TRIBHUVAN UNIVERSITY
CENTRAL DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY CAMPUS
KIRTIPUR

Dissertation

COMPARATIVE ANALYSIS OF SPATIAL FILTERS FOR IMAGE RECTIFICATION AND COMPUTATIONAL COMPLEXITY

(For the Partial Fulfillment of the Requirement for the Degree of Master of Science in Computer Science and Information Technology)

Pushpa Parajuli
November, 2008

COMPARATIVE ANALYSIS OF SPATIAL FILTERS FOR IMAGE RECTIFICATION AND COMPUTATIONAL COMPLEXITY

By
Pushpa Parajuli

(For the Partial Fulfillment of the Requirement for the Degree of Master of Science in Computer Science and Information Technology)

Supervisor: Prof. Dr. Shashidhar Ram Joshi

Previous Degree:

B.Sc. in Computer Science
Gandaki College of Computer Science
Lamachaur, Pokhara, Nepal

Tribhuvan University
Institute of Science and Technology
Central Department of Computer Science and Information Technology

Certificate

This is to certify that the thesis titled “Comparative Analysis of Spatial Filters for Image Rectification and Computational Complexity", submitted by Pushpa Parajuli in partial fulfillment of the requirement for the award of the degree of Master of Science in Computer Science and Information Technology has been carried out under my supervision and guidance. The thesis fulfills the requirement related to the nature and standard of the work for the award of Master of Science in Computer Science and Information Technology. In my best knowledge this is an original work in computer science and no part of this dissertation has been published or submitted for the award of any degree else where in the past.

Prof. Dr. Shashidhar Ram Joshi,
Institute of Engineering (IOE)
Pulchowk Tribhuvan University
(Supervisor)

Approval

We certify that we have read this dissertation entitled "Comparative Analysis of Spatial Filters for Image Rectification and Computational Complexity" and in our opinion, it is satisfactory in the scope and quality as a dissertation in the partial fulfillment for the requirement of Master's Degree in Computer Science and Information Technology.

Evaluation Committee

Prof. Dr. Shashidhar Ram Joshi,
Institute of Engineering (IOE)
Pulchowk Tribhuvan University
(Supervisor)

Dr. Tanka Nath Dhamala,
Head,
Centeral Department of Computer Science
and Information Technology,
Tribhuvan University, Nepal

Mr. Roshan Chitrakar
Nepal College of Information
Technology (NCIT)
Pokhara University
(External Examiner)

Dr. Subarna Shakya
Institute of Engineering (IOE)
Pulchowk, Tribhuvan University
(Internal Examiner)

Date: ……

Acknowledgement

A dissertation of this nature calls for intellectual nourishment, and help and encouragement from different individuals. I would not have been able to justify my work without acknowledging those who helped and guided me to prepare this thesis report. I would like to extend my heartiest gratitude to my advisor Prof. Dr. Shashidhar Ram Joshi for his valuable suggestions, guidance and continuous inspiration throughout this research.

I also owe special thanks to Dr. Tanka Nath Dhamala and all other teachers of Central Department of Computer Science and Information Technology for their guidance and suggestions.

I would like to express my sincere gratitude to all my colleagues', family and other friends, for their direct and indirect support in completing this thesis.

Pushpa Parajuli
November, 2008

Table Of Contents

List of Figures

List of Tables

List of Charts

List of Abbreviations

R.M.S	**Root Mean Square**
S.D.	**Standard Deviation**
Err.	**Error**
A.M.	**Arithmetic Mean**
V.	**Variance**
MS	**Milisecond**

ABSTRACT

Image Rectification is an important preprocessing step in many fields such as computer vision, pattern matching and many real time image-processing applications. Rectifying different things is image Rectification. But here the algorithm that helps to rectify impurity presents in image is proposed. This impurity can be noise and distortions. The sources of impurity in digital images arise during image acquisition (digitization) and /or transmission. Impurity has two kinds of properties 1) Spatial properties 2) Frequency properties. Frequency properties refer to the frequency content of impurity in the Fourier sense. The spatial characteristics are concerned with statistical behavior of the gray level values in the impurity component. So the proposed algorithm focuses in spatial properties of impurity. There are two types of spatial filters. But noise reduction can be achieved effectively with a nonlinear filter whose basic function is to compute the median gray-level value in the neighborhood in which the filter is located.

There exist many works on minimizing impurity of images by using spatial filters. The most widely used spatial filters are Median Filter, Kuwahara filter and Gaussian filter. All are responsible to minimize impurity to some extent. The proposed algorithm that helps to minimize impurity is better than these filters mentioned above. Here, checking of the minimization of error is done by calculating RMS errors between input image and output image. The proposed algorithm is also based on the spatial filters. Experiment showed that the final rectified images are satisfactory and it makes it easy for further image manipulation.

The efficiency of an algorithm depends on the complexity (time and space). Time complexity is total time required for execution, whreas space complexity is total memory space required for execution of an algorithm. Complexity of Median Filter, Kuwahara filter and Gaussian filter and proposed filter are analysed for better efficiency.

Keywords: Noise, Distortions, Acquisition, Spatial, Impurity, Illumination,Gray level, Filter mask, Root Mean Square Error, Pixel

CHAPTER 1
1. INTRODUCTION

Image filtering is done to remove noise, sharpen contrast, or highlight contours in digital images. This document will discuss the basic distinctions between types of filters and some of the uses for each[2,18]. Two of the most common classifications of filters are based on their linearity and frequency response. Spatial filters whose response is based on ordering (ranking) the pixels contained in the image area encompassed by the filter[8], and then replacing the value of the center pixel with the value of determined by the ranking result. The best-known example in this category 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. In digital image application, impurity may cause many problems. On the first steps we minimize the impurity and make the image smooth. Such process makes further image manipulations easier. Mainly there are two kinds of spatial filters a) Linear filters b) Non Linear filters. For linear spatial filtering, the response is given by a sum of products of the filter coefficients and the corresponding image pixels in the area spanned by the filter mask. In linear spatial filtering the same operation is applied to each pixel location. The process of linear filtering is similar to a frequency domain concept called convolution. For this reason linear spatial filtering is often referred to as convolving a mask with an image. Non-linear filters which are not space invariant; these attempt to locate edges in the noisy image before applying smoothing, a difficult task at best, in order to reduce the blurring of edges due to smoothing[1,11].

In proposed algorithm, first of all, system takes a matrix of size 5*5 and divides that matrix into five regions. A pixel is always influenced by its near pixels so system assigns more weight to those pixels in each region, which is near the considered pixel. Different regions have different variances. The variance of each region is measured and mean brightness is calculated of that region which has less variance. The output value of the center pixel in the window is the mean value of that region that has smallest variance.

The complexity of computational problems can be discussed by choosing a specific machine as a model of computation and considering how much time and/or space the machine of that type requires for the solution of that problem. The quality (efficiency) of such computational problems can be measured in terms of the resources needed by the algorithm for its execution. The two important resources used for executing a given algorithm are time and memory, that are required to execute that algorithm[6,9].

2. BACKGROUND

Smoothing filters are used for blurring and noise reduction. Blurring is used in preprocessing steps, such as removal of small details from an image. Linear spatial filter is simply the average of the pixel contained in the neighborhood of the filter mask. These filters are sometimes, called averaging filters. The idea behind smoothing filters is straightforward. By replacing the value of very pixels in an image by the average of the gray levels in the neighborhood defined by filters masks. This process results in an image with reduced sharp transition in gray levels. The most obvious application of smoothing is noise reduction. However, edges (which almost always are desirable features of image) also are characterized by sharp transition in gray levels, so averaging filters have the undesirable side effect that they blur edges. Another application of this type of process includes the smoothing of false contours that result from using an insufficient number in gray levels. A major use of averaging filter is in the reduction of irrelevant details in an image. By irrelevant details mean, pixel regions that are small with respect to the size of the filter mask.

In simple smoothing filter, average of the gray levels of the pixel in the size 3 * 3 neighborhood is defined by the mask. The coefficients of the filter are all 1's. The idea here is that it is computationally more efficient to have coefficients valued 1. At the end of the filtering process I divide the entire image. A spatial averaging filter in which all coefficients are equal is sometimes called a box filter. The second mask is called weight average, the terminology are used to indicate that pixel are multiplied by different coefficients, thus giving more importance (weight) to some pixel at the expense of others.

The pixel at the center of the mask is multiplied by a higher value than any other, thus giving this pixel more importance in the calculation of the average[13,17]. The other pixels are inversely weighted as a function of their distance from the center mask. An important application of spatial averaging is to blur an image for the purpose of getting a gross representation of objects of interest, such that the intensity of smaller objects blends. Many impurity check rectification algorithms work based on spatial filtering methods but they carry the following limitations:

- Disturbance the sharp of image
- High time complexity
- Support small Image size
- Minimize impurity in little

So, minimizing impurity in image is still an area where a lot more research is needed to overcome limitations explored.

3. PROBLEM STATEMENT

Impurity has different nature. It is one of the difficult tasks to calculate the nature of impurity. They may arise from various sources and processes. If we have to determine nature of impurity, impurity model should be used. According to the nature of impurity its reducing process can be different. Complexity is different for different algorithms. It depends on the number of variables, loops and program instruction. According to the nature of an algorithm complexity varies.

4. RESEARCH OBJECTIVE

Objective behind this research work is to make a spatial based filter to minimize the impurity of an image without disturbing the nature and sharpness of image. The proposed algorithm will be faster then the existing filter and complexity of different existing filters as well as proposed algorithm will be analysed.

5. PROCEDURE

The procedure applied in proposed algorithm are given below

1. Take size of 5*5 matrix
2. Divide it into 5 regions
3. Assign weightage to the pixels
4. Find region with minimum variance
5. Find mean brightness of region of minimum variance
6. The output value of the center pixel in the window is the mean value of that region that has smallest variance

The procedure for complexity are given below

i. Time Requirement:

- Starting time of an algorithm is captured
- Stoping time of an algorithm is captured
- Total time for executing an algorithm is calculated

ii. Storage Requirement:

- Number of variables and loops are detected
- Memory required for those variables are found
- Total memory space required for an algorithm is calculated

CHAPTER 2

LITERATURE REVIEW

1. IMAGE ACQUISITION

Image acquisition is the first process in digital image processing. It gives information about the origin of digital images. Generally, the image acquisition stage involves preprocessing, such as scaling. Images which are generated by the combination of an "illumination" source and the reflection or absorption of energy from that source by the elements of the "scene" being imaged[10]. The illumination may originate from a source of electromagnetic energy such as radar, infrared, or x- ray energy. It could originate from less traditional sources, such as ultrasound or even a computer-generated illumination pattern. Similarly, the scene elements could be familiar object, but they can just as easily be molecules, buried rock formation, or a human brain. Depending on the nature of the source, illumination energy is reflected from, or transmitted through, objects. An example in the first category is light reflected from a planar surface. An example in the second category is when X-rays pass through a patient' body for the purpose of generating a diagnostic X-ray film[8]. In some application, the reflected or transmitted energy is focused onto a photo converter (e.g., phosphor screen), which converts the energy into visible light. There are three principal sensor arrangement used to transform illumination energy into digital images

1.1 Image acquisition using a single sensor

The most familiar sensor of this type is photodiode, which is constructed of silicon materials and whose output voltage waveform is proportional to light. In order to generate a 2-D image using single sensor, there has to be relative displacement in both the x-and y-directions between the sensor and the area to be imaged. It is used in high precision scanning, where a film negative is mounted onto a drum whose mechanical rotation provides displacement in one dimension. The single sensor is mounted on a lead screw that provides motion in the perpendicular direction[8].

1.2 Image Acquisition using sensor strips

A geometry that is used much more frequently that single sensors consists of an in-line arrangement of sensor in the form of a sensor strip. The strip provides imaging elements in one direction. Motion perpendicular to the strip provides imaging in the other directions. This is the type of arrangement used in most flat bed scanners. Sensing devices with 4000 or more in line sensors are possible. Inline sensor are used routinely in airborne imaging applications in which the imaging system is mounted on an aircraft that flies at constant altitude and speed over the geographical area to be imaged. The imaging strip gives one line of an image at a time, and the motion of strip completes the other dimension of a two dimensional image. Sensor strips mounted in a ring configuration are used in medical and industrial imaging to obtain cross sectional images of 3D objects. A rotating X- ray source provides illumination and the portion of the sensors opposite the source collect the X-ray energy that pass through the object.

1.3 Image Acquisition using Sensor Arrays

In this approach, numerous electromagnetic and some ultrasonic sensing devices are frequently arranged in an array format. This is also the predominant arrangement found in digital cameras. A typical sensor for these cameras is a CCD array, which can be manufactured with a broad range of sensing properties and can be packaged in rugged array of size 4000* 40000 elements or more. CCD sensors are used widely in digital cameras and others light sensing instruments. The response of each sensor is proportional to the integral of the light energy projected onto the surface of the sensor, a property that is used in astronomical and other applications requiring low noise images. The sensor array, which is coincident with the focal plane, produces outputs proportional to the integral of the light received at each sensor.

2. A SIMPLE IMAGE FORMATION MODEL

Any image can be denoted by two-dimensional function of the form f (x, y). The value or amplitude of f at spatial coordinates (x, y) is a positive scalar quantity whose physical meaning is determined by the source of the image. Most of images are said to span the gray scale. When image is generated from a physical process, its values are proportional to energy radiated by a physical source (e.g., electromagnetic waves). As a consequence, f (x, y) must be non zero and finite that is $0<f(x, y)<\infty$.

The function f(x y) may be characterized by two components: 1) the amount of source illumination incident on the scene being viewed , and 2) the amount of illumination reflected by the object in the scene. So there are called the illumination and reflectance components and are denoted by i (x, y) and r (x, y). The two functions combine as a product to form f (x, y)

$$f(x, y)= i(x, y)\, r(x, y)$$

Where $0<i(x, y)<\infty$ and $0<r(x, y)<1$

The nature of i (x, y) is determined by the illumination source and r (x, y) is determined by the characteristics of the images objects.

3. DIGITIZATION OF IMAGE

Digital image consists of $N \times M$ pixels, each represented by k bits. A pixel can thus have $2k$ different values. In practical applications, the pixel values are considered as integers varying from 0 (black pixel) to $2k$-1 (white pixel)[12]. The images are obtained through a digitization process, in which a two-dimensional sampling grid covers the object. The main parameters of the digitization are:

• Image resolution: the number of samples in the grid.

• Pixel accuracy: how many bits are used per sample.

These two parameters have a direct effect on the image quality but also to the storage size of the image. In general, the quality of the images increases as the resolution and the bits per pixel increase. There are a few exceptions when reducing the number of bits increases

the image quality because of increasing the contrast. Moreover, in an image with a very high resolution only very few gray-levels are needed. In some applications it is more important to have a high resolution for detecting details in the image whereas in other applications the number of different levels (or colors) is more important for better outlook of the image. To sum up, if we have a certain amount of bits to allocate for an image, it makes difference how to choose the digitization parameters. The properties of human eyes imply some upper limits. For example, it is known that the human eye can observe at most one thousand different gray levels in ideal conditions, but in any practical situations 8 bits per pixel (256 gray level) is usually enough. The required levels decreases even further as the resolution of the image increases. In a laser quality printing, as in this lecture notes, even 6 bits (64 levels) results in quite satisfactory result. On the other hand, if the application is e.g. in medical imaging or in cartography, the visual quality is not the primary concern. For example, if the pixels represent some physical measure and/or a computer will analyze the image, the additional accuracy may be useful. Even if human eye cannot detect any differences, computer analysis may recognize the differences. The requirement of the spatial resolution depends both on the usage of the image and the image content. If the default printing (or display) size of the image is known, the scanning resolution can be chosen accordingly so that the pixels are not seen and the image appearance is not jagged (blocky). However, the final reproduction size of the image is not always known but images are often achieved just for "later use". Thus, once the image is digitized it will most likely (according to Murphy's law) be later edited and enlarged beyond what was allowed by the original resolution. The image content sets also some requirements to the resolution. If the image has very fine structure exceeding the sampling resolution, it may cause so-called aliasing effect where the digitized image has patterns that does not exists in the original

4. IMAGE ENHANCEMENT

Image enhancement approaches fall into two broad categories: Spatial domain methods and frequency domain methods[4,18]. The term spatial domain refers to the image plane itself, and approaches in this category are based on direct manipulation of pixels in an

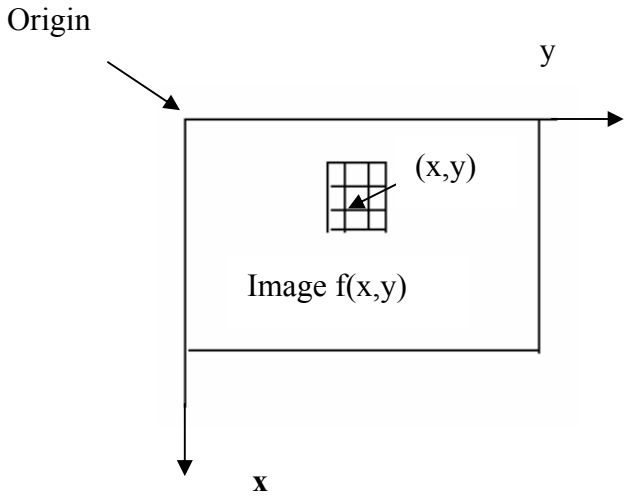
image. Frequency domain processing techniques are based on modifying the Fourier transform of an image. Spatial domain processes will be denoted by the expression

$$g(x, y)=T[f(x, y)]$$

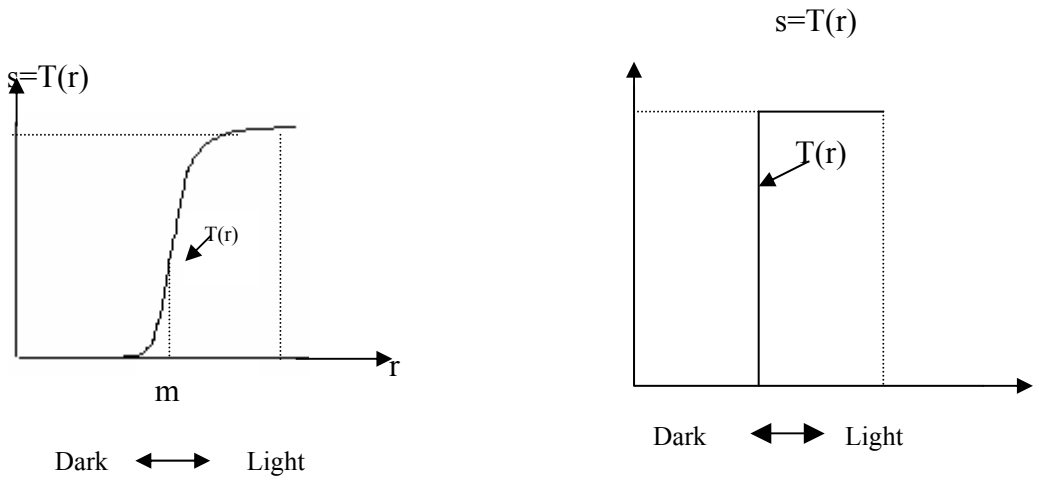
Where f (x, y) is the input image, g(x, y) is the processed image, and T is an operator on f, defined over some neighborhood of (x, y). In addition, T operates on a set of input images, such as performing the pixel- by-pixel sum of K image for noise reduction. The principle approach in defining a neighborhood about a point (x, y) is to use a square or rectangular sub image area centered at (x, y). The center of the sub image is moved from pixel to pixel starting, say, at the top left corner. The operator T is applied at each location (x, y) to yield the output, g, at that location. The process utilizes only the pixels in the area of the image spanned by the neighborhood. Although other neighborhood shapes, such as approximations to a circle, sometimes are used, square and rectangular arrays are by far the most predominant because of their ease of implementation. The simplest form of T is when the neighborhood is of size 1*1 (that is , a single pixel). In this case , g depends only on the value of f at (x, y) and T becomes a gray-level (also called an intensity or mapping) transformation function of the form

$$s=T(r)$$

Where, for simplicity in notation, r and s are variables denoting, respectively, the gray level of f (x, y) and g(x, y) at any point (x, y)



Figures 1: 3*3 neighborhood about a point (x, y) in an image



Figures 2:Gray level transformation functions for contrast enhancement

If T(r) has the form shown in fig the effect of this transformation would be to produce an image of higher contrast than the original by darkening the levels below m and brightening the levels above m in the original image. In this technique known as contrast stretching, the values of r below m are compressed by the transformation function into a narrow range of s, towards black. Enhancement at any point in an image depends only on the gray level at that point. This technique is known as point processing. The general approach is to use a function of the values of f in a predefined neighborhood of (x, y) to determine the value of g at (x, y). One of the principal approaches in this formation is based on the use of so called masks (also referred as filters, kernels, templates or windows). Enhancement of these techniques is known as mask processing or filtering.

5. SMOOTHING SPATIAL FILTERS

Smoothing filters are used for blurring and noise reduction. Blurring is used in preprocessing steps, such as removal of small details from an image prior to object extraction and bridging of small gaps in lines or curves. Noise reduction can be accomplished by blurring with linear filter and also by non-linear filtering

5.1 Filtering

Filtering is an image processing operation where the value of a pixel depends on the values of its neighbor pixels. Each of the pixels is processed separately with a predefined *window* (or *template,* or *mask*). Weighted sum of the pixels inside the window is calculated using the weights given by a mask. The result of the sum replaces the original value in the processed image.

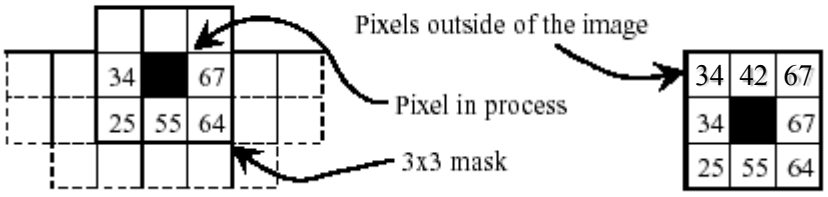
$$f(x) = \sum_{i=1}^{m*n} w_i . x_i$$

where m is the number of row and n is the number of column.

In case of border pixels, the part of the mask lying outside of the image is assumed to have the same pixel values as that of the border pixels. The filtering is a parallel operation, i.e. the neighboring values used in the calculations are always taken from the original image, not from the processed image.

W1	**W2**	**W3**
W4	**W5**	**W6**
W7	**W8**	**W9**

Figures 3: General mask for filtering with a 3·3 window.



Figures 4: Example of filtering operation in the case of border pixel.

5.2 Smoothing Linear Filter

The output of a smoothing, linear spatial filter is simply the average of the pixels contained in the neighborhood of the filter mask. These filters are called averaging filter and they also referred to low pass filters. Low-pass filtering (or averaging filtering, or smoothing) reduces the high frequency components (or noise) in the image by averaging the pixel values over a small region (block). This reduces noise and makes the image generally smoother, especially near the edges. The level of smoothing can be changed by increasing the size of the window. High-pass filtering is the opposite operation to low-pass filtering. The low frequency components are eliminated and only the high frequency components in the image are retained. The operation can be applied in image enhancement by adding the result of the filtering to the original image. This is known as sharpening. It enhances the pixels near edges and makes it easier to observe details in the image. In uniform filter, the output image is based on a local averaging of the input filter where all of the values within the filter support have the same weight. In Triangular filter the output image is based on a local averaging of the input filter where the values within the filter support have differing weights. In Gaussian filte, the use of the Gaussian kernel for smoothing has become extremely popular. This has to do with certain properties of the Gaussian (e.g. the central limit theorem, minimum space-bandwidth product) as well as several application areas such as edge finding and scale space analysis. It helps to remove noise but smears the image [18].

5.3 Smoothing Non Linear Filter

Statistics filters are nonlinear spatial filter whose response is based on ordering the pixels contained in the image area encompassed by the filter, and then replacing the value of the center pixel with the value determined by the ranking result. Low-pass and high-pass filters are in the class of linear filters; a weighting mask can always describe them. Median filtering, on the other hand, belongs to a class of rank filters. Here the pixels within the window are ranked (or sorted) and the result of the filtering is chosen according to the ordering of the pixel values. In median filtering the new value of a pixel is the median of the pixel values in the window. The parameter of the filtering is the size

and the shape of the filtering window (mask). The median filter is used for removing noise. It can remove isolated impulsive noise and at the same time it preserves the edges and other structures in the image. Contrary to average filtering it does not smooth the edges. Edges play an important role in our perception of images as well as in the analysis of images. It is important to be able to smooth images without disturbing the sharpness and, if possible, the position of edges. A filter that accomplishes this goal is termed an edge-preserving filter and one particular example is the Kuwahara filter. Although this filter can be implemented for a variety of different window shapes, the algorithm will be described for a square window of size J = K = 4L + 1 where L is an integer. The window is partitioned into four regions.

6. NOISE

The principal sources of noise in digital images arise during image acquisition (digitization) and / or transmission. The performance of imaging sensor is affected by a variety of factors, such as environment conditions during image acquisition, and by the quality of the sensing elements themselves. For instance in acquiring images with a CCD camera, light levels and sensor temperatures are major factors affecting the amount of noise in the resulting image. Images are corrupted during transmission principally due to interference in the channel used for transmission. For example, an image transmitted using a wireless network might be corrupted as a result of lighting or other atmospheric disturbance[8,18].

Noise has spatial and frequency properties. Frequency properties refer to the frequency content of noise in the Fourier sense (i.e., as opposed to the electromagnetic spectrum). For example, when the Fourier spectrum of noise is constant, the noise usually is called white noise. This terminology is a carry-over from the physical properties of white light, which contains nearly all frequencies in the visible spectrum in equal proportions.

7. COMPUTATIONAL COMPLEXITY

Computational complexity considers not only whether a problem can be solved at all on a computer, but also how efficiently the problem can be solved.For it two major aspects are considered: time complexity and space complexity, which are how many steps does it

take to perform a computation, and how much memory is required to perform that computation respectively[9].

In order to analyze how much time and space a given algorithm requires, express the time or space required to solve the problem as a function of the size of the input problem. For example; finding a particular number in a long list of numbers becomes harder as the list of numbers grows larger. If there are *n* numbers in the list, then if the list is not sorted or indexed in any way you may have to look at every number in order to find the number beeing sought. In order to solve this problem, a computer needs to perform a number of steps that grows linearly in the size of the problem.

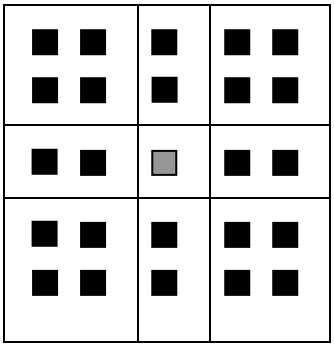
To simplify this problem Big O notation, which allows functions to be compared in a way that ensures that particular aspects of a machine's construction does not need to be considered, rather only the asymptomatic behavior on problems become large.

CHAPTER 3

IMPLEMENTATION DETAILS

1. TAKE 5*5 MATRIX

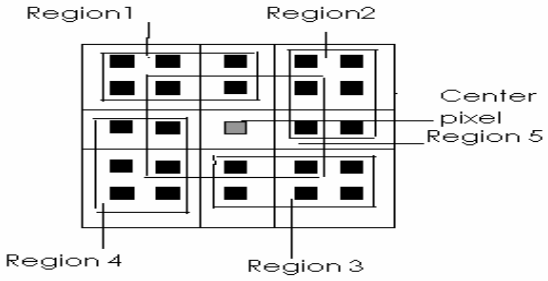
First of all we have to take square window of size five[4]. So it is matrix of five by five. Then there are total twenty-five pixels in that matrix. The below figure shows the matrix of proposed algorithm.



Figures 5: Shows matrix of size 5 * 5

2. DIVIDE IT INTO 5 REGIONS

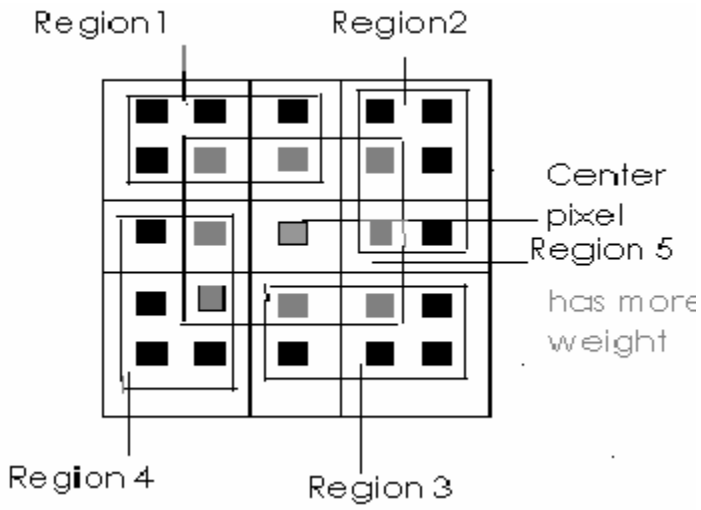
All twenty-five pixels are allocated into five regions. There are four regions at boundary and one is at the center. The given figure shows how to allocate the pixels into different regions.



Figures 6: Show Region Allocation

3. ASSIGNING WEIGHTS TO PIXELS

Center pixel is affected from its near pixels. So near pixels give more weight than other pixels .In the fig pixel in region five are near to center than others so that they get more weight than others. They are shown in different colors



Figures 7: Assigning weightage values to pixels

4. FIND REGION WITH MINIMUM VARIANCE

4.1 Standard deviation

The standard deviation, s_a, of the brightnesses within a region ($\Re$) with Δ pixels is called the sample standard deviation and is given by:

$$S_a = \sqrt{\frac{1}{\Delta - 1}\sum_{m\pi\in\Re}(a[m,n] - m_a)^2}$$

$$= \sqrt{\frac{\sum_{m\pi\in\Re} a^2[m,n]\Delta - m^2{}_a}{\Delta - 1}}$$

where Δ is the number of pixels.

The square of the standard deviation is known as the variance. So that region of minimum variance is calculated.

5. FIND MEAN BRIGHTNESS OF REGION OF MINIMUM VARIANCE

5.1 Average

The average brightness of a region is defined as the *sample mean* of the pixel brightness within that region. The average, m_a, of the brightnesses over the pixels within a region ($\Re$) is given by:

$$m_a = \frac{1}{\Delta} \sum_{(m.\pi)\in\Re} a[m,n]$$

The output value of the center pixel in the window is the mean value of that region that has smallest variance At last center pixel has equal value of mean value of that region which as smallest variance. That value is assigned to that center pixel. From all above process all new images will form.

6. FIND COMPUTATIONL COMPLEXITY FOR IMAGE FILTER

6.1 Time Requirement

Time complexity is total time required for execution of an algorithm. Overall filtering could caried out in time O(n) then the overall running time for each filter is given by *O(n log n).* If we consider one process takes 1 millisecond then for 100 pixels, the computation of the filter takes 100 millisecond.

6.2 Storage Requirement

Space complexity is the total memory space required for execution of an algorithm. If filter requires n number of variables then storage requirement for all algorithm is given by $O(n^2)$ where n is the number of variables because all algorithm uses nested loop.

6.3 Complexity per pixel

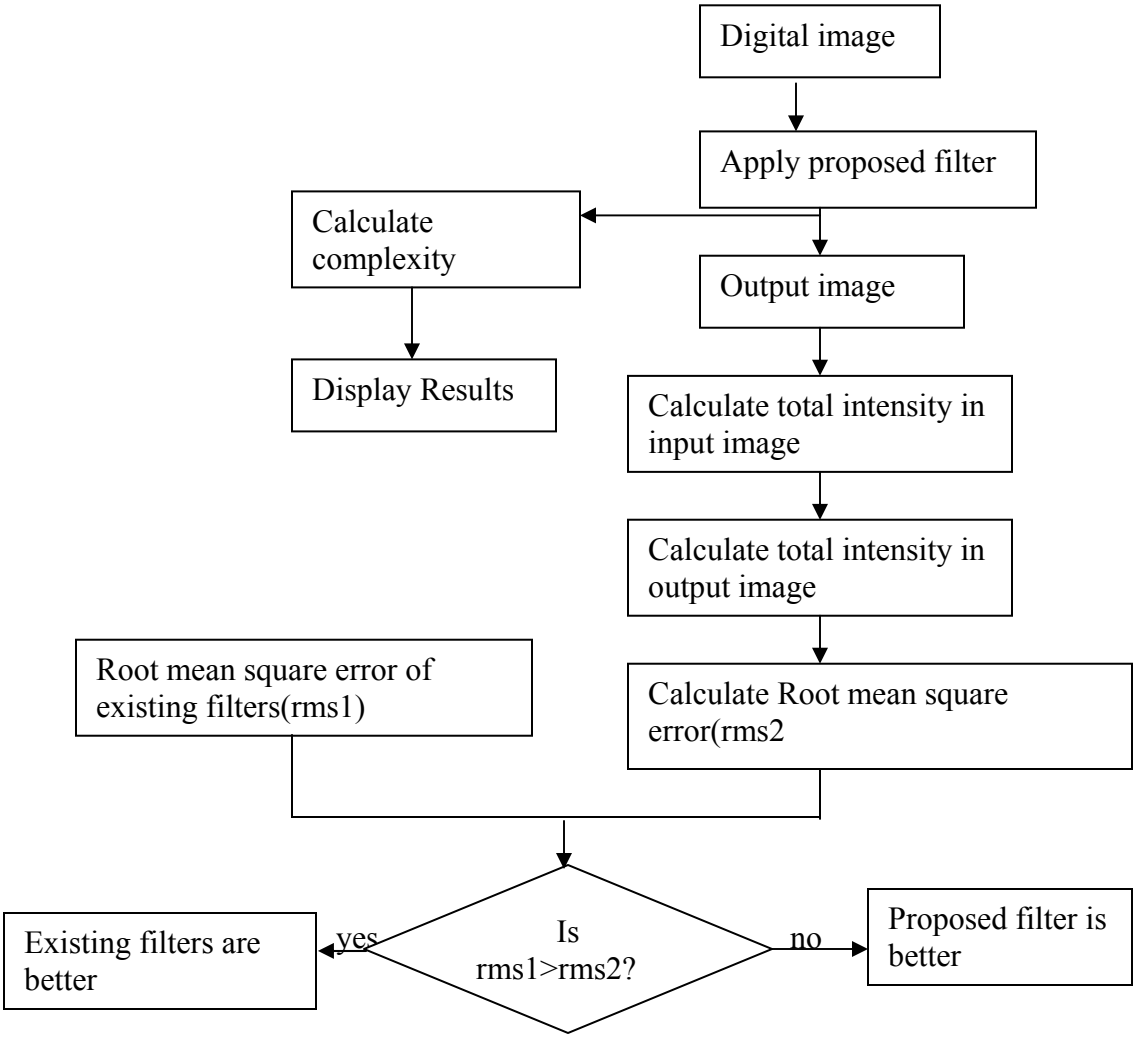
Algorithm	Domain	Complexity per pixel
Median	Space	O(K)[a]
Gussian	Space	O(Constant)[a]
Kuwahara	Space	O(J* K)
Algorithm	Space	O(J* K)

Table 1: Complexity per pixel

CHAPTER 4

SYSTEM PROCESS DESCRIPTION

Figure8 below shows the overall process of the proposed Comparative study of image rectification on spatial domain



Figures 8: Show process of System

First of all any digital image is provided as an input image for the system. Then, filtering options are provided. Any types of filter can choose for minimizing impurity. For that purpose there are most popular spatial filter such as median filter, Kuwahara filter, and Gaussian filter. These entire filters are widely used for image processing application. So I choose this filter for my analysis. We can choose any type of image filters for the first step,then, total intensity of input images is calculated. The output of that filter appeares. Total intensity of output image is also calculated. Root mean square error is calculated to determine how much impurity is reduced from that filter. From that process we can get RMS error of all filters.

Similarly,we can apply my proposed algorithms and total intensity of input and output image is calculated. Root Mean Square error is also calculated. We can compare RMS error values of my proposed algorithms and exiting popular filters. The RMS error is less than RMS error of exiting filters.

1. To calculate the total intensity

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 pair of coordinates (x, y) is called the intensity or the gray level of the image at that point . When x, y and the amplitude values of f are all-finite, we call the image a digital image. 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 or pixels. So we have sum of intensity values of pixels

$$TotalIntensitySum = \sum_{X=0,Y=0}^{m-1,n-1} \sum f(X,Y)$$

2. Calculate the total intensity in output images

After filtering the image we get new output image. This image has different total intensity than the original image. We have calculated the total intensity of output images to determine impurity reduction ratio.

$$TotalIntensitySum1 = \sum_{X=0,Y=0}^{m-1,n-1} \sum f'(X,Y)$$

3. To Calculate Root Mean square (RMS) error

Let f (x, y) represents an input image and let f' (x, y) denote an output image. For any values of x and y , the error e(x, y) between f(x, y) and f'(x, y) can be defined as

$$e(x,y) = f'(x,y) - f(x,y)$$

So that the total error between the two images is

$$TotalErr = \sum_{X=0,Y=0}^{m-1,n-1} f'(X,Y) - \sum_{X=0,Y=0}^{m-1,n-1} f(X,Y)$$

where the size of image is m*n. The Root mean square error, e_{rms} between f(x, y) and f'(x, y) is the square root of the squared error averaged over the m*n array

$$TotalErms = \left[\frac{1}{mn} * \sum_{X=0,Y=0}^{m-1,n-1} \left[f'(X,Y) - \sum_{X=0,Y=0}^{m-1,n-1} f(X,Y) \right]^2 \right]^{1/2}$$

4. Comparison of RMS error

We get RMS error from above formula. Then, the RMS of image for all filters is calculated. Then we can easily compare each other. In above RMS calculation process, which has little values then, we can easily guess that the process reduces more impurity then others.

```
If (rms1>rms2)
Then rms2 has less impurity then rms1.
 If (rms1<rms2)
Then rms1 has less impurity then rms2.
```

5. Compariosn of Computationl Complexity

Algorithm	Domain	Complexity per pixel
Median	Space	O(K)[a]
Gussian	Space	O(Constant)[a]
Kuwahara	Space	O(J* K)
Proposed algorithm	Space	O(J* K)

Table 2:Complexity per pixel

CHAPTER 5

EXPERIMENTAL RESULTS

Model prototyping shows satisfactory result for small and medium image databases.

1. Experimental Result for Kuwahara Filter.

Image	Size (KB)	Total intensity in input image	Total intensity in output image	Root Mean Square Error (RMS)	Time Complexity (MS)
1wall.bmp	104	33539	7189	11.238665	1400
2dbox.bmp	105	42194	3417	11.125982	1350
3dbar.bmp	102	39105	595	11.093336	1330
4dcman.bmp	107	40712	595	11.014090	1335
5dwall2.bmp	100	44551	8791	11.190162	1320
6droad.bmp	118	26403	5088	11.118701	1315
7dcave.bmp	117	34610	595	11.182601	1318
8ddoor.bmp	122	28985	1531	11.087574	1298
9win.bmp	122	44447	7815	11.191154	1322
10dwomen.bmp	114	34500	580	11.192502	1315
11tree.bmp	125	45337	8016	11.201254	1350
12canvas.bmp	119	35710	600	11.21351	1345
13sat.bmp	116	34410	575	11.087574	1314
14chair.bmp	111	33300	570	11.092502	1310
15cap.bmp	120	42345	7790	11.190254	1400
16tab.bmp	114	34500	580	11.192502	1315
17dflower.bmp	103	39108	593	11.094120	1310
18light.bmp	102	39105	595	11.093336	1305
19plane.bmp	101	38100	570	11.1025663	1300
20flag.bmp	106	35600	615	11.123564	1325

Table 3: Experimental Result for Kuwahara Filter

2. Experimental Result for Gaussian Filter

Image	Size (KB)	Total intensity in input image	Total intensity in output image	Root Mean Square Error (RMS)	TimeComplexity (MS)
1wall.bmp	104	33539	33414	11.494037	3500
2dbox.bmp	105	42194	40770	11.420481	3200
3dbar.bmp	102	39105	38613	11.790548	3600
4dcman.bmp	107	40712	39476	11.263325	3100
5dwall2.bmp	100	44551	42414	11.260066	3075
6droad.bmp	118	26403	26397	12.004144	3700
7dcave.bmp	117	34610	34100	11.471344	3350
8ddoor.bmp	122	28985	28611	11.686043	3550
9windows.bmp	122	44447	42820	11.261982	3060
10dwomen.bmp	114	34500	33700	11.361892	3120
11tree.bmp	135	45337	44201	11.372194	3250
12canvas.bmp	119	35710	34601	11.481982	3340
13sat.bmp	116	34410	33901	11.384581	3512
14chair.bmp	111	33300	33201	11.215678	3542
15cap.bmp	120	42345	41100	11.345614	3312
16tab.bmp	114	34500	33200	11.361892	3405
17dflower.bmp	103	39108	38500	11.789145	3340
18light.bmp	102	39105	37900	11.568921	3350
19plane.bmp	101	38100	37500	11.723456	3345
20flag.bmp	106	35600	34905	11.452356	3354

Table 4: Experimental Result for Gaussian Filter

3. Experimental Result for Median Filter

Image	Size (KB)	Total Intensity in input image	Total intensity in output image	Root Mean Square Error (RMS)	Time Complexity (MS)
1wall.bmp	104	33539	33167	11.635696	1000
2dbox.bmp	105	42194	41821	11.548772	975
3dbar.bmp	102	39105	37725	11.501963	970
4dcman.bmp	107	40712	39045	11.323892	965
5dwall2.bmp	100	44551	44259	11.627915	972
6droad.bmp	118	26403	26376	11.584725	969
7dcave.bmp	117	34610	33367	11.542884	971
8ddoor.bmp	122	28985	28204	11.567791	968
9windows.bmp	122	44447	42820	11.625469	974
10dwomen.bmp	114	34500	33400	11.614521	978
11tree.bmp	135	45337	44100	11.586584	981
12canvas.bmp	119	35710	34600	11.605689	957
13sat.bmp	116	34410	33200	11.594789	956
14chair.bmp	111	33300	32900	11.594568	955
15cap.bmp	120	42345	41300	11.553689	958
16tab.bmp	114	34500	33400	11.578962	960
17dflower.bmp	103	39108	38801	11.568974	948
18light.bmp	102	39105	37850	11.568423	968
19plane.bmp	101	38100	37560	11.584326	959
20flag.bmp	106	35600	34580	11.598742	958

Table 5: Experimental Result for Median Filter

4. Experimental Result for Proposed Algorithm

Image	Size (KB)	Total intensity in input image	Total intensity in output image	Root Mean Square Error (RMS)	Time Complexity (MS)
1wall.bmp	104	33539	7174	11.227622	1800
2dbox.bmp	105	42194	3402	11.1173717	1750
3dbar.bmp	102	39105	580	11.084783	1675
4dcman.bmp	107	40712	579	11.0125803	1650
5dwall2.bmp	100	44551	8776	11.180870	1775
6droad.bmp	118	26403	5073	11.108418	1700
7dcave.bmp	117	34610	580	11.171830	1725
8ddoor.bmp	122	28985	1516	11.087012	1668
9windows.bmp	122	44447	7800	11.182632	1760
10dwomen.bmp	114	34500	575	11.0845235	1784
11tree.bmp	135	45337	8860	11.182546	1685
12canvas.bmp	119	35710	7450	11.1145627	1753
13sat.bmp	116	34410	560	11.065485	1758
14chair.bmp	111	33300	595	11.075421	1764
15cap.bmp	120	42345	8575	11.0845142	1784
16tab.bmp	114	34500	575	11.095125	1764
17dflower.bmp	103	39108	675	11.089512	1754
18light.bmp	102	39105	650	11.0854623	1768
19plane.bmp	101	38100	595	11.052658	1684
20flag.bmp	106	35600	560	11.0984251	1657

Table 6: Experimental Result for Proposed Algorithm

5. Experimental Charts

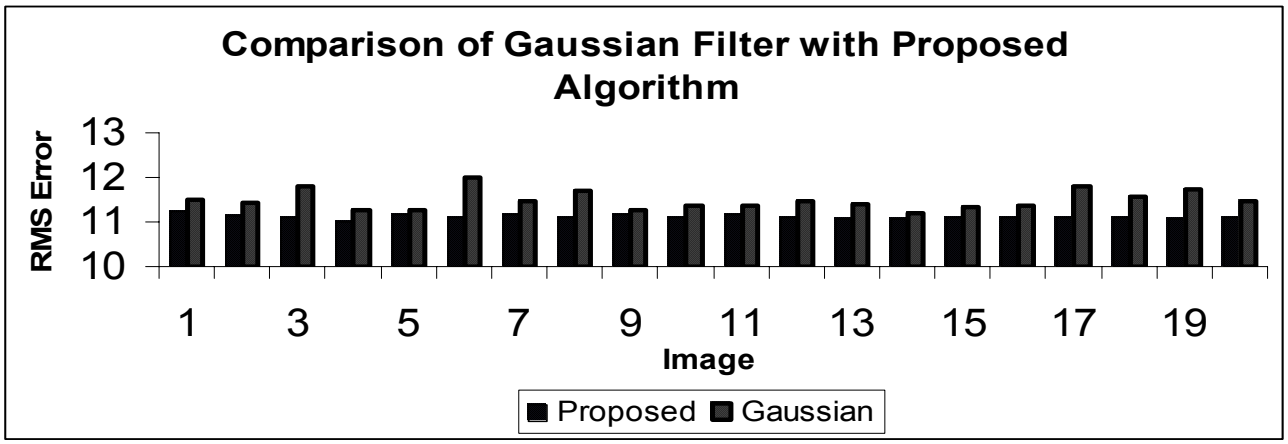


Chart 1: Shows comparison between Gaussian filter and Proposed algorithm

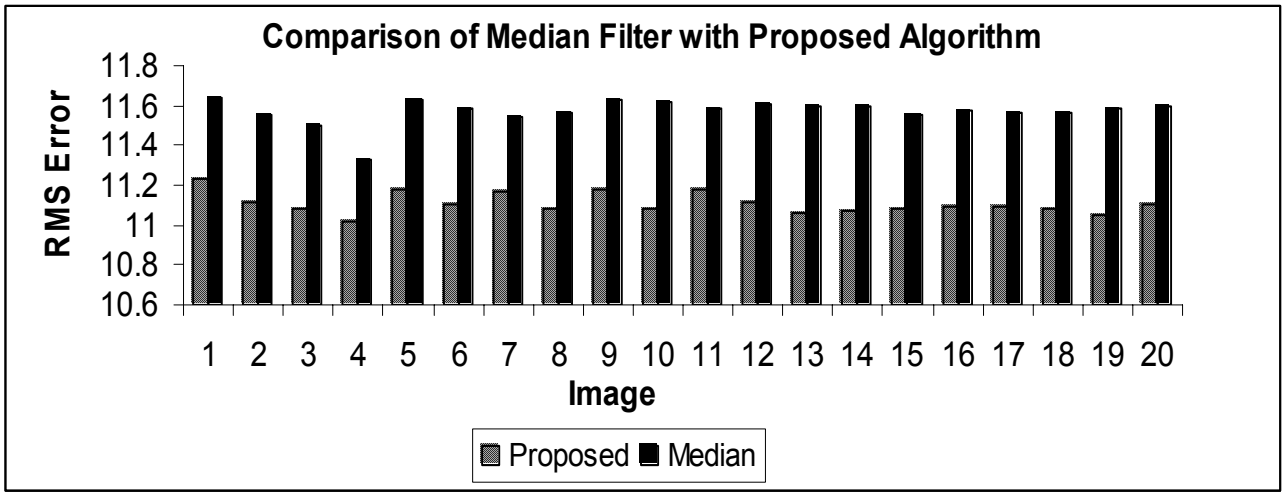


Chart 2: Shows comparison between Median filter and Proposed algorithm

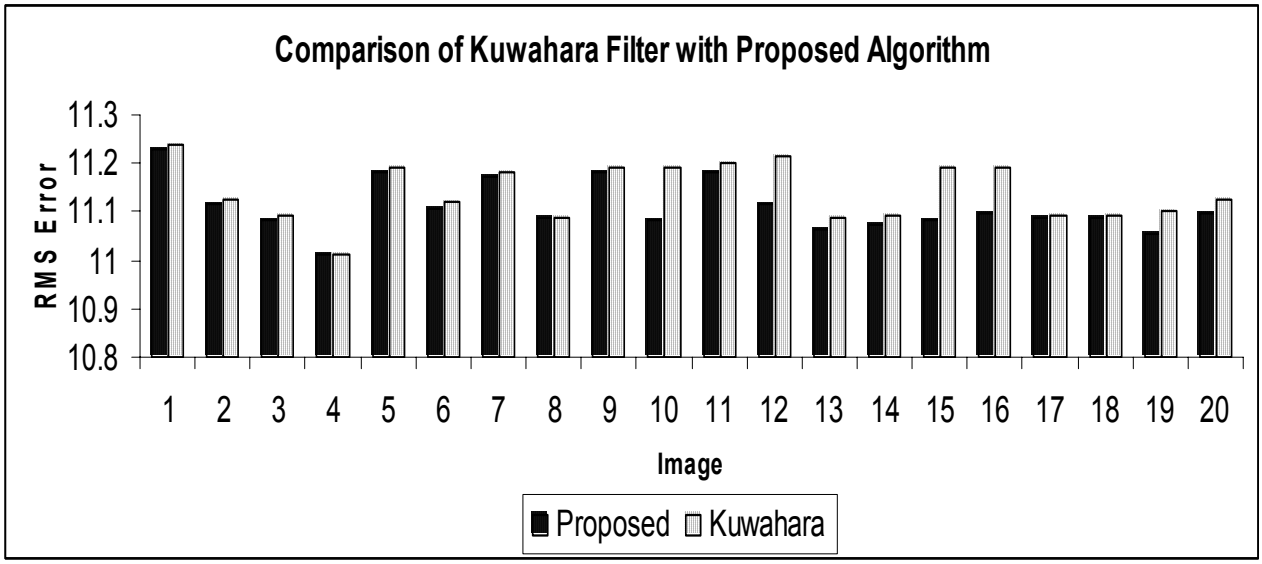


Chart 3: Shows comparison between Kuwahara filter and Proposed algorithm

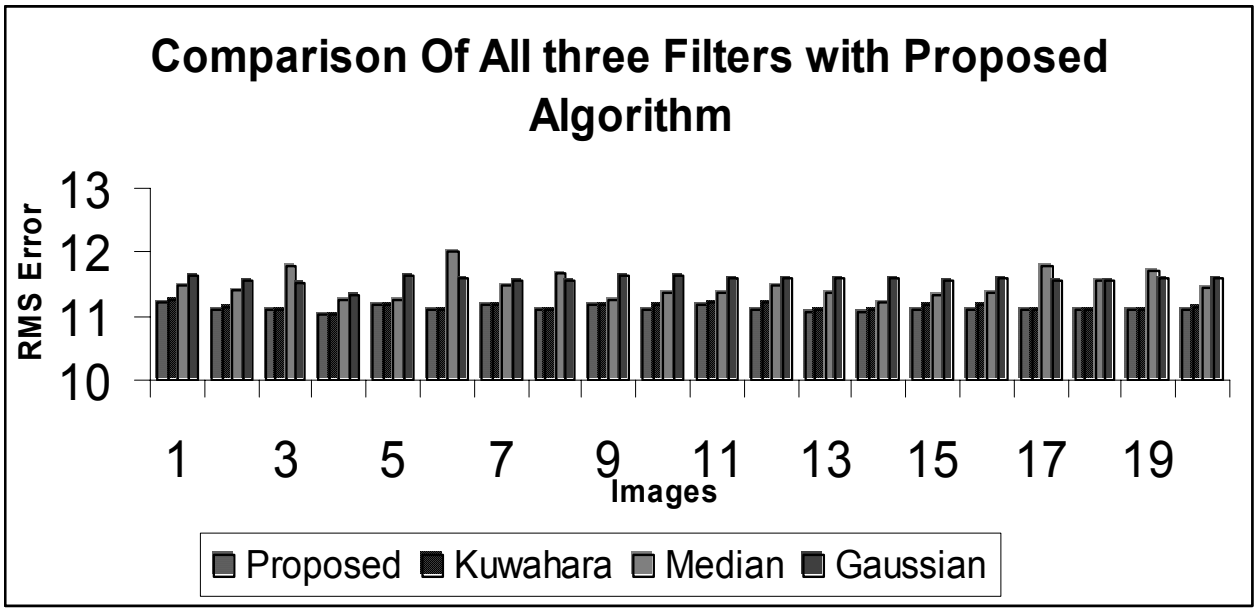


Chart 4: Shows comparison of Gaussian, Median and Kuwahara filters with Proposed Algorithms

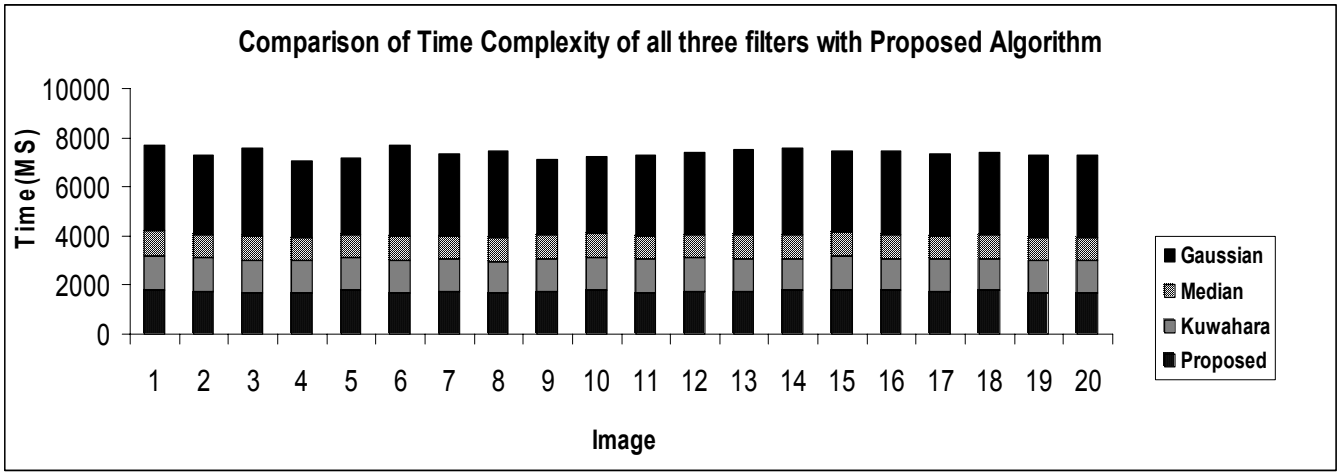


Chart 5:Shows comparison of time complexity among Median filter ,Gaussian filter. Kuwahara filter and Proposed Algorithm.

CHAPTER 6

1. LIMITATIONS

- It is spatial based algorithms.
- Image having large size and large intensity cannot filter by this algorithm.
- Complexity can't be reduced in higher level.

2. FUTURE ENHANCEMENT

- It can be used in image restoration process.
- Nature of impurity can be found before applying then, result can be better.
- Computational complexity (Execution time) is reduced a little bit.
- It is let to identify either the algorithm can be efficiently used as non-spatial domain
- It also remains to study toward removing the space complexity of the algorithm.
- Much more work is left to extend the algorithm to the large size image.

These works are left for furthere study.

3. CONCLUSION AND RECOMENDATION

In this dissertation different approaches for minimizing errors in an image are analyzed. Most of these algorithms are implemented. Finally, a new algorithm is purposed which modifies the kuwahara filter algorithm to remove noise from the images. Output image of this algorithm can be used as an input for other applications. It helps to remove noise in image but it does not loose image details.

CHAPTER 7

1.ANNEX 1

2.ANNEX 2

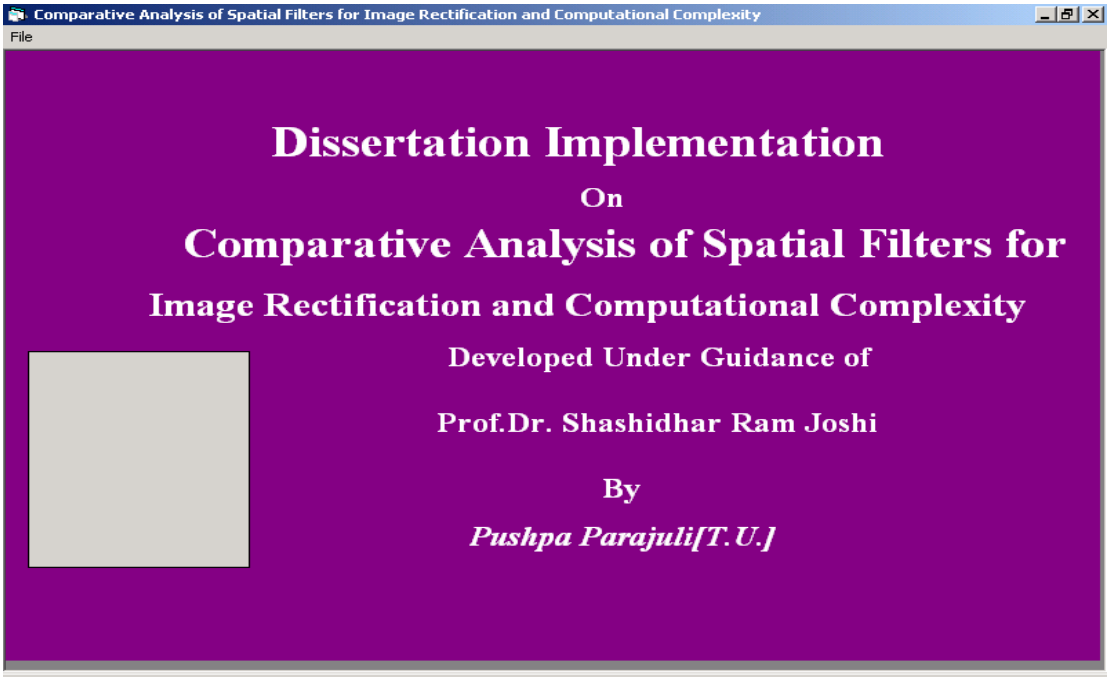
3.REFERENCES

4.BIBLIOGRAPHY

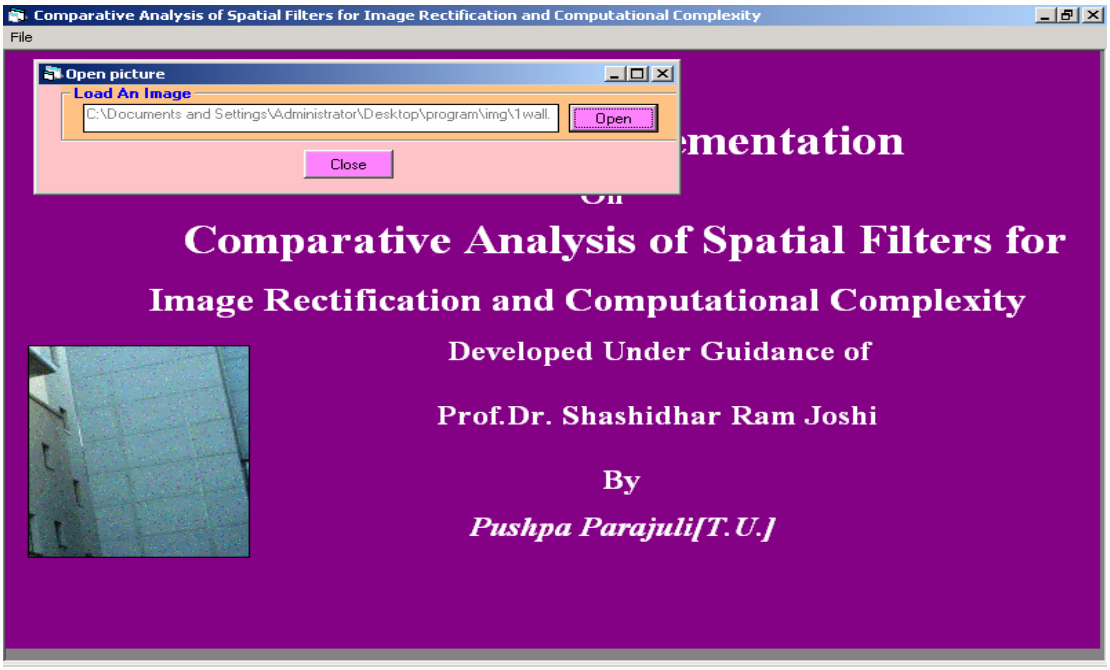
Annex: 1

Snapshots of Model Prototype

Snapshots of Model Prototypes

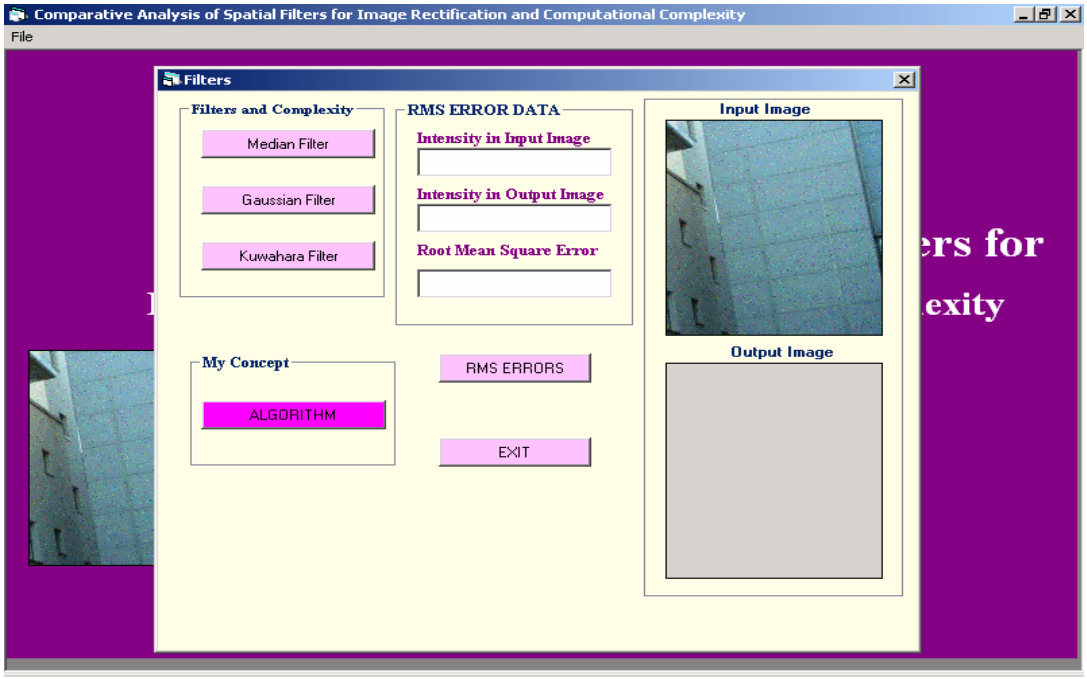


Initial Screen

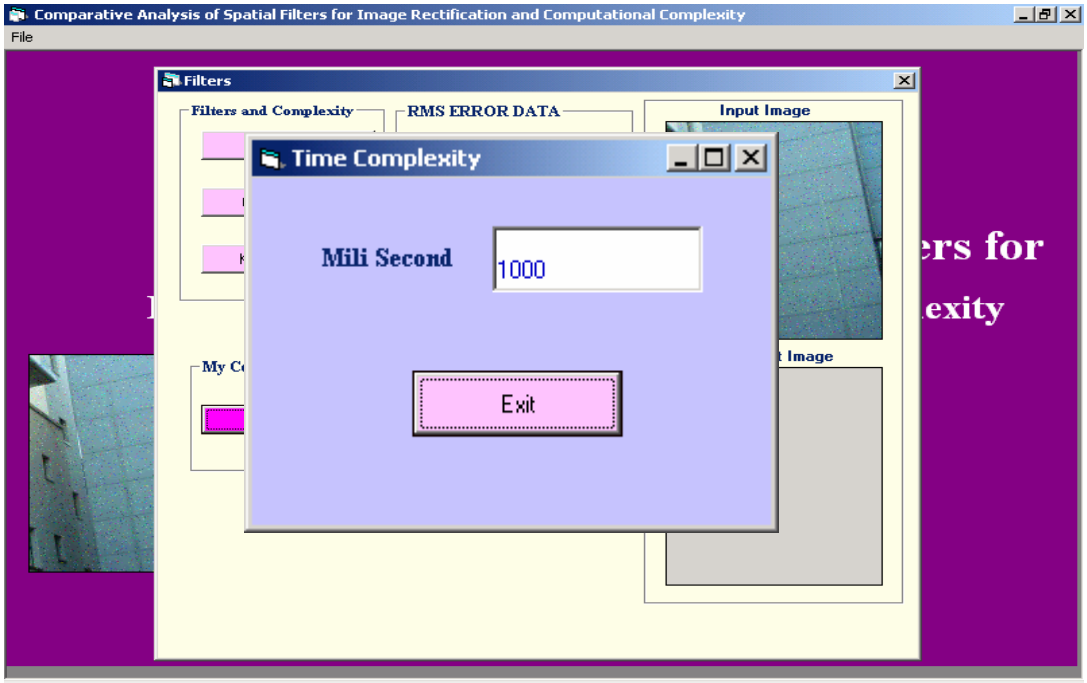


Screen after loading Image

Snapshots of Model Prototypes

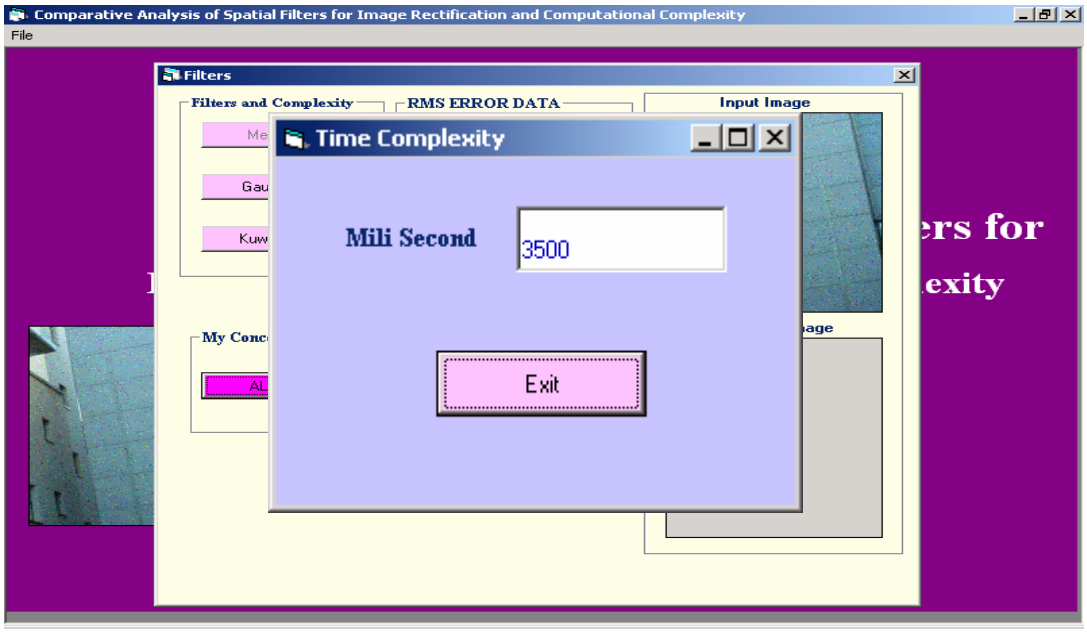


Screen when operation is ready

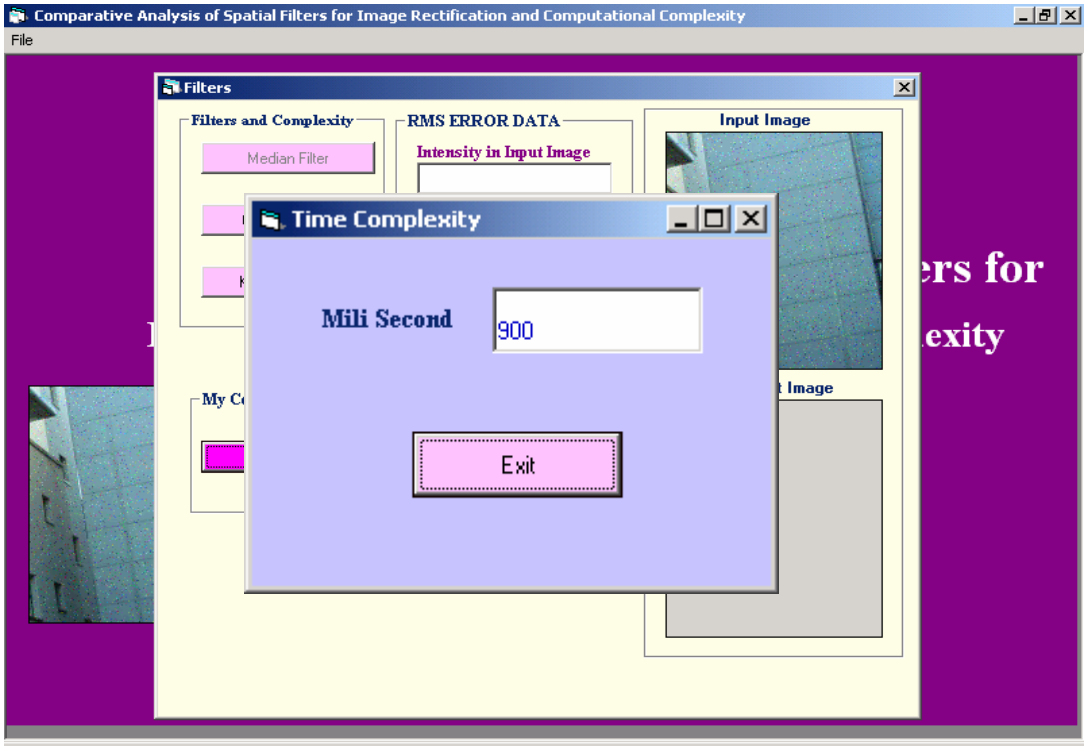


Time complexity of median filter

Snapshot of Model Prototype

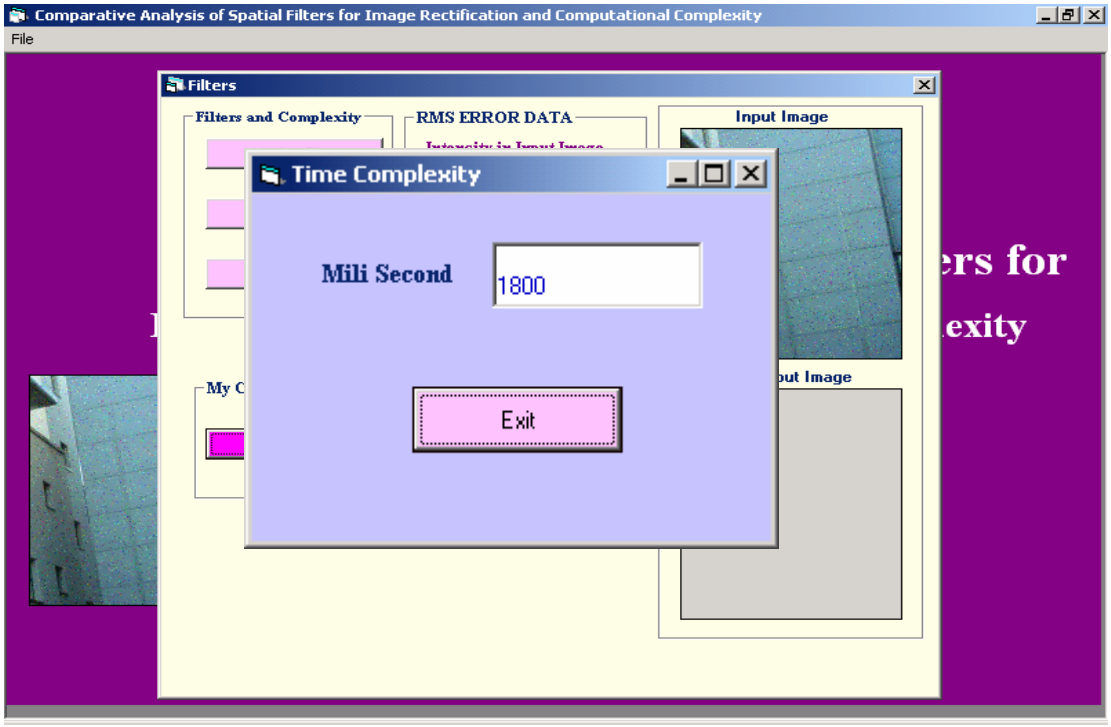


Time complexity of Gaussian filter

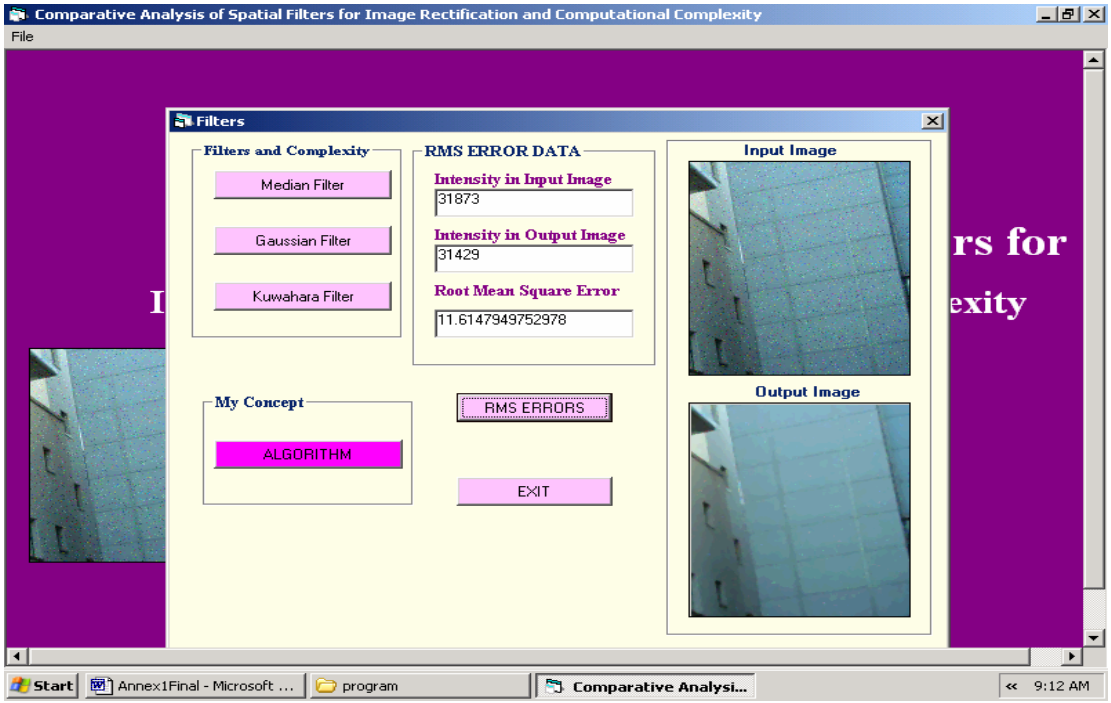


Time complexity of kuwahara filter

Snapshots of Model Prototypes

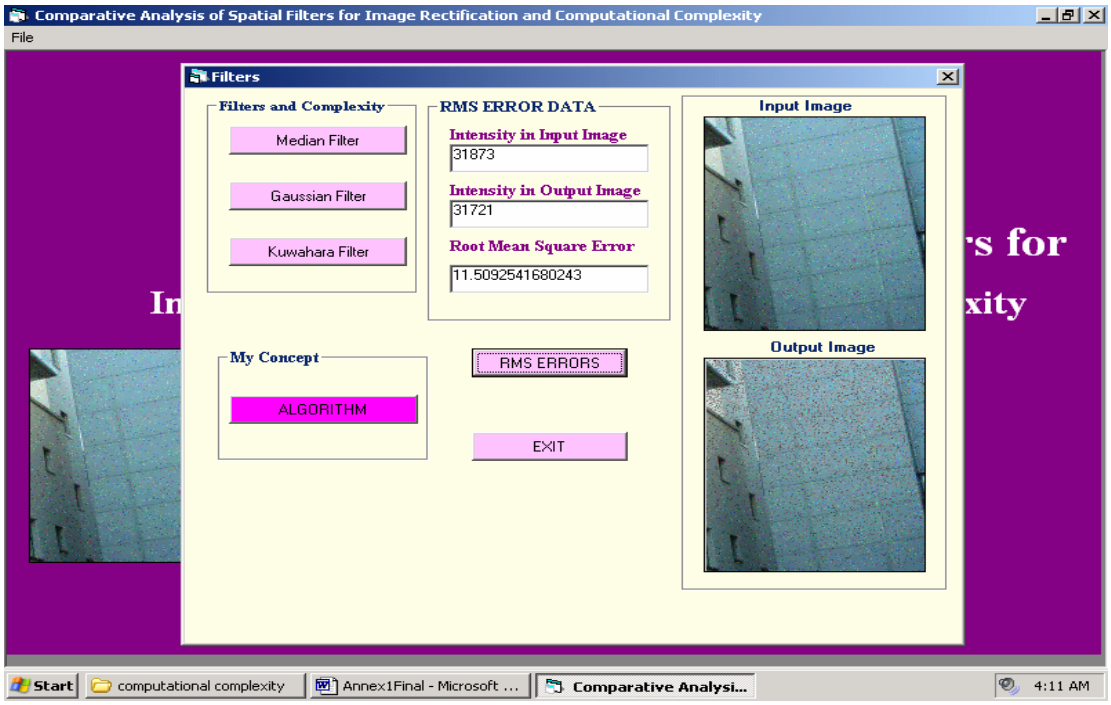


Time complexity of Proposed Algorithm

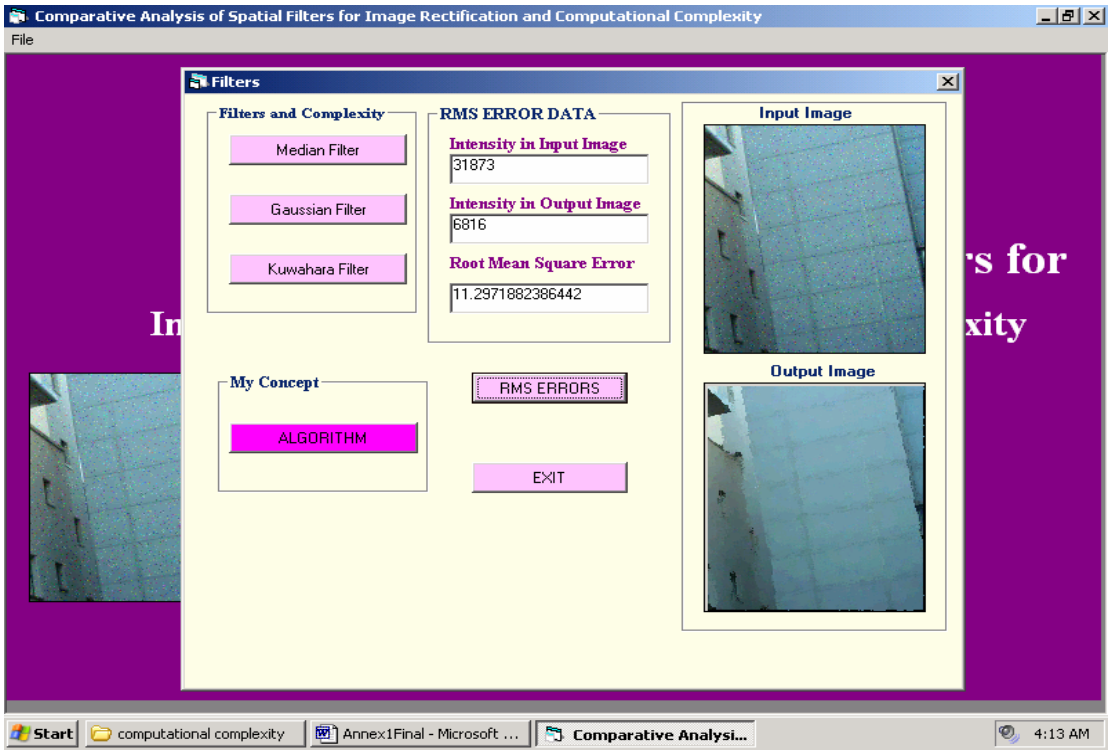


Output result of Median filter for image 1dwall.bmp

Snapshot of model Prototype

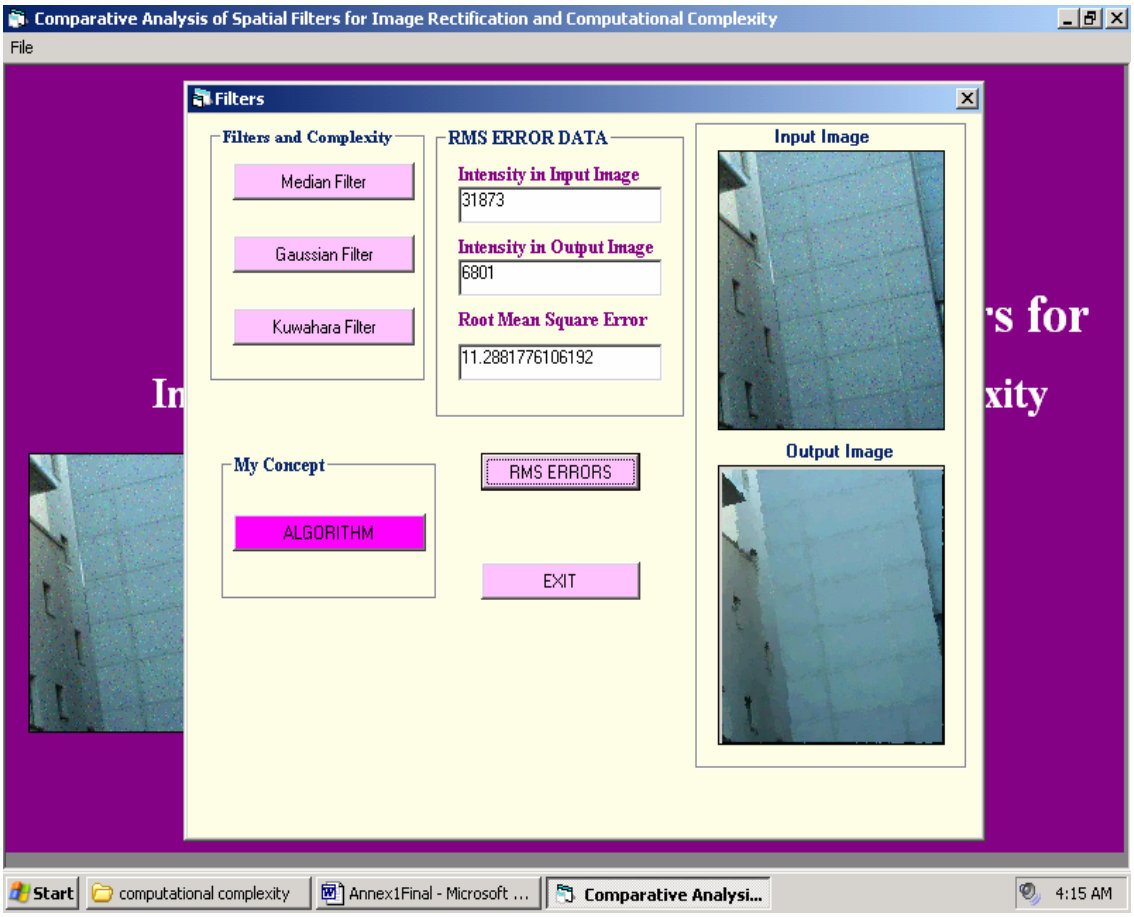


Output result of Gaussian filter for image 1dwall.bmp



Output result of Kuwahara Algorithm for image 1dwall.bmp

Snapshot of Model Prototype



Output result of Proposed Algorithm for image 1dwall.bmp

Annex: 2

Source Code

Source Code

```
'     Window
'A  A  A  A  A
'A  A  A  A  A
'A  A  X  A  A
'A  A  A  A  A
'A  A  A  A  A
Command9.Enabled = False
'Label5.Caption = "Logic : Based on region concept, Divides
segment into 4 regions,"
Picture2.Cls
Dim pt(6) As Double
Dim Min, Index, k As Integer
For I = 2 To Picture1.ScaleWidth
    For J = 2 To Picture1.ScaleHeight
    'region 1
        pt(0) = Picture1.Point(I, J + 1)
        pt(1) = Picture1.Point(I, J + 2)
        pt(2) = Picture1.Point(I - 1, J + 1)
        pt(3) = Picture1.Point(I - 1, J + 2)
        pt(4) = Picture1.Point(I - 2, J + 1)
        pt(5) = Picture1.Point(I - 2, J + 2)
        Call Region(pt, 0)
        'region 2
        pt(0) = Picture1.Point(I + 1, J)        '0
        pt(1) = Picture1.Point(I + 1, J + 1)    '0
        pt(2) = Picture1.Point(I + 1, J + 2)    '1
        pt(3) = Picture1.Point(I + 2, J)        '4
        pt(4) = Picture1.Point(I + 2, J + 1)    '4
        pt(5) = Picture1.Point(I + 2, J + 2)    '5
        Call Region(pt, 1)
        'region 3
        pt(0) = Picture1.Point(I - 1, J)        '0
        pt(1) = Picture1.Point(I - 1, J - 1)    '0
        pt(2) = Picture1.Point(I - 1, J - 2)    '1
        pt(3) = Picture1.Point(I - 2, J)        '4
        pt(4) = Picture1.Point(I - 2, J - 1)    '4
        pt(5) = Picture1.Point(I - 2, J - 2)    '5
        Call Region(pt, 2)
        'region 4
        pt(0) = Picture1.Point(I, J - 1)
        pt(1) = Picture1.Point(I, J - 2)
        pt(2) = Picture1.Point(I + 1, J - 1)
        pt(3) = Picture1.Point(I + 1, J - 2)
        pt(4) = Picture1.Point(I + 2, J - 1)
        pt(5) = Picture1.Point(I + 2, J - 2)
```

```
Call Region(pt, 3)
        DoEvents
        Min = Variance(0)
        For k = 0 To 3
            If Min >= Variance(k) Then
                Min = Variance(k)
                Index = k
            End If
        Next k
        Intensity = Means(Index)
        Picture2.PSet (I, J), Intensity
    Next J
Next I
```

Source code for Kuwahara Filter Implemention

Source code

```
Dim k1, l As Double: k1 = 0
k = 0
'Using Gaussian Smoothing
'smoothing the image before binarization using guassian filter
Dim wt As Integer
For I = 1 To Picture1.ScaleWidth
 For J = 1 To Picture1.ScaleHeight
  DoEvents
  k1 = Picture1.Point(I - 2, J - 2) + Picture1.Point(I - 2, J + 2) +
Picture1.Point(I + 2, J + 2) + Picture1.Point(I + 2, J - 2)
  k1 = k1 + 4 * (Picture1.Point(I - 2, J - 1) + Picture1.Point(I - 2, J +
1) + Picture1.Point(I + 2, J - 1) + Picture1.Point(I + 2, J + 1))
  k1 = k1 + 7 * (Picture1.Point(I, J - 2) + Picture1.Point(I - 2, J) +
Picture1.Point(I, J + 2) + Picture1.Point(I + 2, J))
  k1 = k1 + 16 * (Picture1.Point(I - 1, J - 1) + Picture1.Point(I - 1, J +
1) + Picture1.Point(I + 1, J + 1) + Picture1.Point(I + 1, J - 1))
  k1 = k1 + 26 * (Picture1.Point(I - 1, J) + Picture1.Point(I + 1, J) +
Picture1.Point(I, J - 1) + Picture1.Point(I, J + 1))
  k1 = k1 + 41 * (Picture1.Point(I, J))
  k1 = k1 / 273
  LngToRGB k1
  k1 = (R + G + b) / 3
  LngToRGB Picture1.Point(I, J)
  k = (R + G + b) / 3
  If k > k1 Then
     l = 0
     For p = -5 To 5
     For q = -5 To 5
      l = l + Picture1.Point(I + p, J + q)
     Next q
     Next p
     LngToRGB l
     l = (R + G + b) / 3
    If l < k1 Then
   Picture2.PSet (I, J), RGB(k1, k1, k1)
    Else
```

```
Picture2.PSet (I, J), RGB(k1, k1, k1)
    Else
      Picture2.PSet (I, J), Picture1.Point(I, J)
    End If
  Else
  Picture2.PSet (I, J), Picture1.Point(I, J)
  End If
  Next J
  Next I
```

Source code for Gaussain Filter Implemention

Source code

```
Dim aR(9), aG(9), aB(9) As Integer
Dim k As Long
Command7.Enabled = False
'Label5.Caption = "Logic : Take 8 Connected Pixels along with pixel
,Sort it, Take Median"
Picture2.Cls
For I = 1 To Picture1.ScaleWidth - 1
    For J = 1 To Picture1.ScaleHeight - 1
        k = Picture1.Point(I, J): LngToRGB k
        DoEvents
        aR(0) = R: aG(0) = G: aB(0) = b
        k = Picture1.Point(I + 1, J): LngToRGB k
        aR(1) = R: aG(1) = G: aB(1) = b
        k = Picture1.Point(I + 1, J + 1): LngToRGB k
        aR(2) = R: aG(2) = G: aB(2) = b
        k = Picture1.Point(I, J + 1): LngToRGB k
        aR(3) = R: aG(3) = G: aB(3) = b
        k = Picture1.Point(I - 1, J + 1): LngToRGB k
        aR(4) = R: aG(4) = G: aB(4) = b
        k = Picture1.Point(I - 1, J): LngToRGB k
        aR(5) = R: aG(5) = G: aB(5) = b
```

```
        k = Picture1.Point(I - 1, J - 1): LngToRGB k
        aR(6) = R: aG(6) = G: aB(6) = b
        k = Picture1.Point(I, J - 1): LngToRGB k
        aR(7) = R: aG(7) = G: aB(7) = b
        k = Picture1.Point(I + 1, J - 1): LngToRGB k
        aR(8) = R: aG(8) = G: aB(8) = b
        Call SortArry(aR, 8)
        Call SortArry(aG, 8)
        Call SortArry(aB, 8)
        Picture2.PSet (I, J), RGB(aR(5), aG(5), aB(5))
    Next J
Next I
```

Source code for median filter Implementation

Source Code

```
Dim pt(6) As Double
Dim cpt(8) As Double
Dim Min, Index, k As Integer
For I = 2 To Picture1.ScaleWidth
    For J = 2 To Picture1.ScaleHeight
    'region 1
        pt(0) = Picture1.Point(I, J + 1)
        pt(1) = Picture1.Point(I, J + 2)
        pt(2) = Picture1.Point(I - 1, J + 1)
        pt(3) = Picture1.Point(I - 1, J + 2)
        pt(4) = Picture1.Point(I - 2, J + 1)
        pt(5) = Picture1.Point(I - 2, J + 2)
        Call Region(pt, 0)
        'region 2
        pt(0) = Picture1.Point(I + 1, J)        '0
        pt(1) = Picture1.Point(I + 1, J + 1)    '0
        pt(2) = Picture1.Point(I + 1, J + 2)    '1
        pt(3) = Picture1.Point(I + 2, J)        '4
        pt(4) = Picture1.Point(I + 2, J + 1)    '4
        pt(5) = Picture1.Point(I + 2, J + 2)    '5
        Call Region(pt, 1)
        'region 3
```

```
pt(0) = Picture1.Point(I - 1, J)        '0
        pt(1) = Picture1.Point(I - 1, J - 1)    '0
        pt(2) = Picture1.Point(I - 1, J - 2)    '1
        pt(3) = Picture1.Point(I - 2, J)        '4
        pt(4) = Picture1.Point(I - 2, J - 1)    '4
        pt(5) = Picture1.Point(I - 2, J - 2)    '5
        Call Region(pt, 2)
        'region 4
        pt(0) = Picture1.Point(I, J - 1)
        pt(1) = Picture1.Point(I, J - 2)
        pt(2) = Picture1.Point(I + 1, J - 1)
        pt(3) = Picture1.Point(I + 1, J - 2)
        pt(4) = Picture1.Point(I + 2, J - 1)
        pt(5) = Picture1.Point(I + 2, J - 2)
        Call Region(pt, 3)
        DoEvents
        'region 5
cpt(0) = Picture1.Point(I - 1, J)
         cpt(1) = Picture1.Point(I + 1, J)
        cpt(2) = Picture1.Point(I + 1, J - 1)
        cpt(3) = Picture1.Point(I + 1, J + 1)
        cpt(4) = Picture1.Point(I - 1, J - 1)
        cpt(5) = Picture1.Point(I - 1, J + 1)
        cpt(6) = Picture1.Point(I, J - 1)
cpt(7) = Picture1.Point(I, J + 1)
        Call CRegion(cpt)
        DoEvents
        Min = CVariance
        For k = 0 To 3
            If Min >= Variance(k) Then
                Min = Variance(k)
                Index = k
            End If
        Next k
        If CVariance = Min Then
        Intensity = CMeans
        Else
        Intensity = Means(Index)
        End If
        Picture2.PSet (I, J), Intensity
    Next J
Next I
Command9.Enabled = True
```

Source code for Proposed Algorithm Implementation

Source code

```
Dim ImagePixels(2, 800, 800)
 Dim I As Integer, J As Integer, total As Long
 Dim sum As Long
  Dim red As Integer, green As Integer, blue As Integer
 Dim pixel As Long

 sum = 0
 total = 0
 For I = 1 To Picture1.ScaleWidth - 1
    For J = 1 To Picture1.ScaleHeight - 1
    pixel = Picture1.Point(I, J)
    red = pixel& Mod 256
    green = ((pixel And &HFF00) / 256&) Mod 256&
    blue = (pixel And &HFF0000) / 65536
    ImagePixels(0, I, J) = red
    ImagePixels(1, I, J) = green
    ImagePixels(2, I, J) = blue
    Next
    sum = red + green + blue / 3
    total = total + sum

     Next
    'total = sum + sum
```

Source Code for finding total intensity in input image

Source code

```
Dim ImagePixels(2, 800, 800)
 Dim I As Integer, J As Integer, total As Long
 Dim sum As Long


 Dim red As Integer, green As Integer, blue As Integer
 Dim pixel As Long

 sum = 0
 total1 = 0
 For I = 1 To Picture2.ScaleWidth - 1
    For J = 1 To Picture2.ScaleHeight - 1
    pixel = Picture2.Point(I, J)
    red = pixel& Mod 256
    green = ((pixel And &HFF00) / 256&) Mod 256&
    blue = (pixel And &HFF0000) / 65536
    ImagePixels(0, I, J) = red
    ImagePixels(1, I, J) = green
    ImagePixels(2, I, J) = blue
    Next
    sum = red + green + blue / 3
    total1 = total1 + sum

    Next
    'total = sum + sum


  Label4.Caption = total1
```

Source code for finding total intensity in output image.

Source code

```
Screen.MousePointer = vbHourglass
Dim ImagePixels(2, 800, 800)
Dim ImagePixels1(2, 800, 800)
 Dim I, k, l As Integer, J As Integer, total As Long
 Dim sum, sum1 As Long

  Dim red As Integer, green As Integer, blue As Integer
 Dim pixel As Long
 Dim diff As Double
 sum = 0
 sum1 = 0
 total = 0
 For I = 1 To Picture1.Width
   For J = 1 To Picture1.Height
    pixel = Picture1.Point(I, J)
    red = pixel& Mod 256
    green = ((pixel And &HFF00) / 256&) Mod 256&
    blue = (pixel And &HFF0000) / 65536
    sum1 = red + green + blue / 3
    pixel = Picture2.Point(I, J)
    red = pixel& Mod 256
    green = ((pixel And &HFF00) / 256&) Mod 256&
    blue = (pixel And &HFF0000) / 65536
    sum2 = red + green + blue / 3
    diff = sum1 - sum2
    diff = diff * diff
    total = total + diff
     Next
  Next
    Label11.Caption = CStr(Abs(Log(((Sqr(total / Picture1.Height
* Picture1.Width)) ^ 0.5) / (Picture1.Width *
Picture1.Height))))
    Screen.MousePointer = vbNormal
```

Source code for finding RMS error

Source Code

```
Public totalInterval As Double
Dim T1 As Double
Sub StartCounting()
T1 = Time

End Sub
Sub StopCounting()
totalInterval = totalInterval + Time - T1

End Sub
Sub ResetTimer()
totalInterval = 0

End Sub
```

Source Code for Time complexity

Source code

```
 Public b As Long
Public G As Long
Public R As Long
Public pict As PictureBox
Public CFRM As Form
Public Means(4) As Double
Public Variance(4) As Integer
Public CVariance As Integer
Public CMeans As Double
Public Sub AlignFrm()
CFRM.Top = (Screen.Height - CFRM.Height) / 2
CFRM.Left = (Screen.Width - CFRM.Width) / 2
End Sub
Public Function SortArry(DesArr, arrLen)
Dim k, M, temp As Integer
For k = 0 To arrLen
    For M = 0 To arrLen - 1 - k
    If (DesArr(M) > DesArr(M + 1)) Then
        temp = DesArr(M)
        DesArr(M) = DesArr(M + 1)
        DesArr(M + 1) = temp
    End If
    Next M
Next k
End Function
Public Function CRegion(pt() As Double)
    Dim aR(8) As Double, aG(8) As Double, aB(8) As Double
    Dim redM As Double, greenM As Double, blueM As Double
    Dim redV As Double, greenV As Double, blueV As Double
    Dim k As Integer

    LngToRGB (pt(0))
    aR(0) = R: aG(0) = G: aB(0) = b
    LngToRGB (pt(1))
    aR(1) = R: aG(1) = G: aB(1) = b
    LngToRGB (pt(2))
    aR(2) = R: aG(2) = G: aB(2) = b
    LngToRGB (pt(3))
    aR(3) = R: aG(3) = G: aB(3) = b
    LngToRGB (pt(4))
```

```
    aR(4) = R: aG(4) = G: aB(4) = b
    LngToRGB (pt(5))
    aR(5) = R: aG(5) = G: aB(5) = b
    LngToRGB (pt(6))
    aR(6) = R: aG(6) = G: aB(6) = b
    LngToRGB (pt(7))
    aR(7) = R: aG(7) = G: aB(7) = b
    redM = greenM = blueM = redV = greenV = blueV = 0
    For k = 0 To 7
        redM = redM + aR(k)
        greenM = greenM + aG(k)
        blueM = blueM + aB(k)
    Next k
    redM = redM / 8
    greenM = greenM / 8
    blueM = blueM / 8

    For k = 0 To 8
        redV = redV + (aR(k) - redM) ^ 2
        greenV = greenV + (aG(k) - greenM) ^ 2
        blueV = blueV + (aB(k) - blueM) ^ 2
    Next k
    redV = redV / 8
    greenV = greenV / 8
    blueV = blueV / 8

    CMeans = RGB(Abs(redM), Abs(greenM), Abs(blueM))
    CVariance = (redV + greenV + blueV) / 20
End Function
Public Function Region(pt() As Double, reg As Integer)
    Dim aR(6) As Double, aG(6) As Double, aB(6) As Double
    Dim redM As Double, greenM As Double, blueM As Double
    Dim redV As Double, greenV As Double, blueV As Double
    Dim k As Integer

    LngToRGB (pt(0))
    aR(0) = R: aG(0) = G: aB(0) = b
    LngToRGB (pt(1))
    aR(1) = R: aG(1) = G: aB(1) = b
    LngToRGB (pt(2))
    aR(2) = R: aG(2) = G: aB(2) = b
```

```
LngToRGB (pt(3))
    aR(3) = R: aG(3) = G: aB(3) = b
    LngToRGB (pt(4))
    aR(4) = R: aG(4) = G: aB(4) = b
    LngToRGB (pt(5))
    aR(5) = R: aG(5) = G: aB(5) = b

    redM = greenM = blueM = redV = greenV = blueV = 0
    For k = 0 To 5
        redM = redM + aR(k)
        greenM = greenM + aG(k)
        blueM = blueM + aB(k)
    Next k
    redM = redM / 6
    greenM = greenM / 6
    blueM = blueM / 6

    For k = 0 To 5
        redV = redV + (aR(k) - redM) ^ 2
        greenV = greenV + (aG(k) - greenM) ^ 2
        blueV = blueV + (aB(k) - blueM) ^ 2
    Next k
    redV = redV / 6
    greenV = greenV / 6
    blueV = blueV / 6

    Means(reg) = RGB(Abs(redM), Abs(greenM), Abs(blueM))
    Variance(reg) = (redV + greenV + blueV) / 3
End Function


Public Sub LngToRGB(ByVal longColor As Long)
    Dim lngColor As Long
    lngColor = longColor
    R = lngColor Mod &H100
    lngColor = lngColor \ &H100
    G = lngColor Mod &H100
    lngColor = lngColor \ &H100
    b = lngColor Mod &H100

End Sub
```

Source code for different modules

References:

[1] Banham, M. R. and Katsaggelos.” Spatially Adaptive Wavelet Based Multiscale Image Restoration”,Proc. of IEEE,86(4), pp. 615-638[1993]

[2] P. Bakker, L.J. van Vliet, P.W.” Edge preserving orientation adaptive filtering”,IEEE journals on spatial filter,43(3),pp. 157-193, Verbeek Pattern Recognition Group, Department of Applied Physics, Delft University of Technology, The Netherlands[2003]

[3] Mark J. T. Smith, and Russell M. Mersereau.” Improved Structures of Maximally Decimated Directional Filter Banks for Spatial Image Analysis”, [IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 13, NO. 11, NOVEMBER 2004]

[4] Shlomo Greenberg, Mayer Aladjem, Daniel Kogan and Itshak Dimitrov.” Fingerprint Image Enhancement using Filtering Techniques”, Electrical and Computer Engineering Department, Ben-Gurion University of the Negev, Beer-Sheva, Israel[2005]

[5] Oya Y. Rieger.” Preservation in the Age of Large Scale Digitization”, [2008]

[6] L. Fortnow and Steve Homer.” A Short History of Computational Complexity”, [2002/2003].

[7] Briana Wandell and Abbaselgamal.” Common Principles of Image Acquisition”, ANDBERNDGIROD,FELLOW,IEEE[2004]

[8] Rafael C. Gonzalez and Richard E. Woods.”Digital Image Processing”,Second Edition

[9] Jan van Leeuwen.” Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity, The MIT Press/Elsevier”, [1990]

[10] Hanspeter Pfister and Leonard McMillan.“Acquisition and Rendering of Transparent and Refractive Objects”, [2002]

[11] Claude Debussy.” Blurring strategies for hyperstack image segmentation”, [2001]

[12] Thomas Porter. “Compositing Digital Images”, [1984]

[13] Shawn Chen and Tian-Yuan Shih. “On the evaluation of edge preserving smoothing filter “,Department of Civil Engineering National Chiao-Tung University Hsin-Chu, Taiwan.[2001]

[14] Yasuyuki Sugaya, Kenichi Kanatani and Yasushi Kanazawa.“Generating Dense Point Matches Using Epipolar Geometry”, Department of Computer Science, Okayama University ,Okayama 700-8530 Japan[1993]

[15] Sing Bing Kang. “A Survey of Image-based Rendering Techniques”, [1997]

[16] D. Lee, I. Kweon, and R. Cipolla. “A biprism stereo camera system”, In Proceedings of the 1999, Conference on Computer Vision and Pattern Recognition, [1999.]

[17] Richard I. Hartley.” Computation of the essential matrix from Points”, GE-CRD, Schenectady, New York [1989]

[18] Du Huynh.” A short tutorial on image rectification”, December[2003]

Bibliography:

1. Mastering Visual Basic 6, Second Edition, Evangelos Petroustsos

2. Digital Image Processing, Second Edition, Rafael C. Gonzalez, Richard E. Woods

3. Introduction to Algorithms, Second Edition, Thomash H. Coremen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein

4. Fundamentals of Digital Image Processing, First Edition, Kenneth R. Castleman