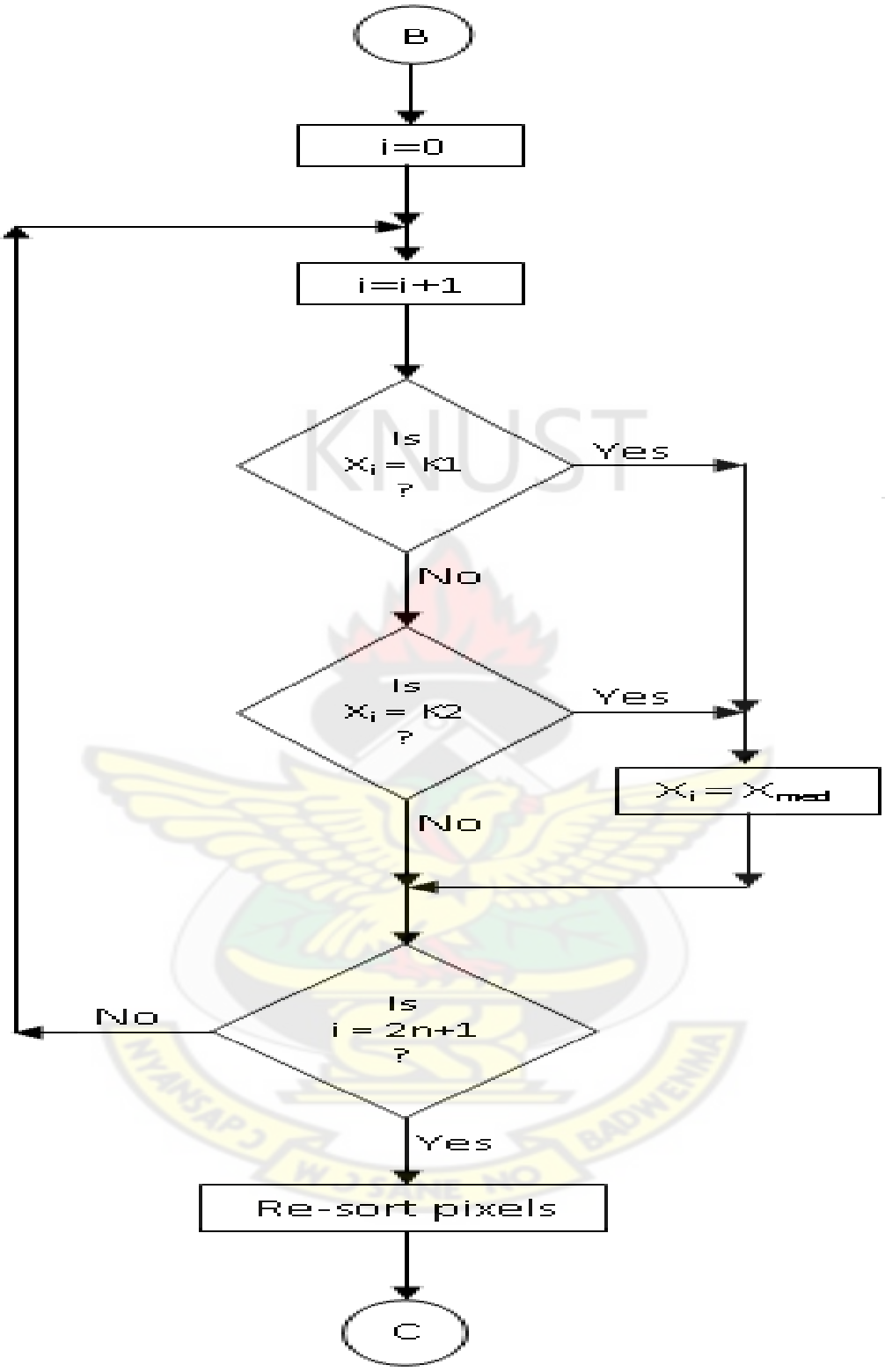


Figure 3.3: Level B Processing

3.3 LEVEL C DEVELOPMENT PHASE

The process continues in this level, in the later part of Level B the pixels in the window are resorted the pixels that are sorted are saved in the sorted list in a virtual memory. The resorting of the pixels is to reorder the pixels, so that a detector is in level C to test the maximum and minimum values to verify if they are distorted or not. This part of the algorithm main purpose is to remove distortion by using the window pixel-to-pixel neighborhood relationship. Any pixel picked in the image has a relationship with its neighbor. The distortion in the image causes a diversion of some of the pixels in the window from the neighborhood relationship. The pixels that are usually affected by the distortions makes the pixels deviate from the relationship which appears as a value higher than the original undistorted maximum value or the original undistorted minimum value.

At this level distortions, if present, are removed. First the differences w_i are computed thus

$$w_i = X_{i+2} - X_{i+1}$$

for $i = 1, 2, \dots, 2n-2$.

Next

$$D = \max_{1 \le i \le 2n-2} w_i$$

is computed followed by the following computations for the extreme values at this stage of processing:

$$X_1 = \begin{cases} X_{med}\ , if\ X_2 - X_1 > D \\ X_1, \qquad Otherwise \end{cases}$$

$$X_{2n+1} = \begin{cases} X_{med}\ , if\ X_{2n+1} - X_{2n} > D \\ X_{2n+1}, \qquad Otherwise \end{cases}$$

The flow charts of these levels of processing are shown in Figures 3.9, 3.10, and 3.11, respectively.

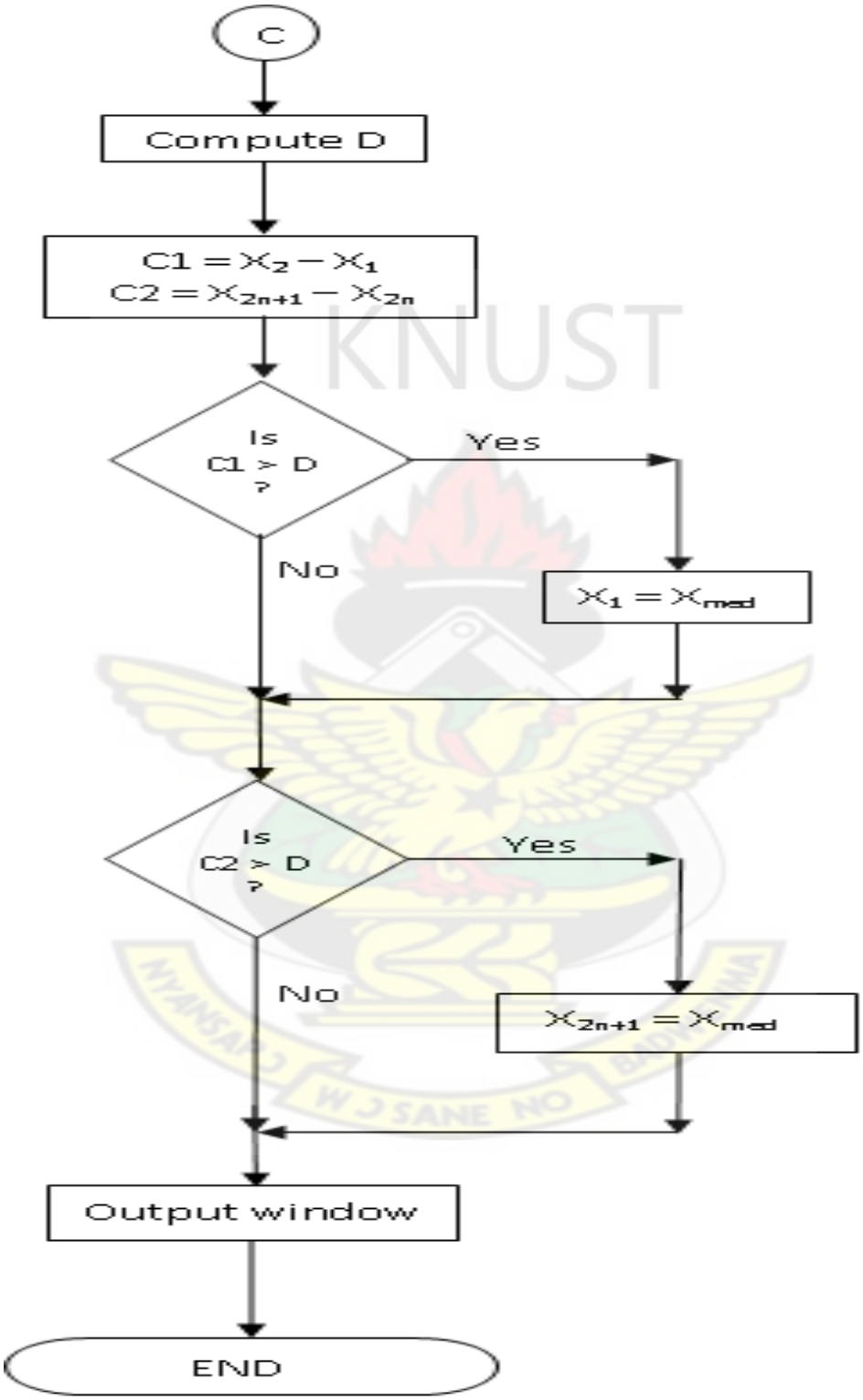


Figure 3.4: Level C Processing

3.5 ILLSTRATION OF THE IMPROVED MEDIAN FILTERING ALGORITHM

Assume the current window is as shown in Figure 3.12. Two pixels in the window have been corrupted by impulse noise. These are the values 0 and 255.

0	50	2
90	255	70
80	60	250

Figure 3.5: A noisy window.

Values in the window are sorted in ascending order top-to-bottom. The median value 70 is picked from the values in the window. This is shown in Figure 3.13.

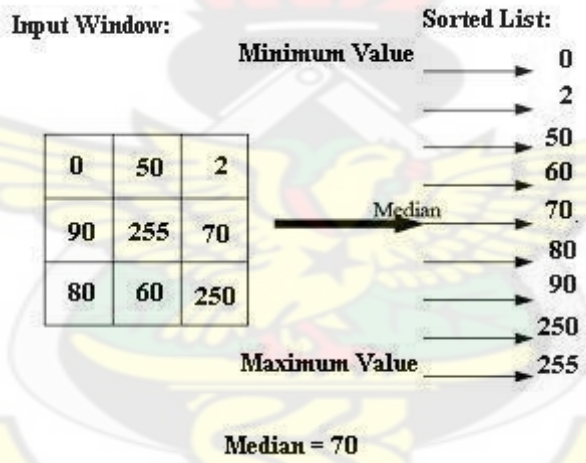


Figure 3.6: Pixels sorted by value and median selected at rank=5.

At Level A processing median value 70 is confirmed (as shown in Figure 3.14) as the effective median X_{med}, since $K1 < median < K2$.

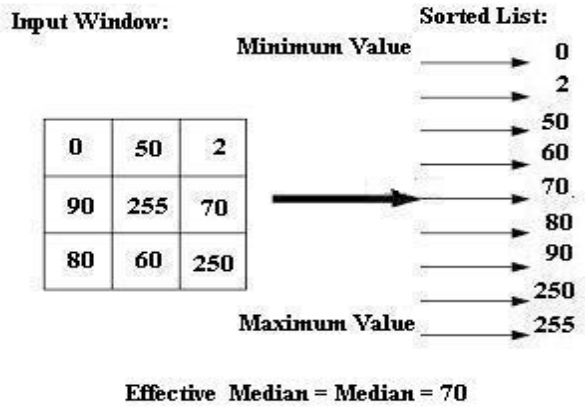


Figure 3.7: Median confirmed as the effective median.

Proceeding to Level B, the two impulses are detected and switched with the effective median as shown in Figure 3.15. Finally, at this level, the pixels are re-sorted and processing continues to the next level.

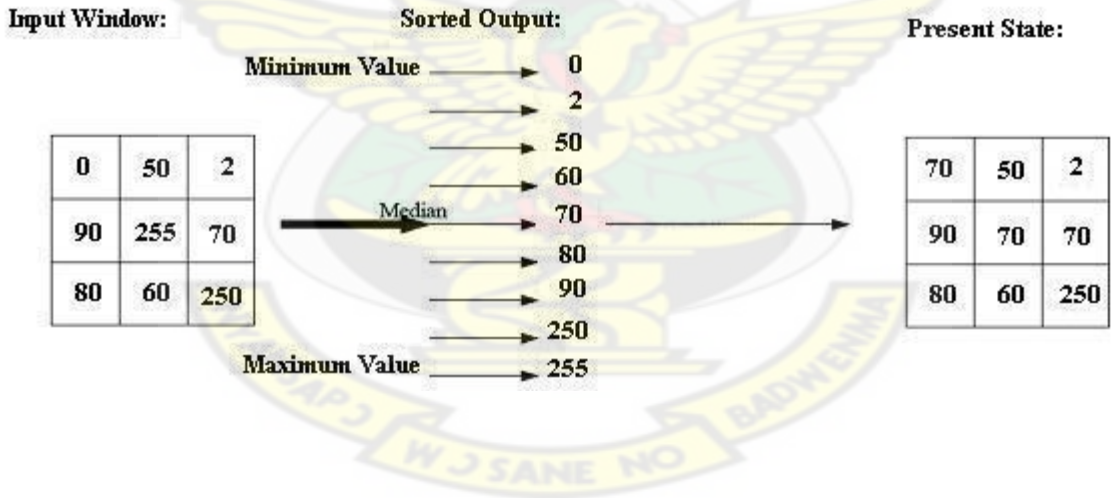


Figure 3.8: Impulses switched with the effective median.

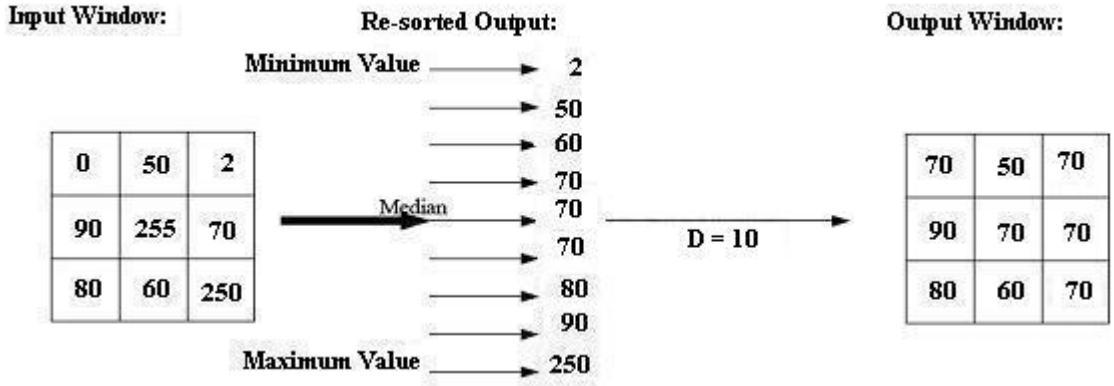


Figure 3.9: D is computed and used to detect distortions the extremes of the re-sorted list.

At Level C, first D is computed by considering the sub-list starting from the second value and ending at the eighth value. D is found to be equal to 10. Since 50-2>10, the value 2 is switched with the effective median. Also, 250-90>10 and so the value 250 is switched with the effective median. The output window in Figure 3.7 shows these replacements.

CHAPTER FOUR

4.0 IMPLEMENTATION AND TESTING OF THE IMPROVED MEDIAN FILTERING ALGORITHM

4.1 IMPLEMENTATION OF THE IMPROVED MEDIAN FILTERING ALGORITHM

The underlying source code for the implementation of the improved median filtering technique for impulse noise reduction is provided in the appendices. This code listing however only contains the important function and script files in the program that demonstrates the design of the improved median filtering software.

The PC software program MATLAB was used to develop the improved algorithm so that its operation could be verified and tested. 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 [45, 46]. The MATLAB programming language was used because it handles very large dimensions of matrices and speedup mathematical operations on matrices.

The implementation of the improved median filtering algorithm is used to reduce noise in the image, like the adaptive median filter and the standard median filter. The simple idea is to examine the pixel values in the selected window of the input signal and replace the noisy and distorted or blurred pixels with the median/effective median.

The program works by using a moving fixed 3 × 3 window of pixel neighborhoods. For every movement the operation of the improved algorithm is applied. The 3x3 fixed window was used to reduce the complexity of the process and also to reduce the time required to sort the pixels values in ascending order. The proposed algorithm achieves very good results when the impulse noise introduced in the image is less than or equal 50%.

The relationship between the pixel in a 3 × 3 window are primary, they have a great influence on each other and for that matter when any of them is replaced with the other the distortion that occurs is very negligible. When the window size grows to a 5 × 5 and 7 × 7 the relationship between the pixel at the far end of the window and the middle pixel value has a secondary relationship in a 5 × 5vwindow and a tertiary relationship with the 7×7 window. Hence the pixel value at the diagonal end in a large window does not have a direct relationship with the middle value and hence may cause a great distortion. The improved median filter adapts the overlapping window techniques from the standard median filter and the adaptive median filter. For loops are used to code the overlapping window, the code is shown below;

```
for rows = ll:nrows-ul
   for cols = ll:ncols-ul

      window_ind = -ul:ul;
      region = I(rows+window_ind,cols+window_ind);
      centerpixel = region(ll,ll);
```

A virtual list is created with the window pixels. The pixels values are sorted in ascending order,

using the function *visort* . The code for the sorting of the pixel values is shown below;

```
function v = visort(v,N)
temp = v;

for i=1:N-1
   m = v(i);
```

```
    k = 1;

    for j = i+1:N
        if v(j) < m
            m = v(j);
            k = j-i+1;
        end
    end

    for j = 1:k-1
        v(i+j) = temp(i+j-1);
    end

    v(i) = m;

    for j=1:N
        temp(j) = v(j);
    end
end
```

The preprocessing step is the preliminary stage that conducts a test on the maximum and the minimum value. The minimum value is tested to verify if it is an impulse or not. If the test confirms that the minimum value is not a noisy pixel then the algorithm proceeds to test if the maximum value is also an impulse or not. If either the maximum value or the minimum value is an impulse then the algorithm proceeds to level A. If neither of them is an impulse then the

algorithm skips levels A and B and proceeds to level C meaning the window have no impulse noise.

The algorithm has three stages after the preprocessing stage. The first level A, computes the median/effective median, the replacement of noisy pixels by the median/effective take place in level B meaning the impulsive pixels are removed from the window and final in level C, smoothening of pixels, reduction of distortions and blurring in the window.

LEVEL A IMPLEMENTATION

The moving window technique adapted from the standard median filtering algorithm is used to select the 3×3 window for impulse noise filtering. The window selection process is show in the preprocessing stage. A function *mover_median(x, dim)* is written using MATLAB M-file to obtain the median from a created list. The list contains values from the active window. The code of the function *median(x, dim)* to find the median of the values is shown below;

```
function y = mover_median(x,dim)
nInputs = nargin;

if isempty(x)
  if nInputs == 1

    % The output size for [] is a special case when DIM is not given.
    if isequal(x,[]), y = nan(1,class(x)); return; end

    % Determine first nonsingleton dimension
    dim = find(size(x)~=1,1);
```

```
  end

  s = size(x);
  if dim <= length(s)
    s(dim) = 1;              % Set size to 1 along dimension
  end
  y = nan(s,class(x));

elseif nInputs == 1 && isvector(x)
  % If input is a vector, calculate single value of output.
  x = sort(x);
  nCompare = numel(x);
  half = floor(nCompare/2);
  y = x(half+1);
  if 2*half == nCompare       % Average if even number of elements
    y = meanof(x(half),y);
  end
  if isnan(x(nCompare))       % Check last index for NaN
    y = nan(class(x));
  end
else
  if nInputs == 1             % Determine first nonsingleton dimension
    dim = find(size(x)~=1,1);
  end
```

```
s = size(x);

if dim > length(s)          % If dimension is too high, just return input.
  y = x;
  return
end

% Sort along given dimension
x = sort(x,dim);

nCompare = s(dim);          % Number of elements used to generate a median
half = floor(nCompare/2);    % Midway point, used for median calculation

if dim == 1
  % If calculating along columns, use vectorized method with column
  % indexing.  Reshape at end to appropriate dimension.
  y = x(half+1,:);
  if 2*half == nCompare
    y = meanof(x(half,:),y);
  end

  y(isnan(x(nCompare,:))) = NaN;   % Check last index for NaN

elseif dim == 2 && length(s) == 2
  % If calculating along rows, use vectorized method only when possible.
```

```
% This requires the input to be 2-dimensional.   Reshape at end.
y = x(:,half+1);
if 2*half == nCompare
  y = meanof(x(:,half),y);
end

y(isnan(x(:,nCompare))) = NaN;   % Check last index for NaN

else

cumSize = cumprod(s);
total = cumSize(end);           % Equivalent to NUMEL(x)
numMedians = total / nCompare;

numConseq = cumSize(dim - 1);    % Number of consecutive indices
increment = cumSize(dim);        % Gap between runs of indices
ixMedians = 1;

y = repmat(x(1),numMedians,1);   % Preallocate appropriate type

% Nested FOR loop tracks down medians by their indices.
for seqIndex = 1:increment:total
  for consIndex = half*numConseq:(half+1)*numConseq-1
    absIndex = seqIndex + consIndex;
    y(ixMedians) = x(absIndex);
```

```
    ixMedians = ixMedians + 1;
  end
end

% Average in second value if n is even
if 2*half == nCompare
  ixMedians = 1;
  for seqIndex = 1:increment:total
    for consIndex = (half-1)*numConseq:half*numConseq-1
      absIndex = seqIndex + consIndex;
      y(ixMedians) = meanof(x(absIndex),y(ixMedians));
      ixMedians = ixMedians + 1;
    end
  end
end

% Check last indices for NaN
ixMedians = 1;
for seqIndex = 1:increment:total
  for consIndex = (nCompare-1)*numConseq:nCompare*numConseq-1
    absIndex = seqIndex + consIndex;
    if isnan(x(absIndex))
      y(ixMedians) = NaN;
    end
    ixMedians = ixMedians + 1;
```

```
    end
  end
 end

 % Now reshape output.
 s(dim) = 1;
 y = reshape(y,s);
end

%============================

function c = meanof(a,b)

c = a + (b-a)/2;
k = (sign(a) ~= sign(b)) | isinf(a) | isinf(b);
c(k) = (a(k)+b(k))/2;
```

An overall function called *Improvedmed_filt(I)* is written which is shown in the Appendices to test if the median is an impulse or not. The median value is used in the process if it is not polluted by impulse noise. On the other hand an effective median is calculated if the median is an impulse. For the median value to be an impulse then fifty percent or more of the pixels in the window are polluted by noise (impulse).

The code for the function *LEffmed* and function *REffmed* is shown in the Appendix to verify if the median value chosen from the created list is corrupted by impulse noise or not.

LEVEL B IMPLEMENTATION

Local processing technique is applied in the filtering of the impulse noise (salt and pepper) from the image. Most of the filtering of the noisy pixels takes place in this level. The main improved median filtering algorithm operation is implemented in this level. The function remove_noise is developed to remove the polluted pixel values in every 3×3 window selected in the image. The pixel values in the window is selected and tested to verify if the value selected is an impulse or not. If the value is not an impulse then it is left unchanged but on the other hand, the value selected is an impulse then it is replaced with the median or effective median value. This process removes almost all the impulse noise present in the window, blurring as a result of pixels polluted by impulse noise is reduce and also some minor distortion are corrected in this part of the algorithm.

```
function X[rows, cols]=remove_noise(min, max)
for rows = ll:nrows-ul
   for cols = ll:ncols-ul
      if X(rows, cols)= = min
         X(rows, cols)= REffect_med;
      elseif X(rows, cols)= = max
         X(rows, cols)= LEffect_med;
      else
         X(rows, cols)=X(rows, cols)
      end
   end
end
```

LEVEL C IMPLEMENTATION

The smoothening and removal of certain forms of distortion and blurring of the image is successfully handled in this level. A function is written in Matlab M-file called remove_distortions to handle the removal of distortion, blurring and to smoothening the image. A detection mechanism of identifying the distortion and blurring in the image window is designed. The neighborhood pixel-to-pixel relationship in the window is used to develop the detection mechanism. A value D is computed to check the maximum and minimum pixels values whether they have deviated from the neighborhood pixel-to-pixel relationship. D is the largest interval between adjacent pixels in the sorted list from the second value X_2 to the second to last value X_{n-1}. The value D is computed after all the pixels that are polluted have been replaced in the window and the window values are resorted in the list. The code of the function remove_distortions is shown below and in the overall program for the improved median filter.

```
function outbuf = remove_distortions_(inbuf)

numpixels = length(inbuf);

tbuf = inbuf;

for ii=eml.unroll(1:numpixels)
   if bitand(ii,uint32(1)) == 1
      tbuf = compare_stage1(tbuf);
   else
      tbuf = compare_stage2(tbuf);
```

```
    end
end

outbuf = tbuf;

end

function outbuf = compare_stage1(inbuf)
numpixels = length(inbuf);
tbuf = compare_stage(inbuf(1:numpixels-1));
outbuf = [tbuf(:)' inbuf(numpixels)];
end

function outbuf = compare_stage2(inbuf)
numpixels = length(inbuf);
tbuf = compare_stage(inbuf(2:numpixels));
outbuf = [inbuf(1) tbuf(:)'];
end

function [outbuf] = compare_stage(inbuf)

step = 2;
numpixels = length(inbuf);

outbuf = inbuf;
```

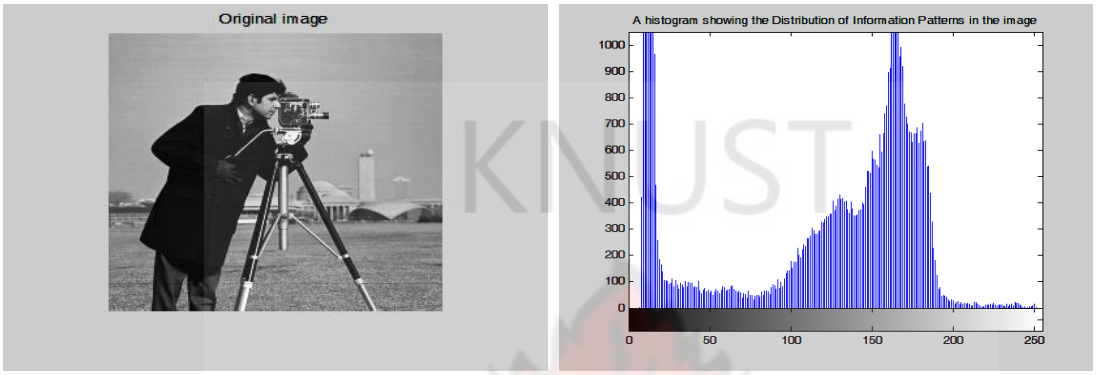
```
 for ii=eml.unroll(1:step:numpixels)
    t = compare_pixels([inbuf(ii), inbuf(ii+1)]);
    outbuf(ii) = t(1);
    outbuf(ii+1) = t(2);
end


end


function outbuf = compare_pixels(inbuf)
if (inbuf(1) > inbuf(2))
    outbuf = [inbuf(1), inbuf(2)];
else
    outbuf = [inbuf(2), inbuf(1)];
end
end
```


4.2 TESTING OF THE IMPROVED MEDIAN FILTERING ALGORITHM

In order to test the performance rate of this proposed algorithm experiments are performed at different noise levels ranging from 20% to 60% on two types of images. The two images cameraman and pout are two dimensional 8-bit grayscale images. Impulse noise of different percentages ranging from 20% to 60% is added to the two types of images. Intensive simulations were carried out on the two polluted images with different percentages of impulse noise (salt and pepper). The standard median filter, the adaptive median filter and the improved median filter are applied to the corrupted images by impulse noise. The result of

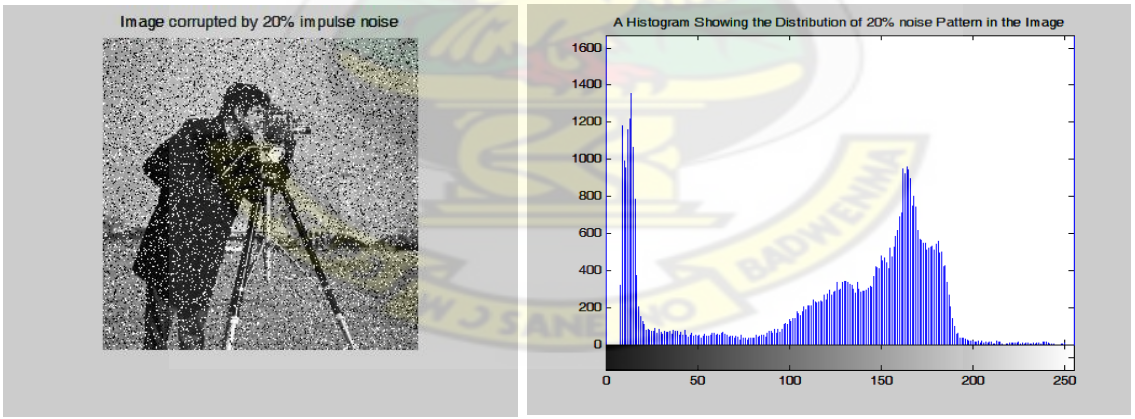
the application of the various types of median filters shows that the improved median filter achieves better results than the standard median filter and the adaptive median filter.

The outcome of the performances of the filtering operations is demonstrated by the output of the images shown below.



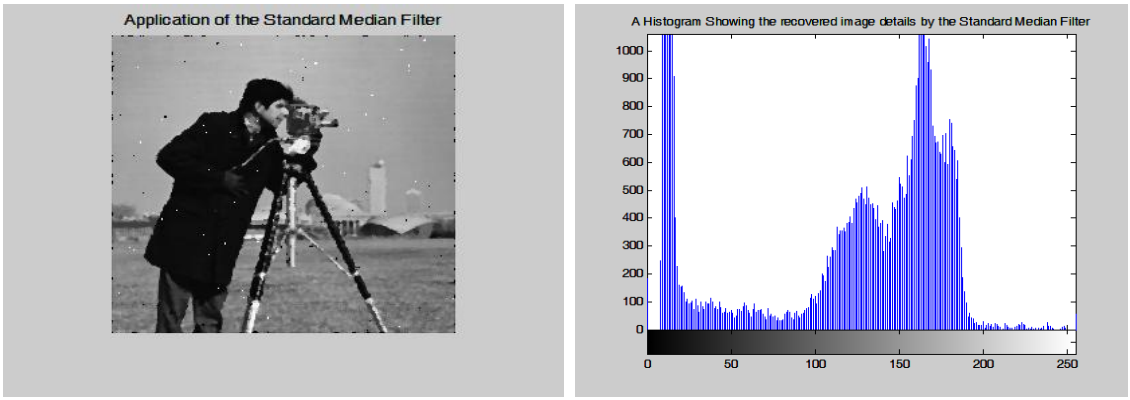
(i) (ii)

Figure 4.1: The cameraman image (i) and its grey level histogram (ii).



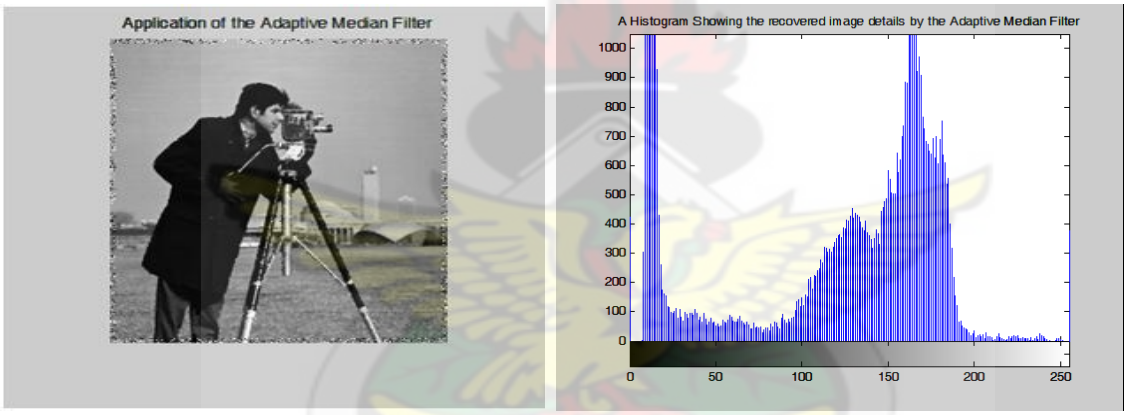
(iii) (iv)

Figure 4.2: The image corrupted by 20% impulse noise (iii) and its grey level histogram (iv).



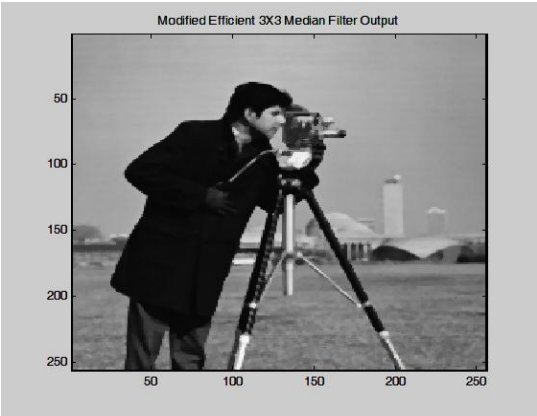
(v) (vi)

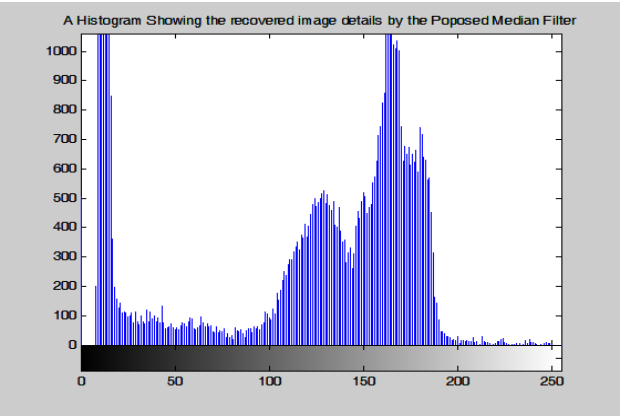
Figure 4.3: Application of the Standard Median filter to image corrupted by 20% impulse noise (v) and its grey level histogram (vi).



(vii) (viii)

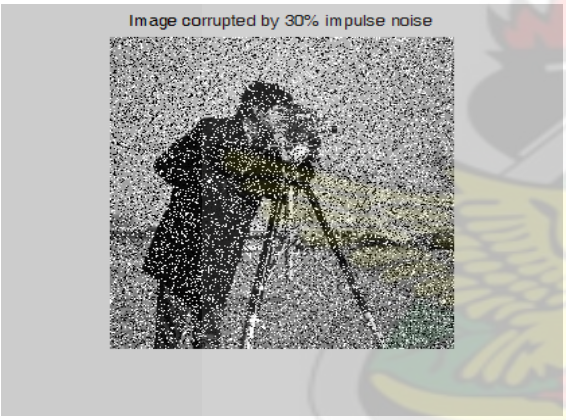
Figure 4.4: Application of the Adaptive Median filter to image corrupted by 20% impulse noise (vii) and its grey level histogram (viii).

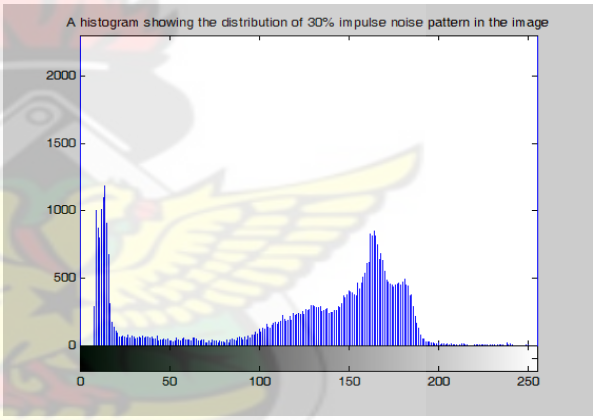




(ix) (x)

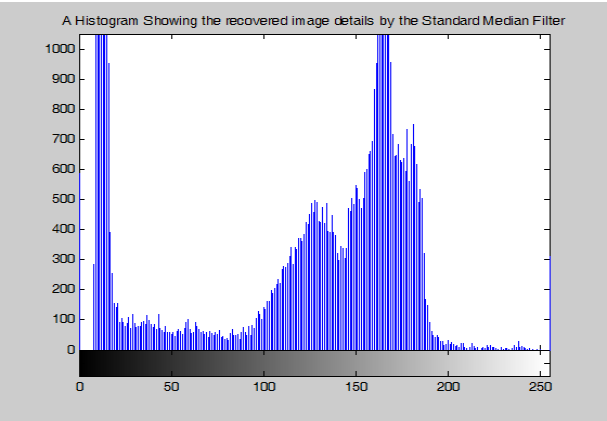
Figure 4.5: Application of the Improved Median filter to image corrupted by 20% impulse noise (ix) and its grey level histogram (x).





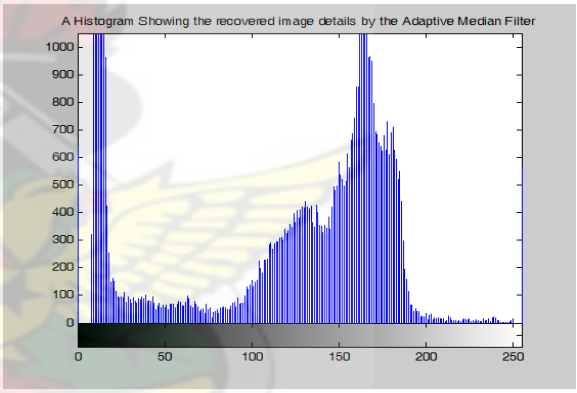
(xi) (xii)

Figure 4.6: The image corrupted by 30% impulse noise (xi) and its grey level histogram (xii).



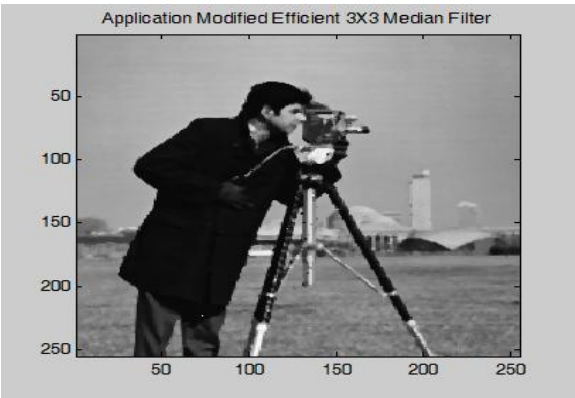
(xiii) (xiv)

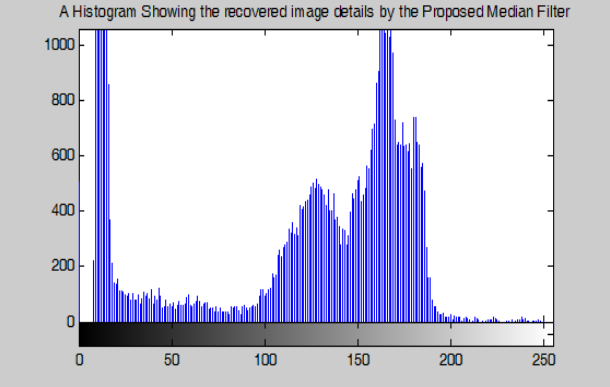
Figure 4.7: Application of the Standard Median filter to image corrupted by 30% impulse noise (xiii) and its grey level histogram (xiv).



(xv) (xvi)

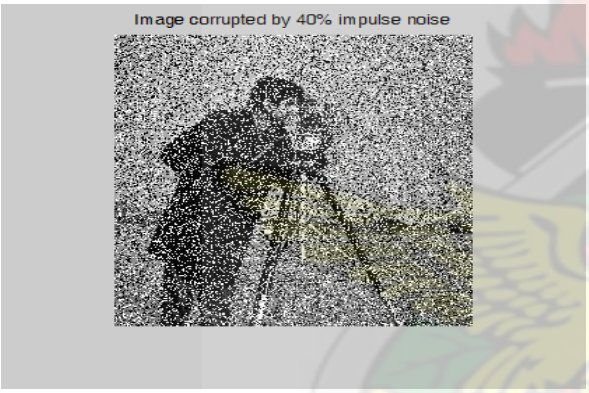
Figure 4.8: Application of the Adaptive Median filter to image corrupted by 20% impulse noise (xv) and its grey level histogram (xvi).

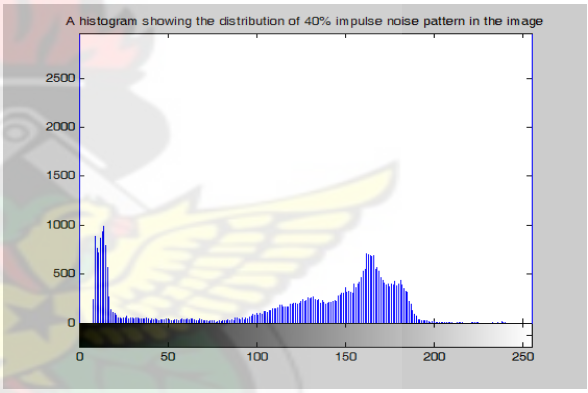




(xvii) (xviii)

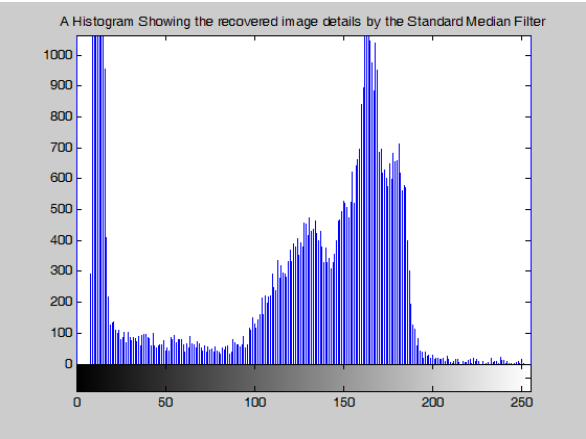
Figure 4.9: Application of the Improved Median filter to image corrupted by 30% impulse noise (xvii) and its grey level histogram (xviii).





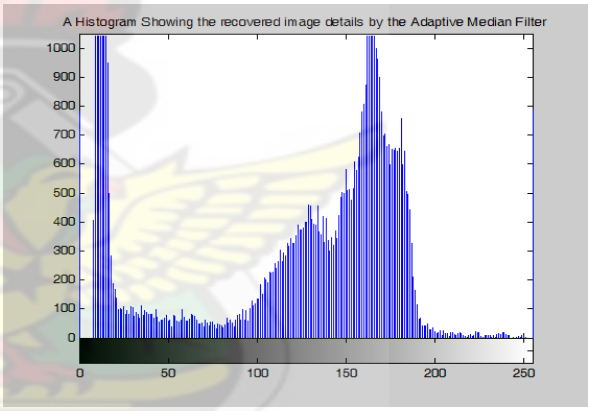
(xix) (xx)

Figure 4.10: The image corrupted by 40% impulse noise (xix) and its grey level histogram (xx).



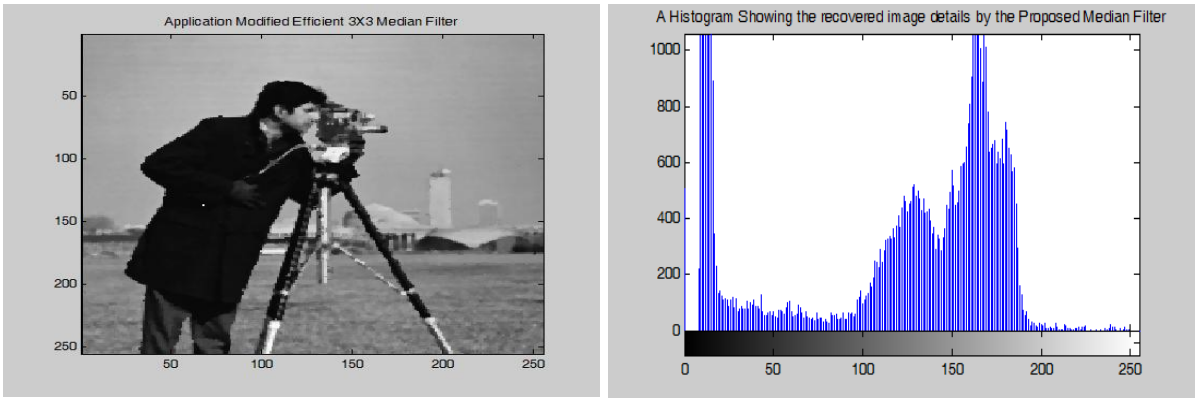
(xxi) (xxii)

Figure 4.11: Application of the Standard Median filter to image corrupted by 40% impulse noise (xxi) and its grey level histogram (xxii).



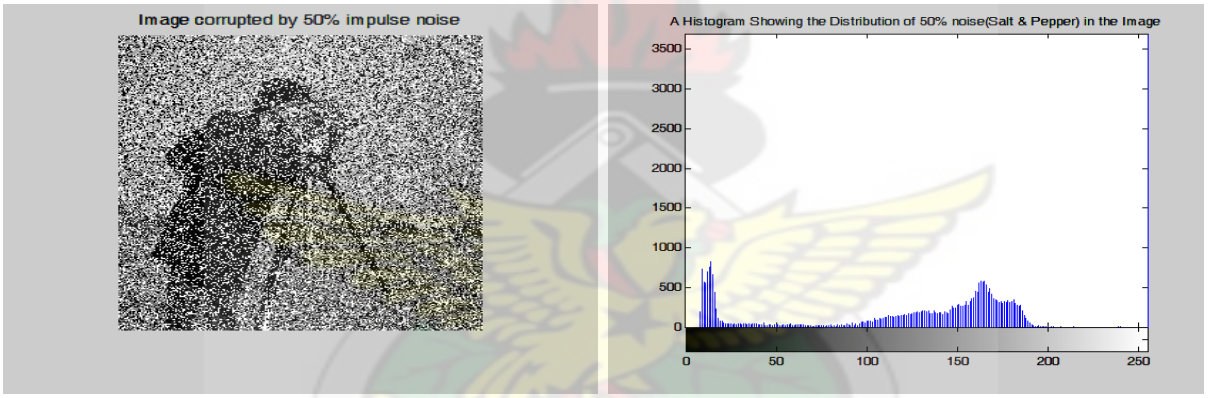
(xxiii) (xxiv)

Figure 4.12: Application of the Adaptive Median filter to image corrupted by 40% impulse noise (xxiii) and its grey level histogram (xxiv).



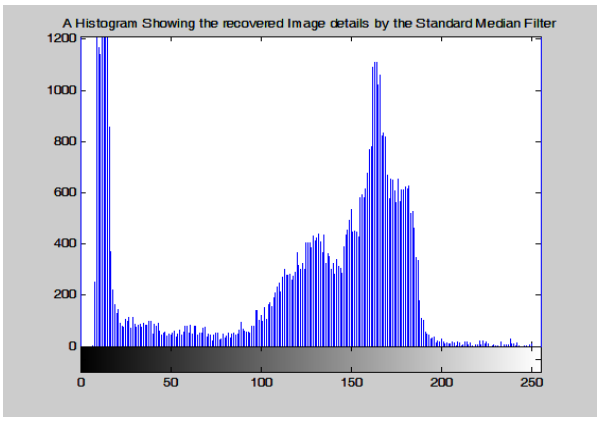
(xxv) (xxvi)

Figure 4.13: Application of the Improved Median filter to image corrupted by 40% impulse noise (xxv) and its grey level histogram (xxvi).



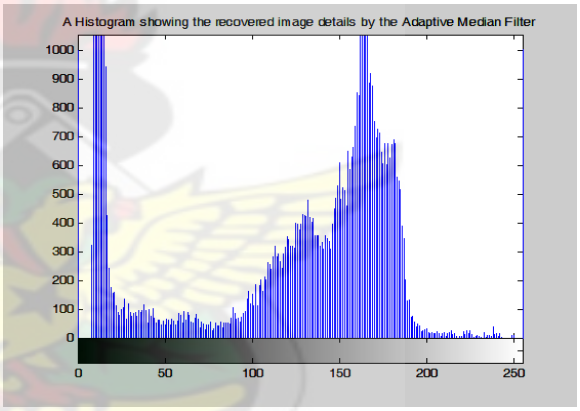
(xxvii) (xxviii)

Figure 4.14: The image corrupted by 50% impulse noise (xxvii) and its grey level histogram (xxviii).



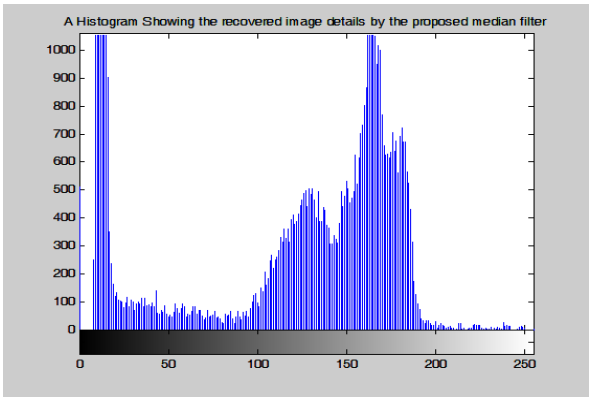
(xxix) (xxx)

Figure 4.15: Application of the Standard Median filter to image corrupted by 50% impulse noise (xxix) and its grey level histogram (xxx).



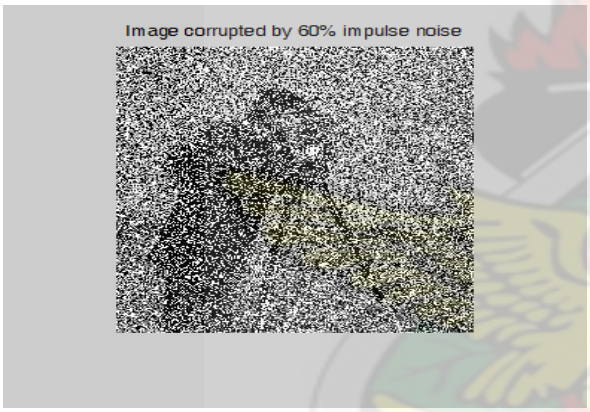
(xxxi) (xxxii)

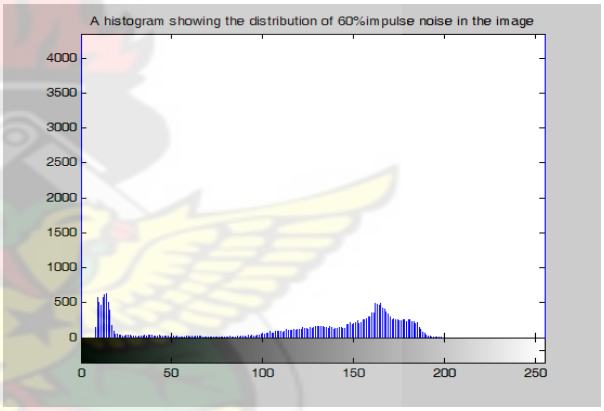
Figure 4.16: Application of the Adaptive Median filter to image corrupted by 50% impulse noise (xxxi) and its grey level histogram (xxxii).



(xxxiii) (xxxiv)

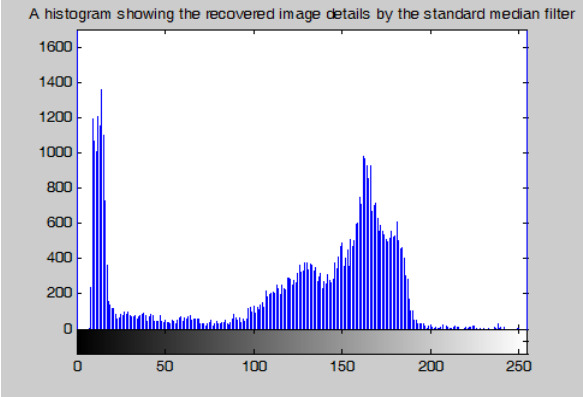
Figure 4.17 Application of the Improved Median filter to image corrupted by 50% impulse noise (xxxiii) and its grey level histogram (xxxiv).





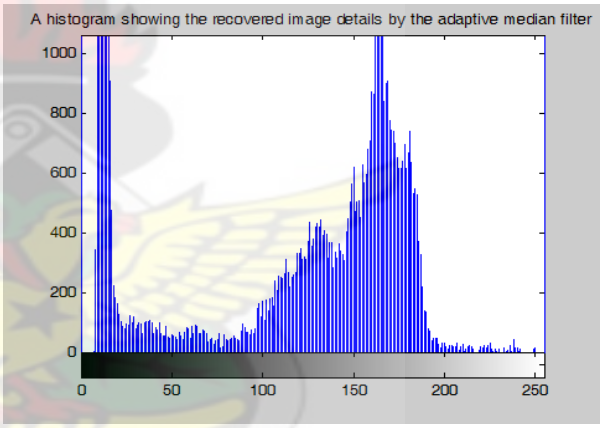
(xxxv) (xxxvi)

Figure 4.18: The image corrupted by 60% impulse noise (xxxv) and its grey level histogram (xxxvi).



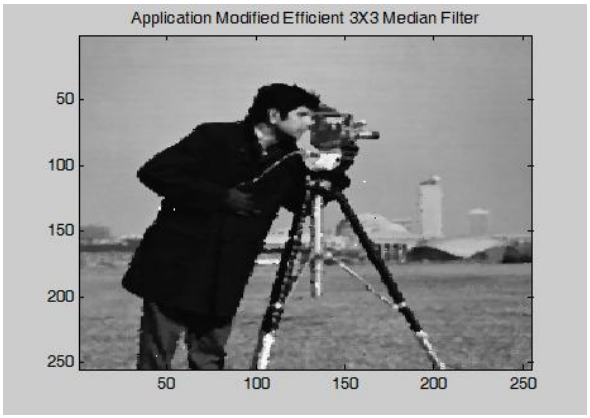
(xxxvii) (xxxviii)

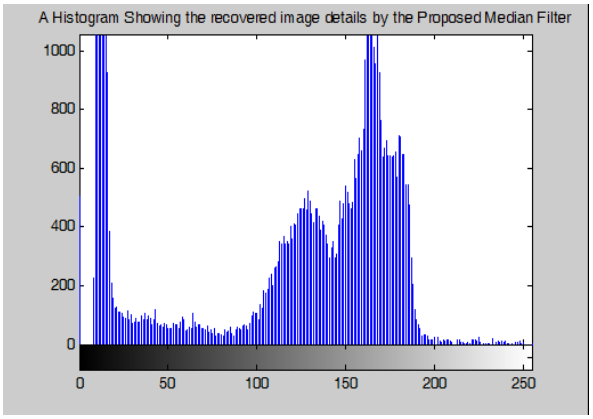
Figure 4.19: Application of the Standard Median filter to image corrupted by 60% impulse noise (xxxvii) and its grey level histogram (xxxviii).



(xxxix) (xxxx)

Figure 4.20: Application of the Adaptive Median filter to image corrupted by 60% impulse noise (xxxix) and its grey level histogram (xxxx).

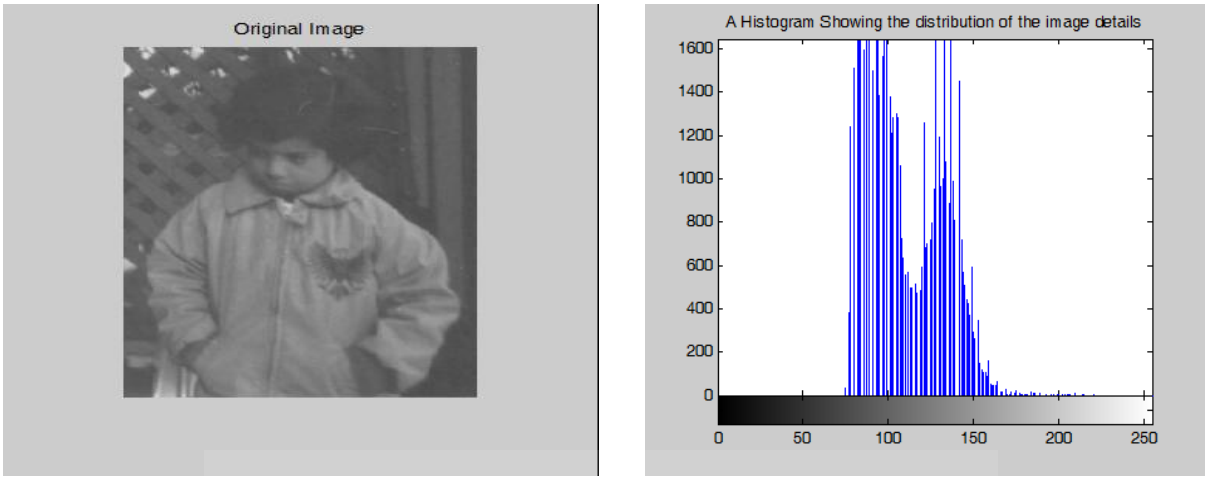




(xxxxi) (xxxxii)

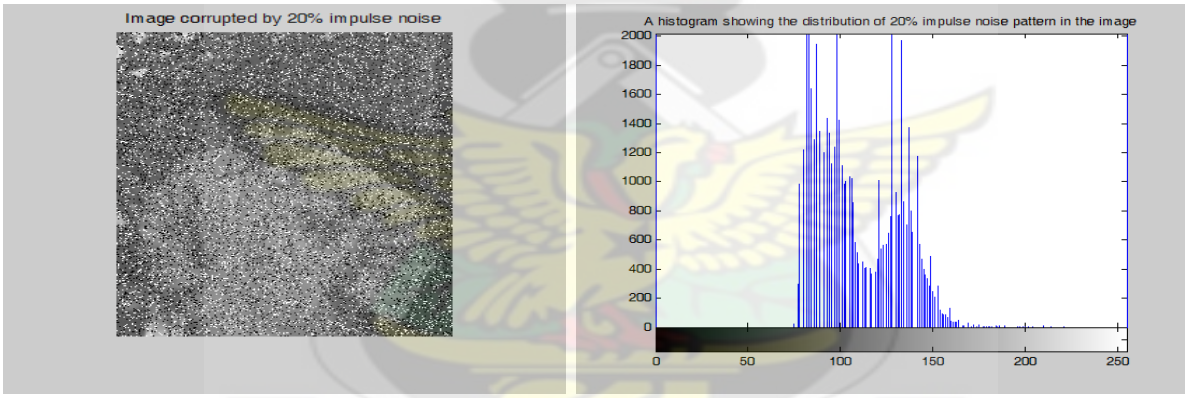
Figure 4.21: Application of the Improved Median filter to image corrupted by 60% impulse noise (xxxxi) and its grey level histogram (xxxxii).

The original image of the cameraman and it subsequent images corrupted by impulse noise of different percentages ranging from 20% to 60% and the application of the standard median filter, adaptive median filter and the improved median filter to the various corrupted images with impulse noise(salt and pepper). The results of the improved median filter are better as compared to the results of the standard median filter and the adaptive median filter.



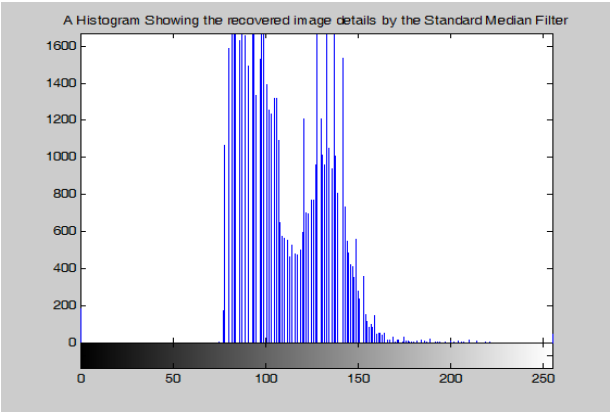
(i) (ii)

Figure 4.22: Original image of Pout (i) and its grey level histogram (ii).



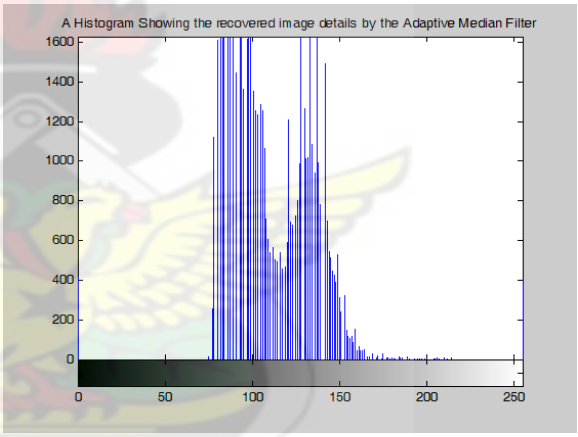
(iii) (iv)

Figure 4.23: The image corrupted by 20% impulse noise (iii) and its grey level histogram (iv).



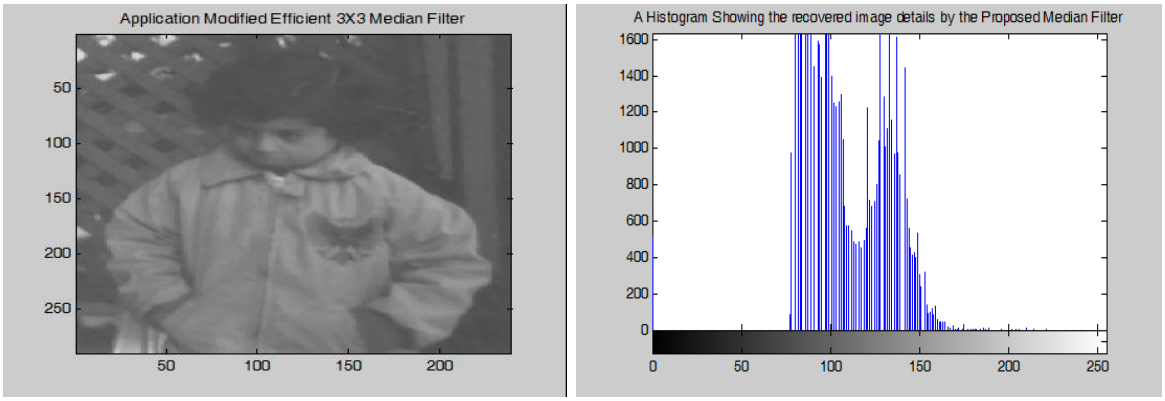
(v) (vi)

Figure 4.24: Application of the Standard Median filter to image corrupted by 20% impulse noise (v) and its grey level histogram (vi).



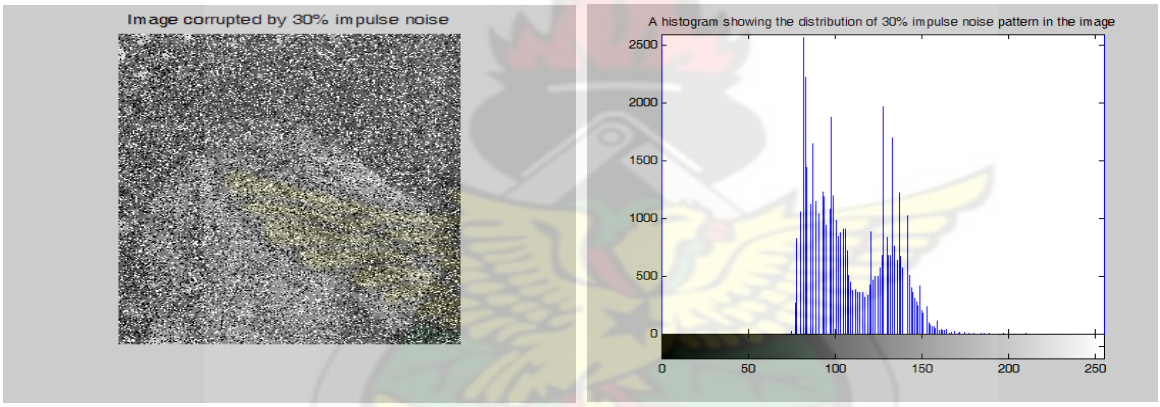
(vii) (viii)

Figure 4.25: Application of the Adaptive Median filter to image corrupted by 20% impulse noise (vii) and its grey level histogram (viii).



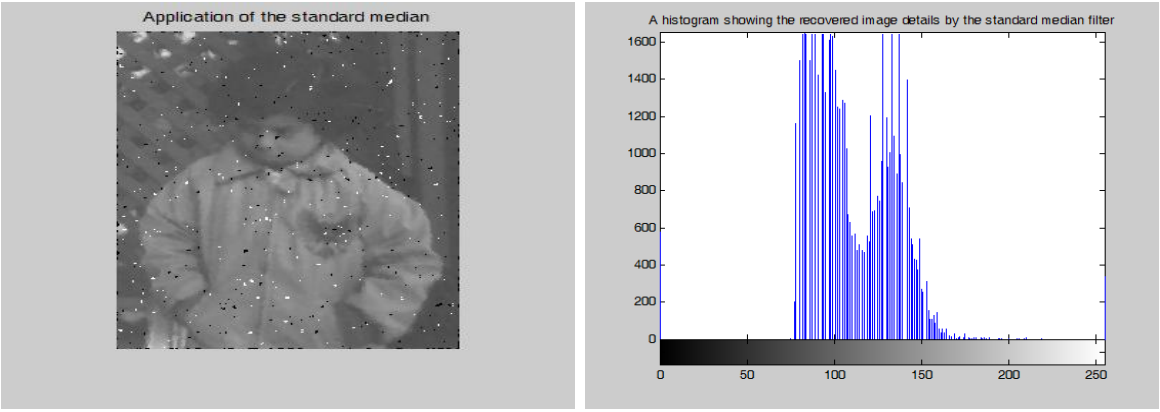
(ix) (x)

Figure 4.26: Application of the Adaptive Median filter to image corrupted by 20% impulse noise (ix) and its grey level histogram (x).



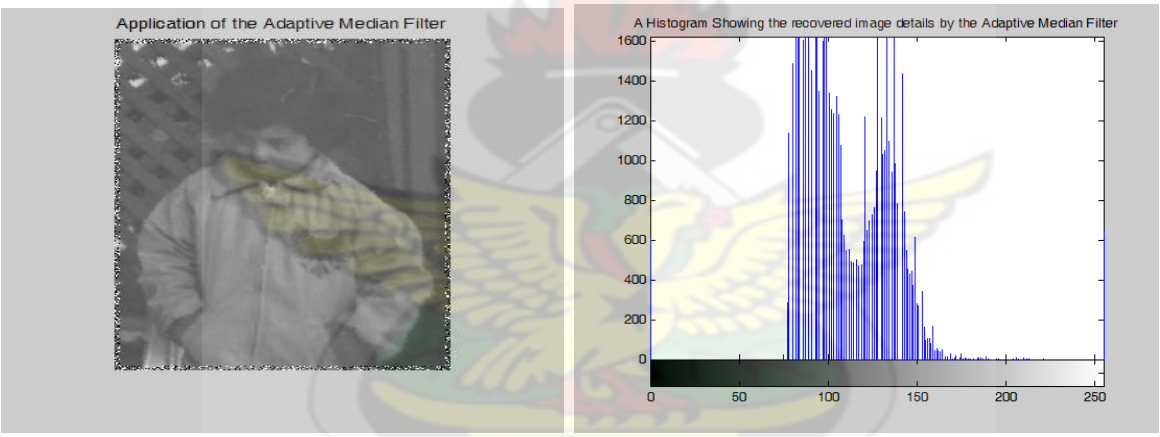
(xi) (xii)

Figure 4.27: The image corrupted by 30% impulse noise (xi) and its grey level histogram (xii).



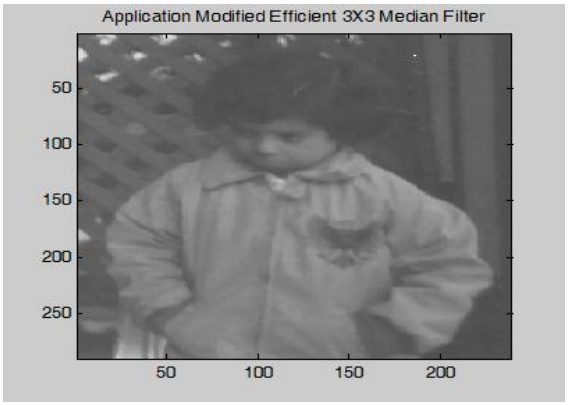
(xiii) (xiv)

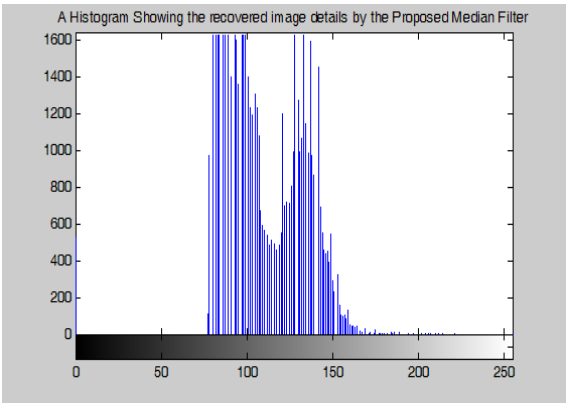
Figure 4.28: Application of the Standard Median filter to image corrupted by 30% impulse noise (xiii) and its grey level histogram (xiv).



(xv) (xvi)

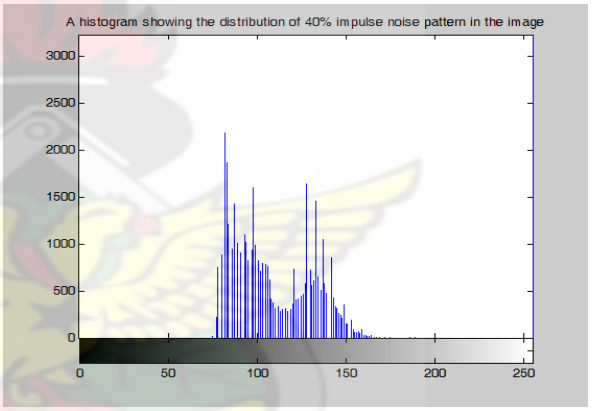
Figure 4.29: Application of the Adaptive Median filter to image corrupted by 30% impulse noise (xv) and its grey level histogram (xvi).





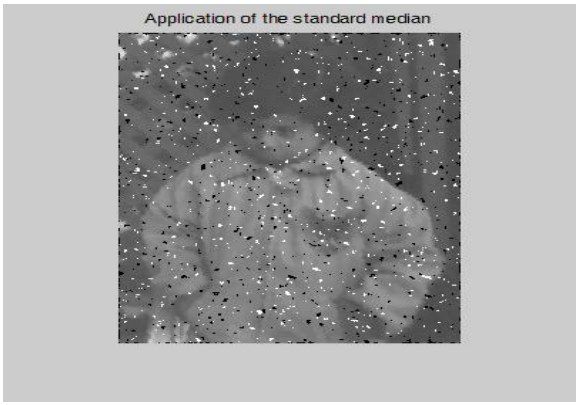
(xvii) (xviii)

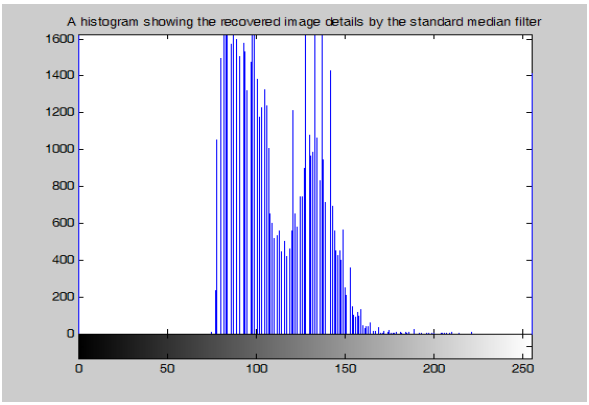
Figure 4.30: Application of the Standard Median filter to image corrupted by 30% impulse noise (xvii) and its grey level histogram (xviii).



(xix) (xx)

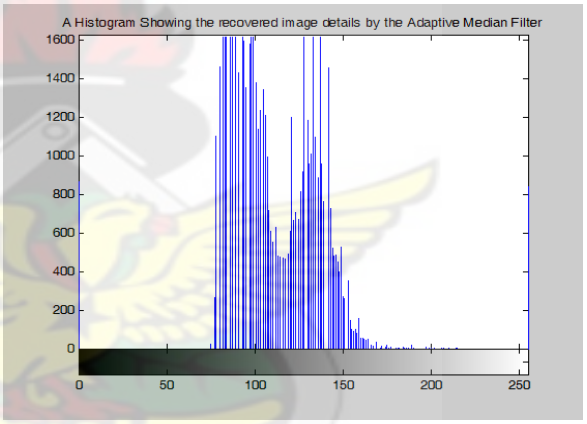
Figure 4.31: The image corrupted by 40% impulse noise (xix) and its grey level histogram (xx).





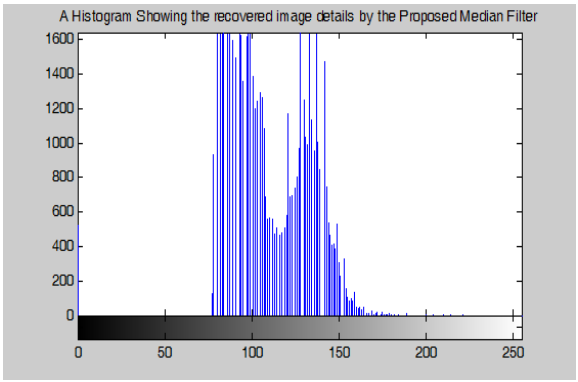
(xxi) (xxii)

Figure 4.32: Application of the Standard Median filter to image corrupted by 40% impulse noise (xxi) and its grey level histogram (xxii).



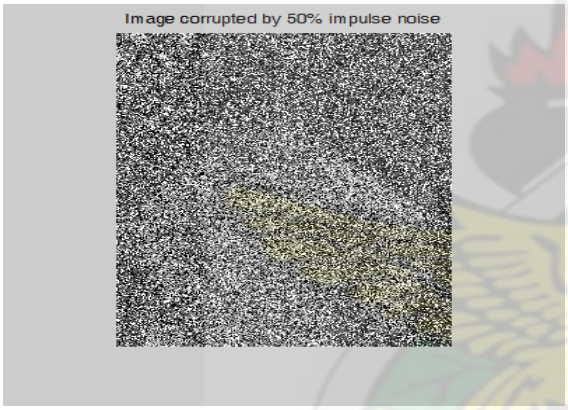
(xxiii) (xxiv)

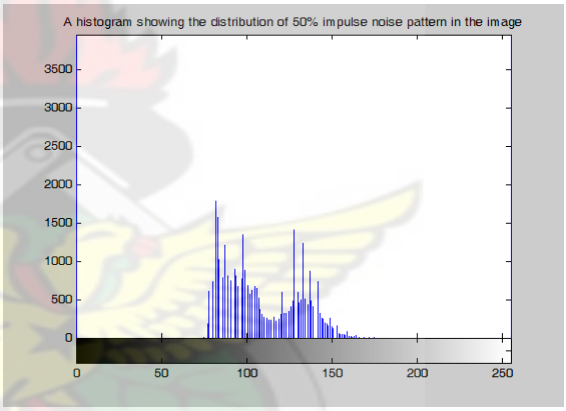
Figure 4.33: Application of the Standard Median filter to image corrupted by 40% impulse noise (xxii) and its grey level histogram (xxiv).



(xxv) (xxvi)

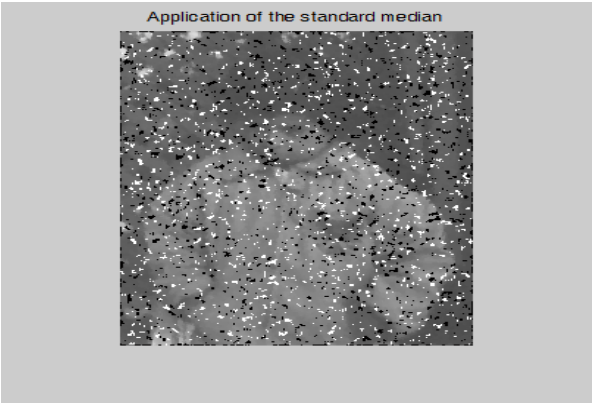
Figure 4.34: Application of the Improved Median filter to image corrupted by 40% impulse noise (xxv) and its grey level histogram (xxvi).

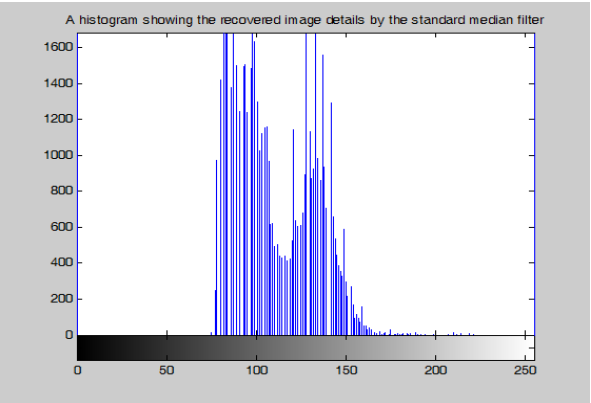




(xxvii) (xxviii)

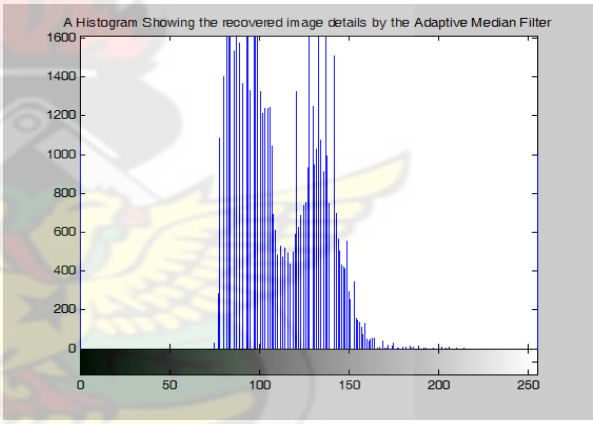
Figure 4.35: The image corrupted by 50% impulse noise (xxvii) and its grey level histogram (xxviii).





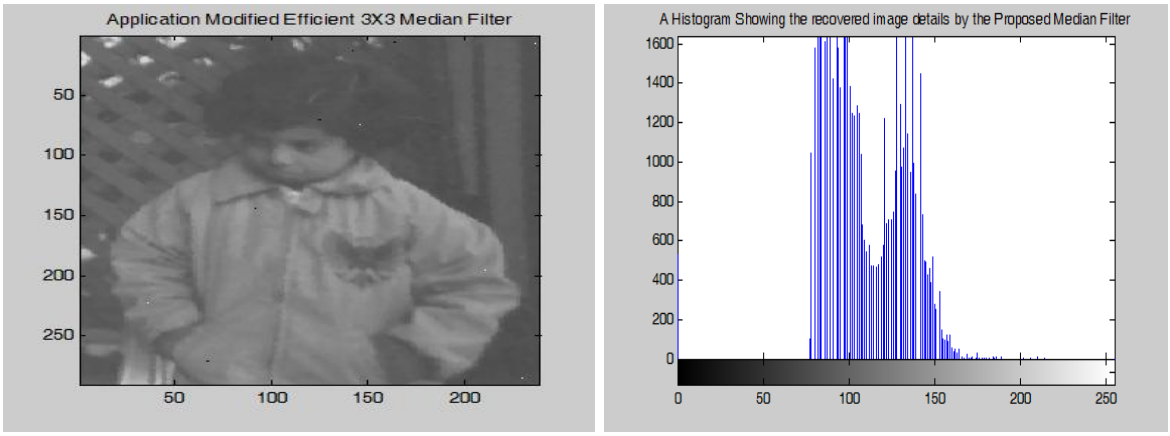
(xxix) (xxx)

Figure 4.36: Application of the Standard Median filter to image corrupted by 50% impulse noise (xxix) and its grey level histogram (xxx).



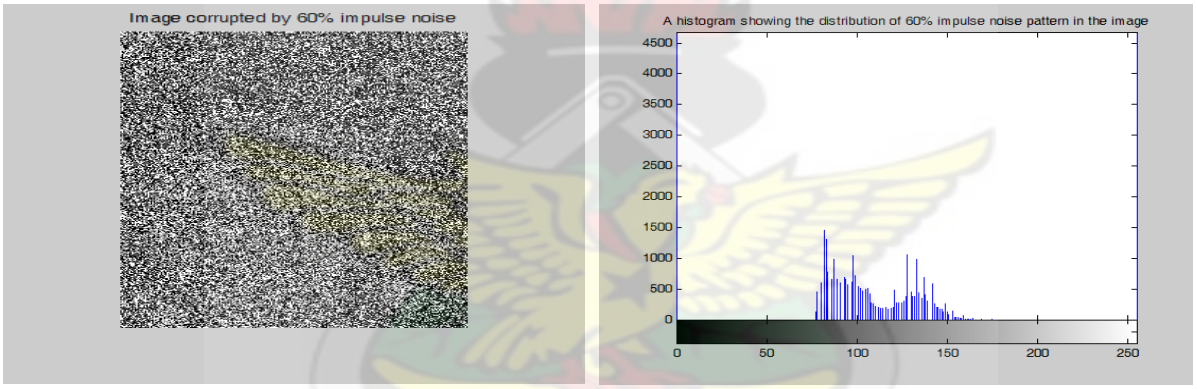
(xxxi) (xxxii)

Figure 4.37: Application of the Standard Median filter to image corrupted by 50% impulse noise (xxxi) and its grey level histogram (xxxii).



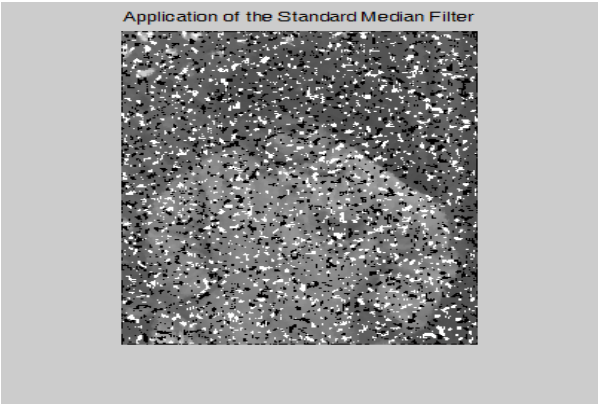
(xxxiii) (xxxiv)

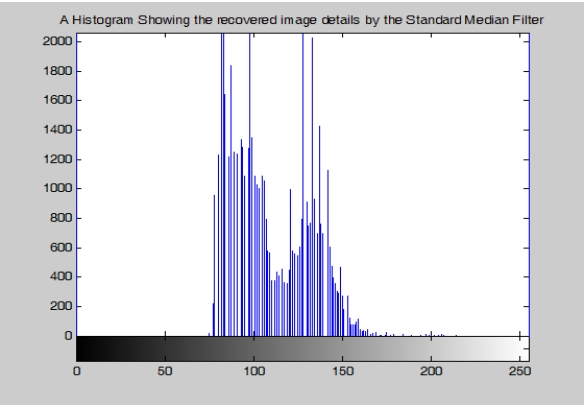
Figure 4.38: Application of the Improved Median filter to image corrupted by 50% impulse noise (xxxiii) and its grey level histogram (xxxiv).



(xxxv) (xxxvi)

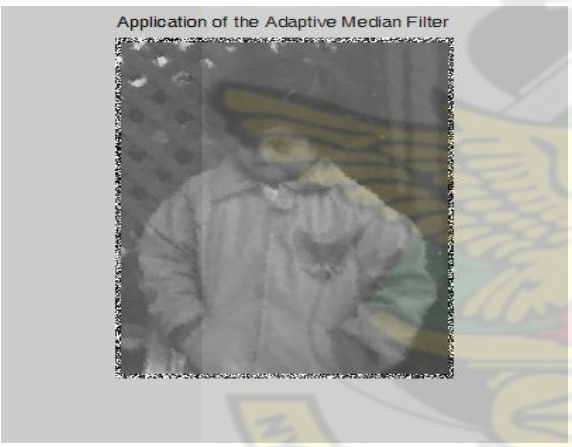
Figure 4.39: The image corrupted by 60% impulse noise (xxxv) and its grey level histogram (xxxvi).

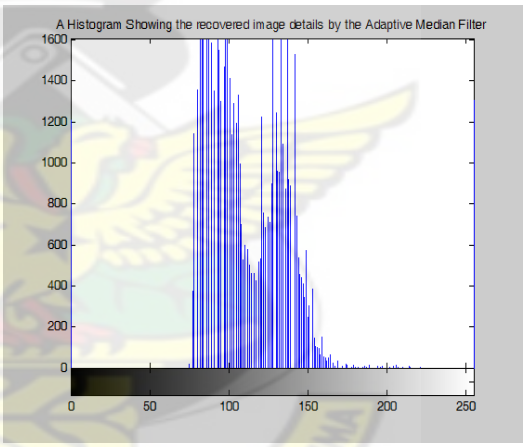




(xxxvii) (xxxviii)

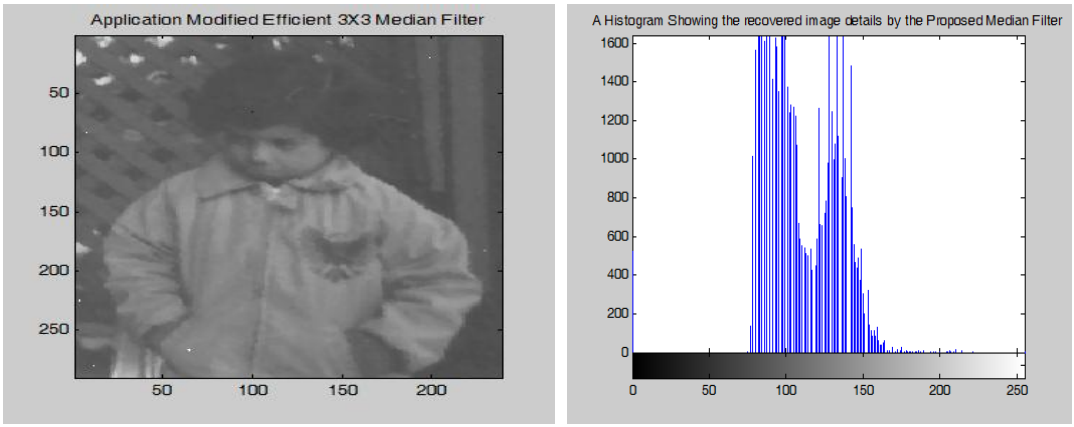
Figure 4.40: Application of the Standard Median filter to image corrupted by 60% impulse noise (xxxvii) and its grey level histogram (xxxviii).





(xxxix) (xxxx)

Figure 4.41: Application of the Adaptive Median filter to image corrupted by 60% impulse noise (xxxix) and its grey level histogram (xxxx).



(xxxxi) (xxxxii)

Figure 4.42: Application of the Standard Median filter to image corrupted by 60% impulse noise (xxxxi) and its grey level histogram (xxxxii).

The original image of the Pout and it subsequent images corrupted by impulse noise of different percentages ranging from 20% to 60% and the application of the standard median filter, adaptive median filter and the improved median filter to the various corrupted images with impulse noise(salt and pepper). The results of the improved median filter are better as compared to the results of the standard median filter and the adaptive median filter.

4.3 EVALUATION OF THE IMPROVED MEDIAN FILTERING ALGORITHM

Filtering a noisy image or an image polluted with impulse noise is significantly different from compressing an image. General purpose filtering techniques can be used to filter out various types of noise but they perform poorly in the presence of impulse noise, to obtain good result the standard median filter is used in the presence of impulse noise. The adaptive median filter performs better than the standard median filter in the filtering of impulse noise (salt and pepper noise) image. The improved median filter achieves excellent result as compared to the two above mentioned filters. This is because the impulse noise has certain statistical

properties which can be best resolved by the improved median filter designed purposely for the removal of impulse noise and distortion. The improved median filter is preferred because apart from the filtering of noise and distortions it also preserves fine details in the image.

A number of quantitative measuring parameters can be used to evaluate the performance of the various types of filters in terms of restoring damage signals quality and quantity after filtering out impulse noise (salt and pepper). The following measuring parameters are used in this thesis;

- Mean Square Error (MSE)
- Peak Signal-to-Noise Ratio

Mean square error of two separate images is computed for various types of filters. The MSE is the cumulative squared error between the noisy image and the filtered image. The mathematical formula for the MSE is;

$$MSE = \frac{1}{MN}\sum_{j=1}^{M}\sum_{i=1}^{N}(G(i,j)-F(i,j))^2 \quad \text{.....(4)}$$

Where, M and N are the total number of pixels in the column and row of the image respectively. G denotes the noisy image and F denotes the filtered image. The quantitative result in Table 4.1and Table 4.2 and the graph in Figure 4.41 and Figure 4.42 show that the proposed effective median filter performed best in terms of impulse noise removal in a corrupted image than the other filters.

The peak signal-to-noise ratio is one of the best known techniques for assessing the amount of noise that an image is polluted with and, for that matter, the amount of noise left in a filtered image. The peak signal-to-noise criterion is adopted to measure the performance of various digital filtering techniques quantitatively. It is the measure of the peak error.

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right) \quad \text{.....(5)}$$

A lower value for MSE means lesser error, and as seen from the inverse relation between the MSE and PSNR, this translates to a high value of PSNR. Logically, a higher value of PSNR is good because it means that the ratio of Signal to Noise is higher. Here, the 'signal' is the original image, and the 'noise' is the error in reconstruction or restoration. So, if you find a filtering or compression scheme having a lower MSE and a high PSNR, you can recognize that it is a better one.

Table 4.1: Peak Signal-to-Noise Ratio for Cameraman

Filter Methods	20% Impulse noise	30% Impulse Noise	40% Impulse noise	50% Impulse Noise	60% Impulse Noise
Minimum Filter	30.3778dB	29.6440dB	29.1562dB	28.8751dB	28.7417dB
Maximum Filter	25.9170dB	26.0235dB	26.1702dB	30.0430dB	26.4269dB
Standard Median Filter	32.1349dB	31.0147dB	30.2251dB	29.6842dB	29.3812dB
Adaptive Median Filter	33.9901dB	32.3710dB	31.2870dB	30.3192dB	29.4614dB
Proposed Method	38.0638dB	35.0285dB	33.0720dB	31.8273dB	30.6626dB

Table 4.2: Peak signal-to-Noise Ratio for Pout

Filter Methods	20% Impulse Noise	30% Impulse Noise	40% Impulse noise	50% Impulse Noise	60% Impulse Noise
Minimum Filter	31.1923dB	30.0774dB	29.3840dB	28.9838dB	28.7919dB
Maximum Filter	27.2161dB	27.1687dB	27.1501dB	30.0619dB	27.1298dB
Standard Median Filter	33.7248dB	32.0699dB	30.9098dB	30.1609dB	29.6767dB
Adaptive Median Filter	34.3248dB	32.5215dB	31.2834dB	30.3181dB	29.5665dB
Proposed Method	39.8657dB	35.4733dB	33.3535dB	31.9648dB	30.8321dB

4.4 RESULTS AND DISCUSSION

The non-linear ordered satistic filtering algorithms are tested on two standard two-dimensional images with normal pixels values from 0 to 255 after adding noise to levels ranging from 20% to 60%. The improved median filter is tested on the MATLAB platform with a 3×3 fixed window throughout. The picture quality obatined from the application of the improved median filtering algorithm on the various noisy image with different noise levels show that the proposed filter is superior in terms of filtering impulse noise from a two dimensional image.

From the figures below, it can be concluded that, when the impulse noise in the image is increased the improved median filtering algorithm achieves more enhanced images than the other nonlinear ordered statistic filter. Like the other median filters, the performance of the improved median filter filter detriotrates as the noise factor increases.

The peak signal to noise ratio of the improved median filtering algorithm, the adaptive median filtering algorithm, the standard median filtering algorithm, the maiximum filtering algorithm and the minimum filtering algorithm are shown in table 4.1 and 4.2. The results obtained from the peak signal to noise ratio computations show that at low to moderate noise levels, the performance of the improved median filtering algorithm stands out. This is clear from the graphical representations in Fig. 4.43 and Fig. 4.44.

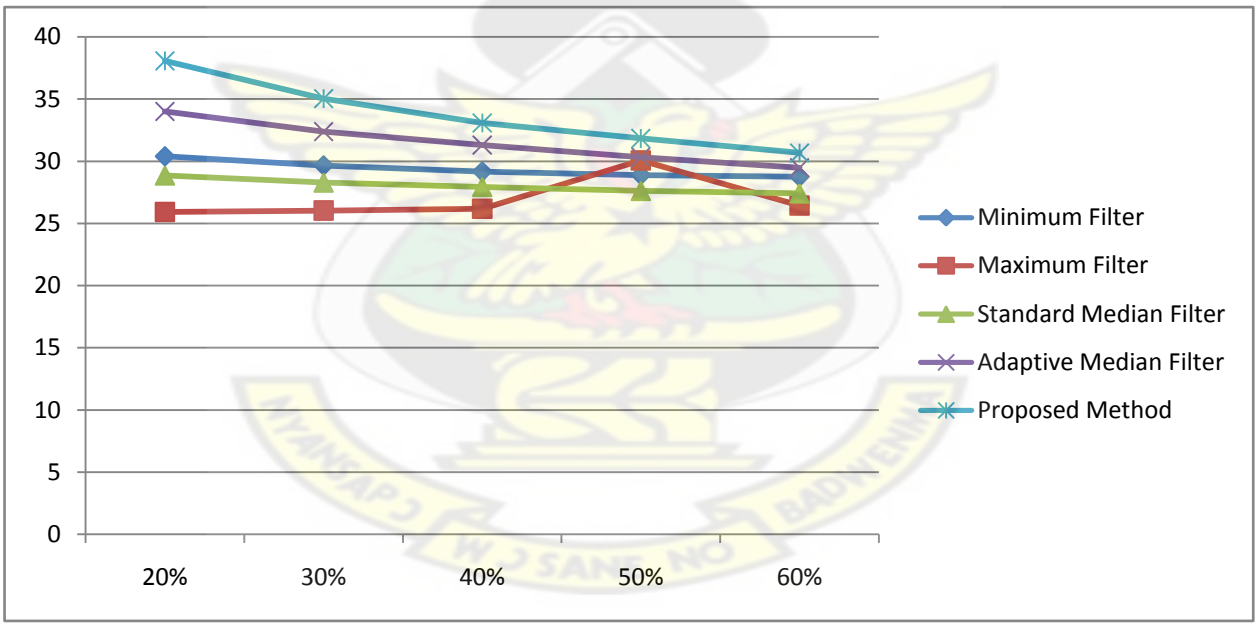


Figure 4.43: A Graphical representation of the result of the peak signal-to-noise ratio for Cameraman image applying the several filtering techniques

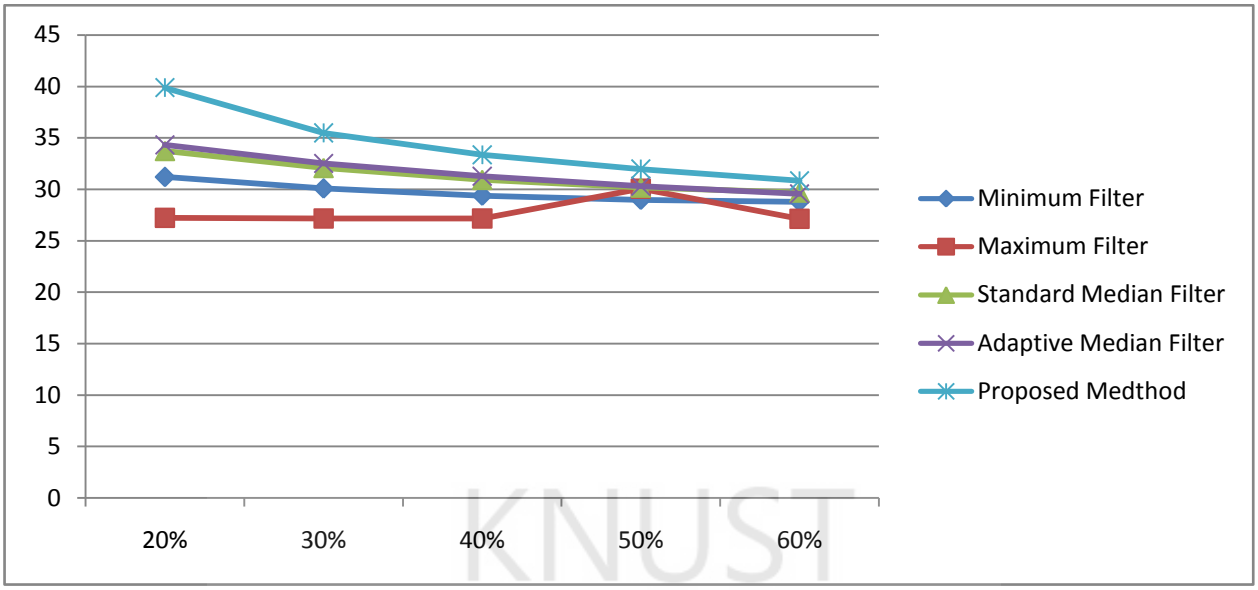


Figure 4.44: A Graphical representation of the result of the peak signal-to-noise ratio for Pout image applying the several filtering techniques

CHAPTER FIVE

5.0 CONCLUSION AND RECOMMENDATIONS

5.1 CONCLUSION

Digital images are often corrupted by Impulse noise due to errors generated in noisy sensor, errors that occur in the process of converting signals from analog-to-digital and also errors that are generated in the communication channels. In order to remove impulse noise and enhance the affected image quality, median filter is applied.

The median filter is an active and interesting area of research, since it is the best nonlinear ordered statistics filter in the sense that it achieves the best results or output when used in the filtering of impulse noise than the rest of its kind.

Several median filtering algorithms have been proposed, some of the median filtering algorithms are the standard median filter, the switching median filter, the weighted median filter and the adaptive median filter.

These filtering methods have been implemented and tested on several standard images in literature and have been proven to achieve good results, when they are applied to images polluted with impulse noise. However, they still have some weakness/shortcomings.

The proposed median filtering algorithm is a simple decision base median filter. Conditions are defined and a series of tests are carried out before a decision is reached.

The sliding window technology is employed in selecting a window in the proposed algorithm. This technology is adapted from the standard median filtering algorithm. The algorithm employs the greedy algorithm technique in decision making to select the median and divide and conquer mechanism in the dividing of the algorithm into several parts for efficient and effective implementation.

The algorithm performs a scan through the window pixels values and the distortions in the form of thickening and thinning are detected with a defined condition and replaced with the effective median.

The proposed median filtering algorithm developed is coded in MATLAB 7.40(R2007a), and applied to some standard images with different percentages of impulse noise introduced in the image ranging from 20% to 60% impulse noise pollution.

The results of the proposed median filtering algorithm implemented are compared to those of the standard and adaptive median filtering method. The improved median filtering algorithm when applied to corrupted images produces better picture quality after the filtering process than the standard and adaptive median filter.

The improved median filter solves the problem of the adaptive median filter because it detects and separates the noisy pixels from the good pixels. A series of test is conducted on all the pixels in the window and the good pixels are left intact while the noisy ones are replaced with the computed effective median.

The improved median filter is good in detecting edge pixels and preserving them. It does not weaken signal since it only replaces noisy pixels in the window with the effective median.

The intensity of the noise in the image does not influence the improved median filtering algorithm to select a noisy pixel as the effective median. And better still the noisy pixel detected is not left intact or replace with another noisy pixels.

The improved median filtering algorithm uses a 3 × 3 fixed window, which reduces the complexity of the window since only nine pixels are involved in the computation of the effective median. The window of the adaptive median filtering algorithm grows from 3 × 3 to 5 × 5 to 7 × 7 to 9 × 9. When the window is selected and the pixels are sorted, if the value selected as the median value is equal to the minimum value or the maximum value the

median is an impulse and hence the input window is extended to a 5 × 5 window. This process is complex because at every increase in window size the pixels must be re-sorted.

The quantitative results obtain from the peak signal to noise ratios and the mean square errors of the various filtering methods prove that indeed the improved median filtering technique is more effective than the standard median filter and the adaptive median filter in filtering impulse noise. Also the improved median filtering algorithm is more effective and efficient in preserving fine details and edge details of the filtered image.

5.2 RECOMMENDATIONS

The improved median filter outperforms several median filters but also has some shortcomings, one of the drawbacks of the proposed method is that when the window is selected and all the pixels in the window happen to be impulses. The mechanism used to find a value to replace the corrupted pixels sometimes does not generate a more robust value. For that matter further research should be conducted on developing a more robust value when the selected window's pixel values are all corrupted.

It is recommended that the further improved method be implemented in Very Large Scale Integrated circuit for the speedup of the process of image filtering.

REFERENCES

1. R.C.Gonzalez and R.E. Wood, Digital Image Processing, Prentice-Hall, India, Second Edition, 2007.

2. Goutsias, J,Vincent, L., and Bloomberg, D. S. (eds.), Mathematical Morphology and Its Applications to Image and Signal Processing, Kluwer Academic Publishers, Boston, MA., 2000.

3. Ritter,G.X. and Wilson, J.N., Handbook of Computer Vision Algorithms in Image Algebra, CRC Press, Boca Raton, FL., 2001.

4. Russ, John C., The Image Processing Handbook (Third Edition), CRC Press, 1999.

5. Geoff Dougherty, Digital Image Processing for Medical Applications, Cambridge University Press, 2007.

6. R.C.Gonzalez, R.E. Wood and Steven L. Edwin, Digital Image Processing Using MATLAB, Prentice-Hall, India, Second Edition, 2008.

7. K. Konstantinides and V. Bhaskaram, "Monolithic architectures for image processing and compression", IEEE Computer Graphicsand Application, pp.75-86, Nov. 1996.

8. Willian K. Pratt, Digital Image Processing; Third Edition, Wiley-intersecience Publication, New York, U.S.A., Copyright 2001.

9. P. Pirch and H-J. Stolberg, "VLSI Implementations on Image and Video Multimedia Processing Systems", IEEE Transaction on Circuits and Systems for Video Technology, Vol. 8, No. 7, Nov., 1998.

10. Wei Wang, M.N.S. Swamy and M.O. Ahmad, "RNS Application for Digital Image Processing", 4th IEEE international workshop on System-on-chip for Real-time Application, pp. 77-80, July 2004.

11. A. C. Bovik, T. S. Huang, and D. C. Munson, "The effect of median filtering on edge estimation and detection," IEEE Trans. Patt. Anal. Mach. Intell., vol. 9, pp. 181-194, 1987.

12. R. V. Hogg, “Adaptive robust procedures: A partial review and some suggestions for future applications and theory,” J. Amer. Statist..Assoc., vol. 69, no. 348, pp. 909-923, 1974.

13. Haykin, Simon, and Barry Van Veen, Signals and Systems. 2nd ed. Hoboken, NJ: John Wiley and Sons, Inc., 2003.

14. McClellan, James H., Ronald W. Schafer, and Mark A. Yoder, Signal Processing First. Upper Saddle River, NJ: Pearson Education, Inc., 2003.

15. A. V. Oppenheim and R. W. Schafer, Discrete-Time Signal Processing. Englewood Cliffs, New Jersey: Prentice Hall, 1989.

16. Wakerly John F, Digital Design: Principles and Practices, 4th Edition, Pearson Prentice-Hall, 2006

17. F. Haugen: Dynamic Systems — Modeling, Analysis and Simulation, Tapir Academic Press, Norway, 2004.

18. C. D. McGillem and G. R. Cooper, Continuous and Discrete Signals and Systems Analysis, 2nd ed., New York: Holt, Rinehart & Winston, 1984.

19. Richard G. Lyons: Understanding Digital Signal Processing, Prentice Hall, ISBN 0-13-108989-7

20. Bernard Mulgrew, Peter Grant, John Thompson: Digital Signal Processing - Concepts and Applications, Palgrave Macmillan, ISBN 0-333-96356-3.

21. Finn Huagen, Discrete-Time Signal and Systems, Tapir Academic Press, Norway, 2005.

22. Steven W. Smith: Digital Signal Processing - A Practical Guide for Engineers and Scientists, Newnes, ISBN 0-7506-7444-X.

23. H. G. Longbotham and N. Barsalou, “The LMS, an adaptive optimal order statistic filter,” in Nonlinear Image Processing, E. J. Delp, Editor, Proc. SPIE, vol. 1247, pp. 78-88, 1990.

24. Z. M. Ryu, The Effect of Nonlinear Filtering on the Resolution of Calibrated Thermal Images. Ph.D. Dissertation, University of Texas at Austin, 1986.

25. Ryu, Zee Man. "The effect of nonlinear filtering on the resolution of calibrated thermographic images." Ph.D. dissertation, University of Texas at Austin (1986).

26. C. G. Boncelet, Jr., R. Hardie, R. Hakami, and G. R. Arce, “LUM filters for smoothing and sharpening,” in Nonlinear Image Processing II, E. R. Dougherty, G. R. Arce, and C. G. Boncelet, Jr., Editors, Proc. SPIE, vol. 1451, pp. 70-74, 1991.

27. Hussain, Z.: “Digital Image Processing – Practical Applications of Parallel Processing Techniques,” Ellis Horwood, West Sussex, UK, 1991.

28. J.-S. Lee, “Digital image enhancement and noise filtering by use of local statistics,” IEEE Trans. Patt. Anal. Mach. Intell., vol. 2, pp. 165-168,1980.

29. M. A. Schulze and J. A. Pearce, “Some properties of the two-dimensional pseudomedian filter,” in Nonlinear Image Processing II, E. R. Dougherty, G. R. Arce, and C. G. Boncelet, Jr., Editors, Proc. SPIE, vol. 1451, pp. 48- 57, 1991.

30. Bezerra Candeias, A. L., J. C. Mura, et al., Interferogram phase noise reduction using morphological and modified median filters, 1995.

31. Nodes, Thomas A. and Neal C. Gallagher, Jr. "Median filters: some modifications and their properties." IEEE Trans Acoust Speech Signal Process, v. ASSP-30 n. 10 (1982), pp. 739-746.

32. Fisher B., Perkins S., Walker A., and Wolfart E.,”Hypermedia Image Processing Reference”, University of Edinburgh, April 2005, UK.

33. Hwang.H and Haddad.R.A, “Adaptive median filters: new algorithms and results,” IEEE Trans. On Image Processing, vol.4, no.4, pp.499-502, 1995.

34. Zdenek Vasicek, Lukas Sekanina, Novel Hardware Implementation of Adaptive Median Filters, 978-1-4244-2277-7/08/ ©2008 IEEE.

35. Bernard Widrow and Samuel D. Steavns, “Adaptive Signal Processing”, Pearson Edition, 2000.

36. D. Dhanasekaran and K. B. Bagan, “High Speed Pipelined Architecture for Adaptive Median Filter”, European Journal of Scientific Research, Vol. 29, No. 4, pp. 454-460, 2009.

37. Justusson, B. I. "Median filtering: statistical properties." in T. S. Huang, ed., Two-Dimensional Digital Signal Processing II, Berlin: Springer-Verlag, 1981, pp. 161-196.

38. A. C. Bovik, T. S. Huang, and D. C. Munson, “The effect of median filtering on edge estimation and detection,” IEEE Trans. Patt. Anal. Mach. Intell., vol. 9, pp. 181-194, 1987.

39. Yueli Hu and Hujie Ji, “Research on Image Median Filtering Algorithm and It’s FPGA Implementation”, IEEE Computer Society: Global Congress on Intelligent Systems, pp. 226-230, 2009.

40. Ch. Ravi Kumar and S. K. Srivathsa, Hardware Implementation For Design of Modified Adaptive Median Filter for Image Processing, International Journal of Computer Science and Network Security, Vol. 10, No. 3, 2010, pp. 93-97.

41. M. Juneja and P. S. Sandhu, “Design and Development of an Improved Median Filtering Method for Impulse-Detection”, International Journal of Computer and Electrical Engineering, Vol. 1, No. 5, pp. 627-630, Dec., 2009.

42. Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest, Introduction to Algorithms, MIT press, Cambridge - London, 1990.

43. Alfred V. Aho, John E. Hopcroft and Jeffrey D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974.

44. Brian D. Hahn and Daniel T. Valentine, Essential MATLAB for Engineers and Scientist, Third Edition, Elsevier ltd., Italy, 2007.

45. Andy H. Register, A Guide to MATLAB: Object-Oriented Programming, SciTech Publishing Inc., U.S.A., 2007.

APPENDICES

Program to filter the impulse noise using the improved median filtering technique

```
function J = improvedmed_filt(I)
% 2-D Improve Median Filter
J = Improvedmed_filt(I) performs an improved median filtering of the matrix
%   A in two dimensions. Each output pixel contains the improved median
%   value in the M-by-N neighborhood around the corresponding
%   pixel in the input image.
smax = 3;
K1=0;
K2=255;
% Initialize Output Image (J)
J = I;

% Calculate valid region limits for filter
[nrows ncols] = size(I);
ll = ceil(smax/2);
ul = floor(smax/2);

% Loop over the entire image ignoring
for rows = ll:nrows-ul
   for cols = ll:ncols-ul

      window_ind = -ul:ul;
      region = I(rows+window_ind,cols+window_ind);
```

```
        s=vsort(region(nrows,ncols));


    end
 end


Xmax=s(1);
Xmin=s(9);
if Xmax==K2 || Xmin==K1
for i = ll:nrows-ul
    for j = ll:ncols-ul
        Xmed=median(region(i,j)
    end
end


 function X[rows, cols]=remove_noise(min, max)
for rows = ll:nrows-ul
    for cols = ll:ncols-ul
        if X(rows, cols)= = min
            X(rows, cols)= REffect_med;
        elseif X(rows, cols)= = max
            X(rows, cols)= LEffect_med;
        else
            X(rows, cols)=X(rows, cols)
        end
    end
```

```
end

function y = mover_median(x,dim)
nInputs = nargin;

if isempty(x)
  if nInputs == 1

    % The output size for [] is a special case when DIM is not given.
    if isequal(x,[]), y = nan(1,class(x)); return; end

    % Determine first nonsingleton dimension
    dim = find(size(x)~=1,1);

  end

  s = size(x);
  if dim <= length(s)
     s(dim) = 1;              % Set size to 1 along dimension
  end
  y = nan(s,class(x));

elseif nInputs == 1 && isvector(x)
  % If input is a vector, calculate single value of output.
  x = sort(x);
```

```
  nCompare = numel(x);
  half = floor(nCompare/2);
  y = x(half+1);
  if 2*half == nCompare        % Average if even number of elements
    y = meanof(x(half),y);
  end
  if isnan(x(nCompare))        % Check last index for NaN
    y = nan(class(x));
  end
else
  if nInputs == 1              % Determine first nonsingleton dimension
    dim = find(size(x)~=1,1);
  end

  s = size(x);

  if dim > length(s)           % If dimension is too high, just return input.
    y = x;
    return
  end

  % Sort along given dimension
  x = sort(x,dim);

  nCompare = s(dim);           % Number of elements used to generate a median
```

```
half = floor(nCompare/2);    % Midway point, used for median calculation

if dim == 1
  % If calculating along columns, use vectorized method with column
  % indexing.  Reshape at end to appropriate dimension.
  y = x(half+1,:);
  if 2*half == nCompare
    y = meanof(x(half,:),y);
  end

  y(isnan(x(nCompare,:))) = NaN;   % Check last index for NaN

elseif dim == 2 && length(s) == 2
  % If calculating along rows, use vectorized method only when possible.
  % This requires the input to be 2-dimensional.   Reshape at end.
  y = x(:,half+1);
  if 2*half == nCompare
    y = meanof(x(:,half),y);
  end

  y(isnan(x(:,nCompare))) = NaN;   % Check last index for NaN

else

  cumSize = cumprod(s);
```

```
total = cumSize(end);           % Equivalent to NUMEL(x)
numMedians = total / nCompare;

numConseq = cumSize(dim - 1);    % Number of consecutive indices
increment = cumSize(dim);        % Gap between runs of indices
ixMedians = 1;

y = repmat(x(1),numMedians,1);   % Preallocate appropriate type

% Nested FOR loop tracks down medians by their indices.
for seqIndex = 1:increment:total
  for consIndex = half*numConseq:(half+1)*numConseq-1
    absIndex = seqIndex + consIndex;
    y(ixMedians) = x(absIndex);
    ixMedians = ixMedians + 1;
  end
end

% Average in second value if n is even
if 2*half == nCompare
  ixMedians = 1;
  for seqIndex = 1:increment:total
    for consIndex = (half-1)*numConseq:half*numConseq-1
      absIndex = seqIndex + consIndex;
      y(ixMedians) = meanof(x(absIndex),y(ixMedians));
```

```
        ixMedians = ixMedians + 1;
      end
    end
  end

  % Check last indices for NaN
  ixMedians = 1;
  for seqIndex = 1:increment:total
    for consIndex = (nCompare-1)*numConseq:nCompare*numConseq-1
      absIndex = seqIndex + consIndex;
      if isnan(x(absIndex))
        y(ixMedians) = NaN;
      end
      ixMedians = ixMedians + 1;
    end
  end
 end

 % Now reshape output.
 s(dim) = 1;
 y = reshape(y,s);
end

%============================
```

```
function c = meanof(a,b)


c = a + (b-a)/2;
k = (sign(a) ~= sign(b)) | isinf(a) | isinf(b);
c(k) = (a(k)+b(k))/2;


function v = visort(v,N)
temp = v;


for i=1:N-1
   m = v(i);
   k = 1;


   for j = i+1:N
      if v(j) < m
         m = v(j);
         k = j-i+1;
      end
   end


   for j = 1:k-1
      v(i+j) = temp(i+j-1);
   end


   v(i) = m;
```

```
    for j=1:N
        temp(j) = v(j);
    end
end
```

```
function r = REffmed(Xmed,Xmax,Xmin)
i=1;
j=1;
while Xmed == Xmin || i < 5
        if Xmed == Xmax
```

$$Xmed = \left(\frac{K1 + K2}{4} \right)$$

```
        Else
i = i + 1;
j = j + 4;
Xmed = visort(j);
        end
end
```

```
function r = LEffmed(Xmed,Xmax,Xmin)
i=1;
j=6;
while Xmed == Xmin || i < 5
        if Xmed == Xmax
```

$$Xmed = \left(\frac{K1 + K2}{4} \right)$$

```
        Else
i = i + 1;
j = j - i;
Xmed = visort(j);
        end
end
```



```
function outbuf = remove_distortions_(inbuf)

numpixels = length(inbuf);

tbuf = inbuf;

for ii=eml.unroll(1:numpixels)
   if bitand(ii,uint32(1)) == 1
      tbuf = compare_stage1(tbuf);
   else
      tbuf = compare_stage2(tbuf);
   end
end

outbuf = tbuf;
```

```
 end


function outbuf = compare_stage1(inbuf)
numpixels = length(inbuf);
tbuf = compare_stage(inbuf(1:numpixels-1));
outbuf = [tbuf(:)' inbuf(numpixels)];
end


function outbuf = compare_stage2(inbuf)
numpixels = length(inbuf);
tbuf = compare_stage(inbuf(2:numpixels));
outbuf = [inbuf(1) tbuf(:)'];
end


function [outbuf] = compare_stage(inbuf)


step = 2;
numpixels = length(inbuf);


outbuf = inbuf;


for ii=eml.unroll(1:step:numpixels)
    t = compare_pixels([inbuf(ii), inbuf(ii+1)]);
    outbuf(ii) = t(1);
    outbuf(ii+1) = t(2);
```

```
end


end


function outbuf = compare_pixels(inbuf)
if (inbuf(1) > inbuf(2))
    outbuf = [inbuf(1), inbuf(2)];
else
    outbuf = [inbuf(2), inbuf(1)];
end
end
```