# IMAGE MATCHING AND RETRIEVAL SYSTEM USING OBJECT SHAPE AND STRUCTURE


**SYNDICATE MEMBERS**

CAPT SHAHID ANWER (LDR)
CAPT NASIR MEHMOOD
CAPT ALI AURANGZEB
CAPT AHMED KHIZAR

**PROJECT SUPERVISORS**

LT COL NAVEED SARFRAZ KHAN KHATTAK
MAJ(R) FAZAL AHMED



**Dissertation submitted for partial fulfillment of requirement of MCS/NUST for the award of the B.E. degree in Telecommunication Engineering**



**DEPARTMENT OF ELECTRICAL ENGINEERING**
**MILITARY COLLEGE OF SIGNALS**
**NATIONAL UNIVERSITY OF SCIENCE & TECHNOLOGY**
**RAWALPINDI**



**APRIL 2004**


**ABSTRACT**

# OBJECT SHAPE AND STRUCTURE FOR IMAGE MATCHING AND RETRIEVAL

This project works on the use of shape features for content-based image retrieval. Edges, lines, corners, and straight ribbons have been extracted from a query image. These features and their attributes, along with various relations are than used to match representations of other images in a database.

Our project is mainly based upon the work done by Naveed Sarfraz Khan Khattak and is the continuity of his area of research. He developed his algorithms in MATLAB. We continued upon his work by improving the edge detection and line detection techniques. The better line detection has led to better corner and ribbon detection thus improving our work tremendously. As the project has been implemented in Visual C++ so we have the added advantage of fast processing.

# ACKNOWLEDGMENTS

We want to thank National University of Science and Technology which provided us the opportunity to gain knowledge and to abreast ourselves with the latest trends in the Telecom Sciences. We are thankful to all the faculty members of EE DEPT for their kind help in studies.

We would also like to express our appreciation and special credit to our supervisor Naveed Sarfraz Khan Khattack, whose shear interest led us to this great achievement.

At the same time we also dedicate our whole work to our parents and family members who prayed for our success.

# TABLE OF CONTENTS

# LIST OF TABLES

## LIST OF FIGURES

# 3 Introduction

With the explosive advancement in imaging technologies, image retrieval has attracted the increasing interests of researchers in the fields of image processing, and database systems. The traditional indexing for image retrieval is text-based. Although text annotation is a practical technique, however, this task is labour intensive, language dependent, vocabulary controlled, and human subjective in nature. In some cases, it is rather difficult to characterise certain important real world concepts, entities, and attributes by means of text only. The shape of a single object and the various spatial constraints among multiple objects in an image are examples of such concepts. Motivated by the ultimate goal of automatically computing efficient and effective descriptors which symbolize various properties of images, recent research on image retrieval systems has been directed towards the development of content-based retrieval techniques for the management of visual information such as colour, texture, shape and spatial contents.

An important problem in the field of image processing is the automatic recognition of objects in two-dimensional images for the classification of scenes. Another problem is that of matching images based upon similarities between object features in a database. The database may contain hand-drawn images, images taken from cameras and photographs containing objects with varying geometric properties, colours, and textures, as well as a wide range in image resolution and size. The extraction and subsequent classification of object features must be general enough to deal with the varied contents of the database and must ensure that reliable matching takes place. At the same time, the system should strive for automation so that user interaction is kept to a minimum. Image retrieval can be categorized into exact match searching and similarity based searching. For either type of retrieval, the dynamic aspects of image content require expensive computations and sophisticated methodologies in the areas of image processing and database systems. In order to overcome these problems, several schemes for data modeling and image representation have been proposed.

In this project we will also cover the work done by Naveed Sarfraz Khan Khattak whose main area of concern was to concentrate on extracting shape structures with the intent of developing better representations of scenes and better means of matching in the domain of scenes with significant man-made object content. With a view towards the recognition of man-made objects in scenes, the natural features to use as the basis of the matching algorithms are lines, corners, and ribbons. His retrieval process is divided into three sub processes; the feature extraction process, data representation, and graph matching algorithm.

For edge detection techniques we have studied different papers and in subsequent chapters we will give an overview of some literature and in them the operators being used by researchers. Generally in feature extraction our main area of study was the canny edge detector and Hough Transform. Although these both algorithms are very active in extracting the features accurately such as edges and lines but with some observations which will be discussed in subsequent chapters. The methodology defined by Naveed over come the problems being faced in feature extraction. He used some extra feature like ribbons and corners to achieve better accuracy for image retrieval. These extracted features are then processed for extraction of relative attributes, for example distance, angle, and orientation, for matching purposes. The whole work is marvellous but it requires a lot of computational time and our main focus will be to improve upon his existing work in MATLAB and to minimize the time required in image retrieval and to improve its processing speed by converting it into VC.

The core work of our study is presented in Chapter 2 to Chapter 6. Chapter 2 provides an overview of the image processing. Then detailed description of the techniques used for edge, line, corner, and ribbon detection will be discussed in Chapter 3, 4 and 5 with the results of these detectors applied to both hand-drawn images and manmade object. Chapter 6 is devoted to graph data structure development and the graph matching algorithm. Chapter 7 is about GUI of Software and Chapter 8 summarizes the shortcoming and problems faced in different scenarios.

# 2. Literature Review

## 2.1 CBIR

The earliest use of the term *content-based image retrieval* in the literature seems to have described experiments into automatic retrieval of images from a database by color and shape feature. The term has since been widely used to describe the process of retrieving desired images from a large collection on the basis of features (such as color, texture and shape) that can be automatically extracted from the images themselves. The features used for retrieval can be either primitive or semantic, but the extraction process must be predominantly automatic.

CBIR differs from classical information retrieval in that image databases are essentially unstructured, since digitized images consist purely of arrays of pixel intensities, with no inherent meaning. One of the key issues with any kind of image processing is the need to extract useful information from the raw data (such as recognizing the presence of particular shapes or textures) before any kind of reasoning about the image's contents is possible. CBIR draws many of its methods from the field of image processing and computer vision, and is regarded by some as a subset of that field. It differs from these fields principally through its emphasis on the retrieval of images with desired characteristics from a collection of significant size.

## 2.2 Previous reviews

Several reviews of the literature on image retrieval have been studied, from a variety of different viewpoints.

Aigrain [4] discuss the main principles of automatic image similarity matching for database retrieval, emphasizing the difficulty of expressing this in terms of automatically generated features. They review a selection of current techniques for both still image retrieval and video data management, including video parsing, shot detection, key frame extraction and video skimming. They conclude that the field is expanding rapidly, but that many major research challenges remain, including the difficulty of expressing semantic information in terms of primitive image features, and the need for significantly improved user interfaces.

Eakins [5] proposes a framework for image retrieval classifying image queries into a series of levels, and discussing the extent to which advances in technology are likely to meet users' needs at each level. His conclusion is that automatic CBIR techniques can already address many of users' requirements at level 1, and will be capable of making a significant contribution at level 2 if current research ideas can be successfully exploited. They are however most unlikely to make any impact at level 3 in the foreseeable future.

Idris and Panchanathan [6] provide an in-depth review of CBIR technology, explaining the principles behind techniques for color, texture, shape and spatial indexing and retrieval in some detail. They identify a number of key unanswered research questions, including the development of more robust and compact image content features, more accurate modeling of human perceptions of image similarity, the identification of more efficient physical storage and indexing techniques, and the development of methods of recognizing objects within images.

### 2.3 Current level 1 CBIR techniques

In contrast to the text-based approach, CBIR operates on a totally different principle, retrieving stored images from a collection by comparing features automatically extracted from the images themselves. The commonest features used are mathematical measures of colour, texture or shape; hence virtually all current CBIR systems, whether commercial or experimental, operate at level 1. A typical system allows users to formulate queries by submitting an example of the type of image being sought, though some offer alternatives such as selection from a palette or sketch input. The system then identifies those stored images whose feature values match those of the query most closely, and displays thumbnails of these images on the screen. Some of the more commonly used types of feature used for image retrieval are described below.

### 2.3.1 Colour Based Retrieval

Color is one of the most widely used features in content-based image retrieval systems. The color feature is relatively robust to background complications, image resolution and orientation. Color histogram is most commonly used to extract color features each image added to the collection is analyzed to compute a color histogram, which shows the proportion of pixels of each color within the image. The color histogram for each image is then stored in the database. At search time, the user can either specify the desired proportion of each color (75% olive green and 25% red, for example), or submit an example image from which a color histogram is calculated. Either way, the matching process then retrieves those images whose color histograms match those of the query most closely. The matching technique most commonly used, histogram intersection, was first developed by Swain and Ballard [7].

### 2.3.2 Texture Based Retrieval

A variety of techniques have been used for measuring texture similarity, fundamentally, these calculate the relative brightness of selected pairs of pixels from each image. From these Tamura [8], calculated measures of image texture such as the degree of *contrast, coarseness, directionality and regularity*. Texture queries can be made in a manner similar to color queries, by supplying an example query image. The system then retrieves images with texture measures most similar in value to the query.

### 2.3.3 Shape Based Retrieval

Shape is an important feature in content-based image retrieval. However, shape is not a well-defined concept mathematically. Because of uncertainty in shape representations, the retrieval system may work well only in certain environments. Shapes can be represented either globally such as with aspect ratio, perimeters and moments, or locally such as with boundary segments. Shape matching of three-dimensional objects is a more challenging task, particularly where only a single 2-D view of the object in question is available. While no general solution to this problem is known, some useful inroads have been made into the problem of identifying at least some instances of a given object from different viewpoints.

### 2.3.4 Retrieval by other Types of Primitive Feature

One of the oldest established means of accessing pictorial data is retrieval by its position within an image. Accessing data by spatial location is an essential aspect of geographical information systems and efficient methods to achieve this have been around for many years. Several other types of image feature have been proposed as a basis for CBIR. Most of these depend on complex transformations of pixel intensities. These techniques aim to extract features, which reflect some part of image similarity. The most well known technique of this type uses the wavelet transform to model an image at several different resolutions. Promising retrieval results have been reported by matching wavelet features computed from query and stored images.

The advantage of all these techniques is that they can describe an image at varying levels of detail (useful in natural scenes where the objects of interest may appear in a variety of appearances), and avoid the need to segment the image into regions of interest before shape descriptors can be computed.

Object recognition has also been an area of interest to the computer vision community for many years. Techniques are now being developed for recognizing and classifying objects with database retrieval in mind. All such techniques are based on the idea of developing a model of each class of object to be recognized, identifying image regions which might contain examples of the object, and building up evidence to confirm or rule out the object's presence. Evidence will typically include both features of the candidate region itself (color, shape or texture) and contextual information such as its position and the type of background in the image.

In contrast to these fully automatic methods is a family of techniques, which allow systems to learn associations between semantic concepts and primitive features from user feedback. The earliest such system was Four Eyes from Minka. This encourages the user to explain selected regions of an image, and then proceeds to apply similar semantic labels to areas with similar characteristics. The system is capable of improving its performance with user feedback.

## 2.4 Retrieval by semantic image feature

### 2.4.1 Level 2

The vast majority of current CBIR techniques are designed for primitive-level retrieval. However, some researchers have attempted to bridge the gap between level 1 and level 2 retrieval. More recent research has tended to concentrate on one of two problems. The first is scene recognition. It can often be important to identify the overall type scene depicted by an image, both because this in an important filter which can be used when searching, and because this can help in identifying specific objects present. One system of this type uses colour, texture, region and spatial information to derive the most likely interpretation of the scene, generating text descriptors which can be input to any text retrieval system. Other researchers have identified simpler techniques for scene analysis, using low-frequency image components to train a neural network or colour neighborhood information extracted from low-resolution images to construct user-defined templates.

The second focus of research activity is object recognition, an area of interest to the computer vision community for many years. Techniques are now being developed for recognizing and classifying objects with database retrieval in mind. The best-known work in this field is probably that of Forsyth [9], who have attracted considerable publicity for themselves by developing a technique for recognizing naked human beings within images, though their approach has been applied to a much wider range of objects, including horses and trees.

In contrast to these fully-automatic methods is a family of techniques which allow systems to learn associations between semantic concepts and primitive features from user feedback. This invites the user to annotate selected regions of an image, and then proceeds to apply similar semantic labels to areas with similar characteristics. The system is capable of improving its performance with further user feedback.

### 2.4.2 Level 3

Reports of automatic image retrieval at level 3 are very rare. The only research that falls even remotely into this category has attempted to use the subjective connotations of colour (such as whether a colour is perceived to be warm or cold, or whether two colours go well with each other) to allow retrieval of images evoking a particular mood . It is not at present clear how successful this approach will prove.

## 2.5 Practical Applications of CBIR

A wide range of possible applications for CBIR technology has been identified e.g.

- Crime prevention
- The military
- Architectural and engineering design
- Medical diagnosis
- Geographical information and remote sensing systems
- Education and training
- Web searching
- Robotics
- Industries

# 3. Edge Detection

## 3.1 Edge Definitions

Shapiro and Stockman [1] define Edge to be," A set of connected pixels that lie on the boundary between two regions"

At its simplest, an edge is a sharp discontinuity in gray-level profile. However, the situation is complicated by presence of noise and image resolution. An edge is specified by its magnitude and its direction. A number of linked edge points may be better approximated by a linear segment called an edge. There are many types of edges; they may be classified into three classes: step edge, roof edge and linear edge.

## 3.2 Related Works

To detect these different types of edges many edge operators have been suggested in our project i.e. Roberts, Sobel, Prewitt.

### 3.2.1 Roberts cross-gradient operator

It is one of the simplest ways to implement a first order partial derivative. If it is to be applied on Z5 in figure-1following diagram then equation will be,

Gx = (Z9-Z5)……...…….. 3.1

And

Gy = (Z8-Z6)……...…..... 3.2

| | | |
|---|---|---|
| Z1 | Z2 | Z3 |
| Z4 | Z5 | Z6 |
| Z7 | Z8 | Z9 |

Figure-1(a)

| | |
|---|---|
| -1 | 0 |
| 0 | 1 |

(c)

| | |
|---|---|
| 0 | 1 |
| - 1 | 0 |

(b)

The only drawback is that Roberts operator is of size 2*2 so it is awkward to use as it does not have a clear centre value. It gives distorted and thick lines. It is less sensitive to noise. It has an advantage of less processing. It works well with the geometric shapes but not with manmade objects and buildings. Some lines are missed out when edges are detected.

### 3.2.2 Prewitt operator

In this operator mask size is of 3*3 and in its formulation the difference between the first and the third row of the 3*3 image region approximate the derivative in the X-direction and the difference between the third and the first column approximate the derivative in Y-direction, it is explained in Figure-2 (a),(b) and expressed in equation 3.3 and 3.4.Its performance is better than the Roberts cross operator. It has bigger mask which is better in noisy conditions. The distortion in lines is also less. It is not much effective for man made objects. Some lines are missed out.

| | | |
|---|---|---|
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| 1 | 1 | 1 |

Figure-2(a)

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |

(b)

Gx= (Z7+Z8+Z9)–(Z1+Z2+Z3)……...…….. 3.3

Gy=(Z3+Z6+Z9)– (Z1+Z4+Z7)…… 3.4

### 3.2.3 Sobel operator

For further smoothing the edge detection a weight value of 2 is used to give more importance at the centre. It is mostly used for computing digital gradient but hard to implement as compared to Prewitt. It is explained in figure-3 (a), (b) and can be expressed in equation 3.5, 3.6 and 3.7.It gives improved results than Prewit and Robert but again is not very efficient.

| | | |
|---|---|---|
| -1 | -2 | -1 |
| 0 | 0 | 0 |
| 1 | 2 | 1 |

Figure-3(a)

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

(b)

Gx=(Z7+2Z8+Z9)–(Z1+2Z2+Z3) …… 3.5

Gy = (Z3+2Z6+Z9) – (Z1+2Z4+Z7)……..... 3.6

|G| = |Gx| + |Gy|)……..... 3.7

### 3.3 Canny Edge Detector

Canny in 1986 developed an edge operator, which extracted not only step edges but also ridge and roof edges. Since in our project first step in the feature extraction sequence is to extract edges from the grey scale image, the use of the Canny edge detector to satisfy this step is justified as the Canny operator has a consistent response with single smoothing parameter.

The original grey scale image I [i, j] is smoothed with a Gaussian filter G [i, j, σ], where σ is the spread of the Gaussian that controls the degree of smoothing.

S [i, j] = G [i, j,σ] * I [i, j] ……... 3.8

The gradient magnitude M [i, j] and direction θ [i, j] are then computed from the smoothed image S [i, j]. The magnitude image is used to provide votes for possible edge pixels. Finally, the magnitude of each pixel in the image of possible edges is compared against a given high threshold. If a pixel passes the test, it is classified as an edge pixel, and the magnitudes of its neighbors are recursively tested against the low threshold.

The magnitude, or EDGE STRENGTH, of the gradient is then approximated using the formula:

|G| = |Gx| + |Gy|)……...… 3.9

Finding the edge direction is trivial once the gradient in the x and y directions are known. Whenever the gradient in the x direction is equal to zero, the edge direction has to be equal to 90 degrees or 0 degrees, depending on what the value of the gradient in the y-direction is equal to. If GY has a value of zero, the edge direction will equal 0 degrees. Otherwise the edge direction will equal 90 degrees. The formula for finding the edge direction is just.

Theta = inv tan (Gy / Gx) …….3.10

Once the edge direction is known, the next step is to relate the edge direction to a direction that can be traced in an image. So if the pixels of a 5x5 image are aligned as follows:

| | | | | |
|---|---|---|---|---|
| x | x | x | x | x |
| x | x | x | x | x |
| x | x | a | x | x |
| x | x | x | x | x |
| x | x | x | x | x |

Figure-3.4

Then, it can be seen by looking at pixel "**a**", there are only four possible directions when describing the surrounding pixels - 0 degrees (in the horizontal direction), 45 degrees (along the positive diagonal), 90 degrees (in the vertical direction), or 135 degrees (along the negative diagonal). So now the edge orientation has to be resolved into one of these four directions depending on which direction it is closest to (e.g. if the orientation angle is found to be 3 degrees, make it zero degrees).

Think of this as taking a semicircle in Figure-3.5 and dividing it into 5 regions.

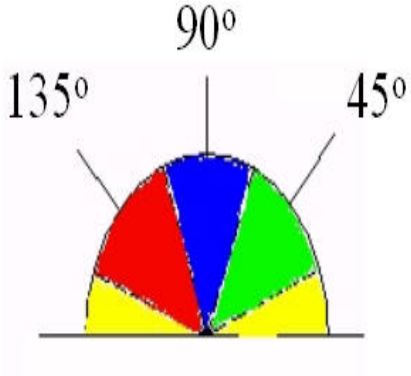



Figure 3-5

Therefore, any edge direction falling within the **yellow range** (0 to 22.5 & 157.5 to 180 degrees) is set to 0 degrees. Any edge direction falling in the **green range** (22.5 to 67.5 degrees) is set to 45 degrees. Any edge direction falling in the **blue range** (67.5 to 112.5 degrees) is set to 90 degrees. And finally, any edge direction falling within the **red range** (112.5 to 157.5 degrees) is set to 135 degrees.

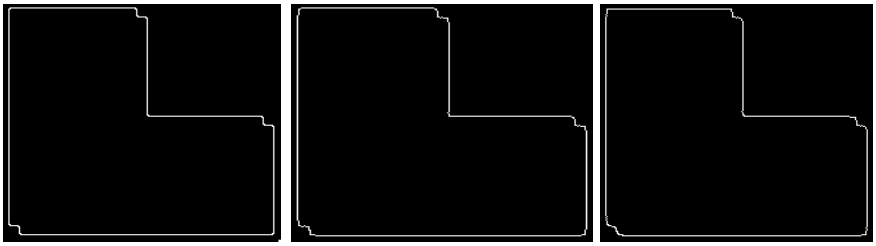
After the edge directions are known, nonmaximum suppression now has to be applied. Nonmaximum suppression is used to trace along the edge in the edge direction and suppress any pixel value (sets it equal to 0) that is not considered to be an edge. This will give a thin line in the output image.

### 3.3.1 Experiment and Results with the Canny Edge Detector

The edge detection is the first stage of the imaging processing , the results of three types of Canny edge detector on different images, including inbuilt MATLAB function, improved version of Canny by Naveed and Canny implemented by us in VC++ are shown in tables with noise in image and with different smoothing parameters i.e. Sigma “σ”:-

In figure 3-6 three different types of Canny Edge Detectors are applied keeping their smoothing factor i.e. sigma on 1 and their results are shown in table 3-1.The results show that all the lines have been detected by three types of canny edge detectors and their performance is quite satisfactory. The processing time with the VC canny is least thus having a very less computational time.

Figure-3.6 (a) L shape structure



(b) Improved canny (c) MAT canny (d) VC canny

| L shape structure | Processing Time | Detection Results |
|---|---|---|
| (b)-Improved Canny | 1.21 sec | Good |
| (c)-Mat Canny | 1.50sec | Good |
| (d)-VC Canny | Less than 1 sec | Good |

Table 3.1 Smoothing Factor i.e. $\sigma = 1$

In figure 3-7 a noise of .02 is added in image to check the efficiency of three edge detectors keeping the sigma 2 and the results are shown in table 3-2. The results indicate that improved canny function and the VC canny both could detect all the lines and remove the noise but matlab function failed in noise removal, a major drawback.

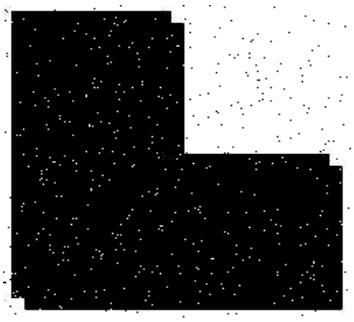
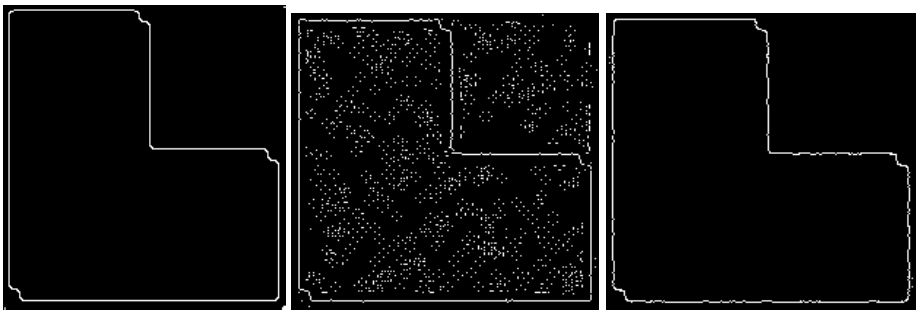


Figure-3.7 (a) L shape structure with Gaussian Noise .02



(b) Improved canny (c) MAT canny (d) VC canny

| L shape structure | Processing Time | Detection Results | Noise Removal |
|---|---|---|---|
| (b)-Improved Canny | 9.5 sec | Good | Yes |
| (c)-Mat Canny | 1.50sec | Good | No |
| (d)-VC Canny | Less than 1 sec | Good | Yes |

Table 3-2 Smoothing Factor i.e. $\sigma = 1$

Similarly In figure 3-8 same image is used with Gaussian noise of .05 and sigma is kept at σ=2.As the noise level has been increased the matlab canny function has completely distorted. But the performance of improved and VC canny function is quite satisfactory. All the lines have been detected and noise has been removed.

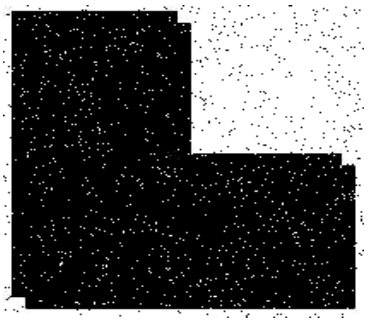
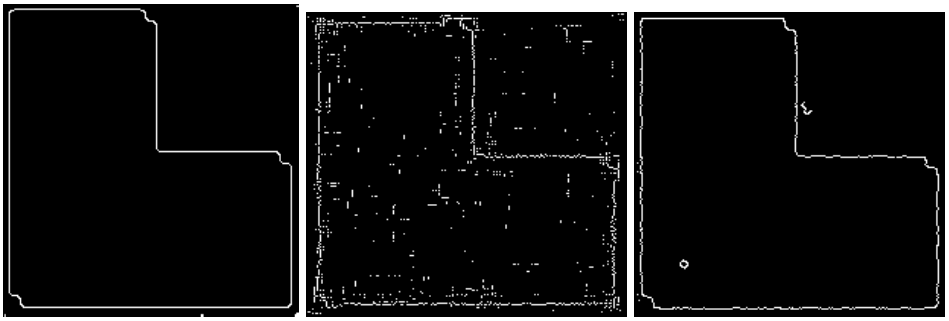


Figure-3.8 (a) L shape structure with Gaussian Noise .05



(b) Improved canny (c) MAT canny (d) VC canny

| L shape structure | Processing Time | Detection Results | Noise Removal |
|---|---|---|---|
| (b)-Improved Canny | 11.5 sec | Good | Yes |
| (c)-Mat Canny | 1.50sec | Distorted | No |
| (d)-VC Canny | Less than 1 sec | Good | Yes |

Table 3-3 Smoothing Factor i.e. σ= 2

In figure 3-9 a noise of .05 is added in image to check the efficiency of three edge detectors at extreme noise conditions while increasing smoothing function. The performance of VC function has slightly degraded. It show that if smoothing factor is increased much than the performance of VC Canny function decreases.

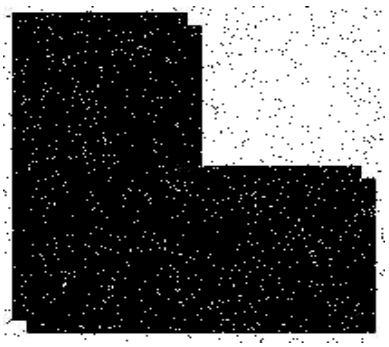
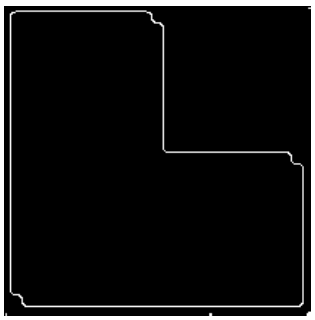
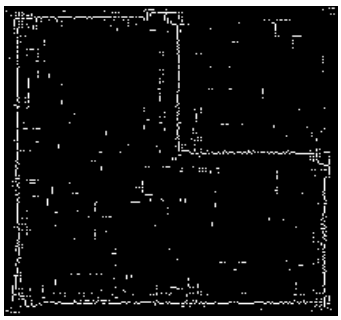
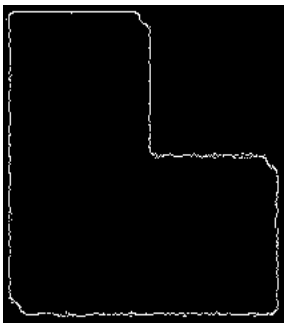


Figure3.9 (a) L shape structure with Gaussian Noise .05







(b) Improved canny (c) MAT canny (d) VC canny

| L shape structure | Processing Time | Detection Results | Noise Removal |
|---|---|---|---|
| (b)-Improved Canny | 12.21 sec | Good | Yes |
| (c)-Mat Canny | 1.50sec | Distorted | No |
| (d)-VC Canny | Less than 1 sec | Fair | Yes |

Table 3-4 Smoothing Factor i.e. $\sigma$=5

In figure 3-10 again three different types of Canny Edge Detectors are applied to check their efficiency on Monitor image keeping smoothing factor i.e. sigma on 1 and their results are shown in table 3-5. This result shows a very interesting thing that improved canny function failed in vertical line detection. This result proves the effectiveness of VC canny function and also that its performance is better on wide variety of figures.

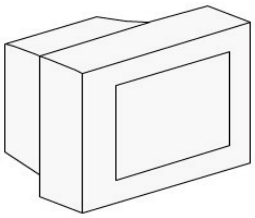
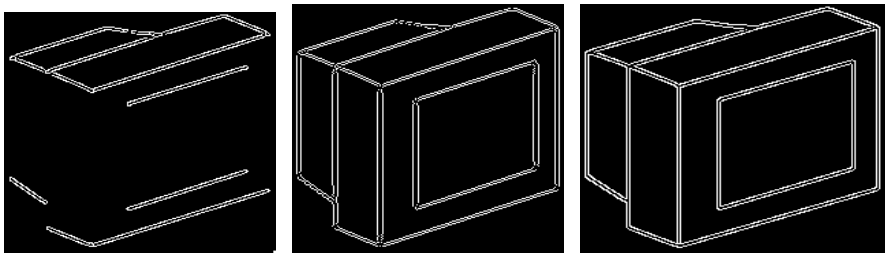


Figure3.10 (a) Monitor



(b) Improved canny (c) MAT canny (d) VC canny

| Monitor | Processing Time | Detection Results | Noise Removal |
|---|---|---|---|
| (b)-Improved Canny | 9.6 | Vertical lines missing | No |
| (c)-Mat Canny | 1.50sec | Good | No |
| (d)-VC Canny | Less than 1 sec | Good | No |

Table 3-5 Smoothing Factor i.e. $\sigma=1$

In figure 3-11 a noise of .02 is added in image keeping the sigma 2 and the results are shown in table 3-6.The increased noise has reduced the performance of matlab function. The only canny function which is working perfectly in this condition is VC canny function. That makes the base of over project very sound.

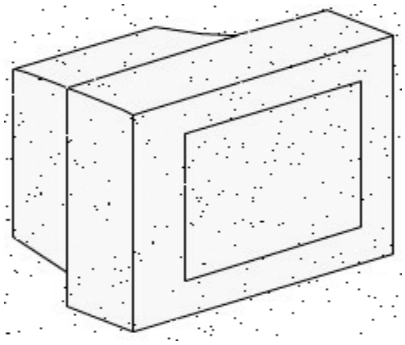
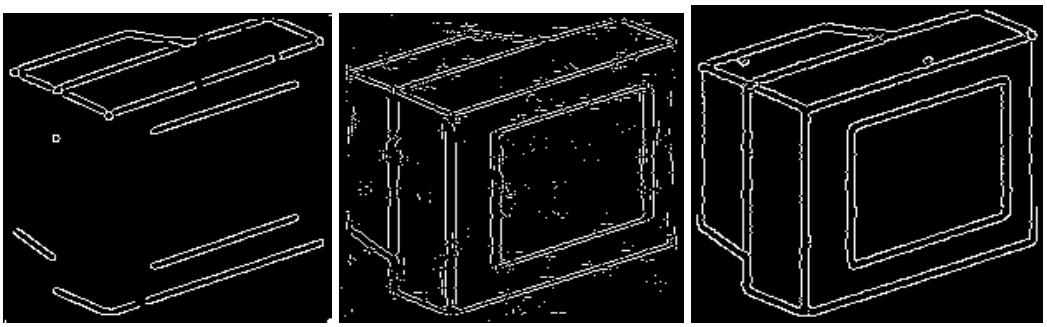


Figure3.11 (a) Monitor with Gaussian Noise .02



(b) Improved canny (c) MAT canny (d) VC canny

| Monitor | Processing Time | Detection Results | Noise Removal |
|---|---|---|---|
| (b)-Improved Canny | 11.5 | Vertical lines missing | Yes |
| (c)-Mat Canny | 1.50sec | Fair | No |
| (d)-VC Canny | Less than 1 sec | Good | Yes |

Table 3-6 Smoothing Factor i.e. σ=2

Similarly In figure 3-12, 3-13 same image is used with Gaussian noise of .05 and two smoothing factor 2 and 4. The results are shown in Table 3-7 and 3-8. The vertical lines of monitor remain undetected in improved canny .However a major draw back of VC canny has been noted when the smoothing factor is increased to 4 that whole image has gone blurred. We should mention here that default smoothing value in our project is 1.5 and it would never be smoothed at sigma 4.So we can compromise this drawback.

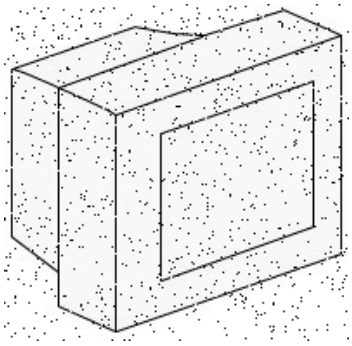
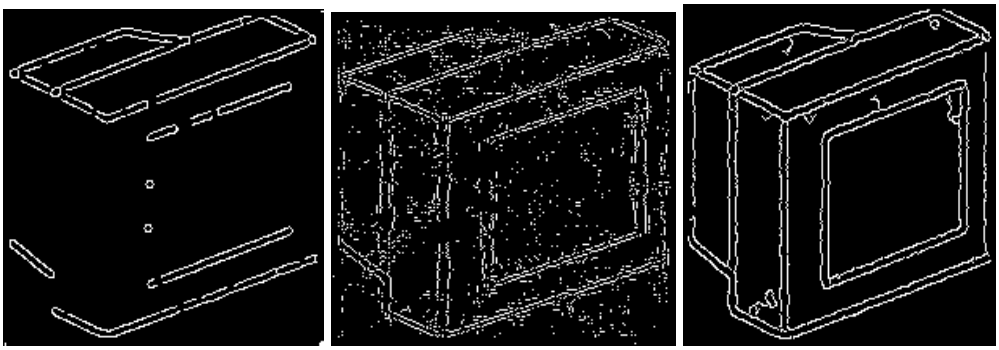


Figure3.12 (a) Monitor with Gaussian Noise .05



(b) Improved canny (c) MAT canny (d) VC canny

| Monitor | Processing Time | Detection Results | Noise Removal |
|---|---|---|---|
| (b)-Improved Canny | 11.5 | Vertical lines missing | Yes |
| (c)-Mat Canny | 1.50sec | Fair | Mo |
| (d)-VC Canny | Less than 1 sec | Good | Yes |

Table 3-7 Smoothing Factor i.e. σ=2

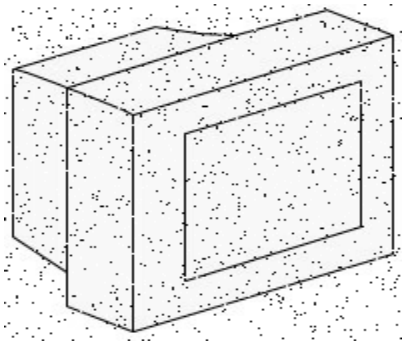
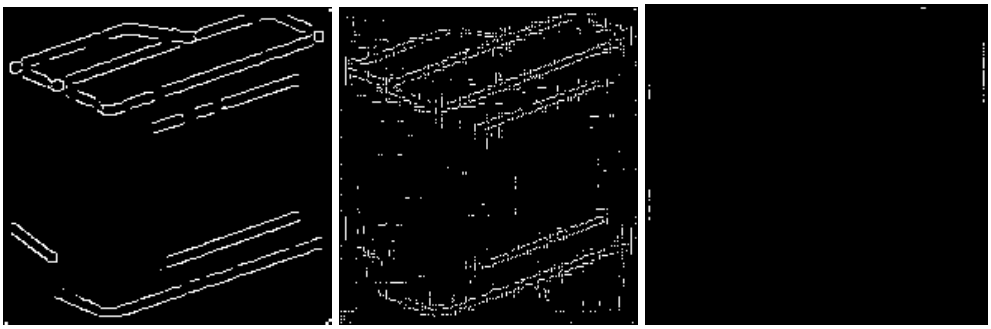


Figure3.13 (a) Monitor with Gaussian Noise .05



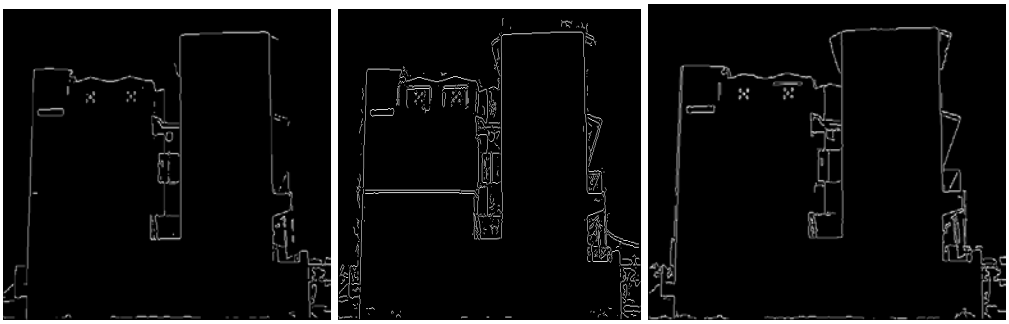
(b) Improved canny (c) MAT canny (d) VC canny

| Monitor | Processing Time | Detection Results | Noise Removal |
|---|---|---|---|
| (b)-Improved Canny | 12.21 sec | Vertical lines missing | Yes |
| (c)-Mat Canny | 1.59 sec | Fair | No |
| (d)-VC Canny | Less than 1 sec | Bad | Yes |

Table 3-8 Smoothing Factor i.e. $\sigma=4$

The whole effort was made in the project to detect the man made objects in the image, so the all three Canny edge detectors are now applied on man made object in figure 3-14 keeping the smoothing factor at 1 and their results are shown in table 3-9. The figure is of Capt shah Muhammad Saud's house side pose. It provides excellent view for edge detection. All three functions have worked well detecting all the lines and giving good results.

Figure3.14 (a) Saud's House



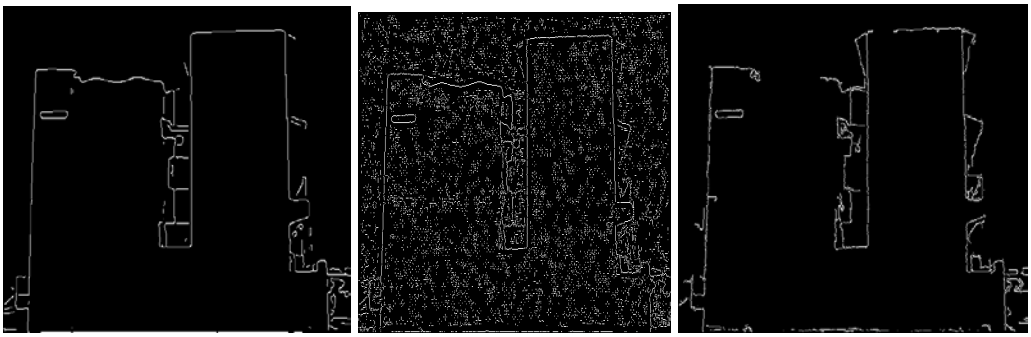
(b) Improved canny (c) MAT canny (d) VC canny

| Saud,s House | Processing Time | Detection Results | Noise Removal |
|---|---|---|---|
| (b)-Improved Canny | 22.66sec | Good | Yes |
| (c)-Mat Canny | 2.45sec | Good | Yes |
| (d)-VC Canny | Less than 2 sec | Good | Yes |

Table 3-9 Smoothing Factor i.e. $\sigma = 1$

In figure 3-15 a noise of .02 is added in image keeping the sigma 2 and the results are shown in table 3-10.With the introduction of slight noise, the matlab function has failed. So here we prove that matlab function is not at all suitable for our project so we needed a good edge detector. The improved function performance is well on Saud's house but it fails on variety of figures. VC canny is dynamic in performance and is robust to noise.

Figure3.15 (a) Saud's House with Gaussian Noise .02



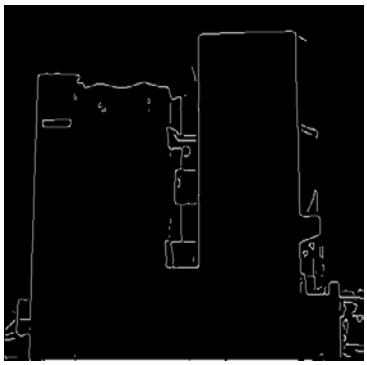
(b) Improved canny (c) MAT canny (d) VC canny

| Saud's House | Processing Time | Detection Results | Noise Removal |
|---|---|---|---|
| (b)-Improved Canny | 38.66sec | Good | Yes |
| (c)-Mat Canny | 2.46sec | Fair | No |
| (d)-VC Canny | Less than 3 sec | Good | Yes |

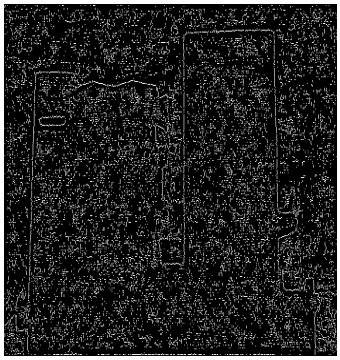
Table 3-10 Smoothing Factor i.e. σ=2

. Similarly In figure 3-16, 3-17 same image is used with Gaussian noise of .05 and two smoothing factors of 2 and 4 and their results are shown in Table 3-11 and 3-12. The only draw back of VC canny which is now confirmed on second figure that with increasing smoothing function it looses it performance. But our assumption of keeping at 1.5 overcomes this drawback.

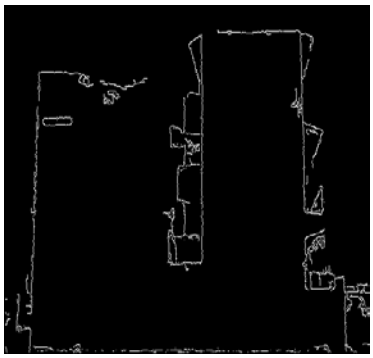
Figure3.16 (a) Saud's House with Gaussian Noise .05



(b) Improved canny



(c) MAT canny

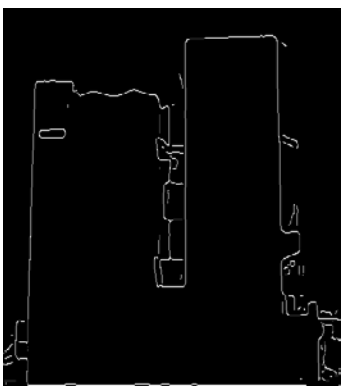


(d) VC canny

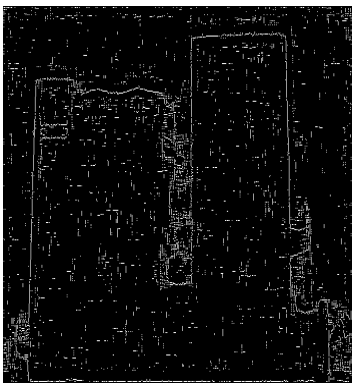
| Saud's House | Processing Time | Detection Results | Noise Removal |
|---|---|---|---|
| (b)-Improved Canny | 38.73 sec | Good | Yes |
| (c)-Mat Canny | 2.46 sec | Bad | No |
| (d)-VC Canny | Less than 4 sec | Good | Yes |

Table 3-11 Smoothing Factor i.e. σ=2

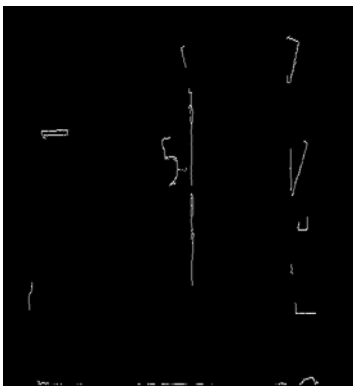
Figure3.17 (a) Saud's House with Gaussian Noise .05



(b) Improved canny



(c) MAT canny



(d) VC canny

| Saud's House | Processing Time | Detection Results | Noise Removal |
|---|---|---|---|
| (b)-Improved Canny | 38.73 sec | Good | Yes |
| (c)-Mat Canny | 2.50 sec | Fair | No |
| (d)-VC Canny | Less than 4 sec | Fair | Yes |

Table 3-12 Smoothing Factor i.e. σ=4

### 3.3.2 Difference between Canny Detectors

We made different combination of the thresholds, smoothing factors, smoothing mask size and non maximal suppression process used in canny and than tried to obtain a better canny edge detector for the manmade buildings and objects. Our efforts proved useful and we designed an enhanced version of canny in its performance. However it should be mentioned that it is almost impossible to design an edge detector which is universal in nature. So our canny may also show some limitations. But we have tried to overcome its problems as much as possible.

| | Matlab Canny | Naveeds Canny | VC++ Canny |
|---|---|---|---|
| Smoothing factor (Default) | 1.0 | 2.0 | 1.5 |
| Non max suppression | 2 or 3 pixel wide edge | 1 or 2 pixel wide edge | 1 or 2 pixel wide edge |
| Low Threshold | Low threshold=0.4*High Threshold | Single threshold is used level=alfa*(I_max-I_min)+I_min | 5 (can be adjusted) |
| High Threshold | | | 20 (can be adjusted) |
| Filter mask size | Gaussian filter is used. 1D mask is used 17*1 | Gaussian filter is used.2D 10*10 Mask is used. | Gaussian filter is used 7*7 mask is used. |

Table 3.13 Comparison of Three Canny's

# 6 Line Detection

Hough transform is the most powerful technique to detect lines from the images. The transform maps patterns from the image plane to the parametric space. It is essentially a voting process where each feature point votes for all the possible patterns passing through that point. The votes are than accumulated in an accumulator array, and the pattern receiving the maximum vote is determined. The advantages of the transform are its robustness to noise in the image and discontinuities in the pattern to be detected. The above two features of the transform make it very attractive for the detection of patterns in the real world images. The disadvantage of the transform is its requirement for a large amount of computing power and storage requirements. The computational requirement of the transform has deterred its widespread use in real time applications. This time and space requirements depend on the number of parameters required to describe the pattern, the resolution of accumulator array, and the image resolution. A straight line is the simplest of all geometric patterns and can be described with only two parameters.

Several variants of the Hough transform have been proposed in the literature to reduce the time and space complexity. Performance of the transform with respect to accuracy of detection has been discussed. When the Hough Transform is used to detect straight lines, it provides only its parameters like slope intercept and the normal forms with infinite lines. It does not provide any information regarding the length, position and endpoints of the line segment in the image plane. However, in computer vision application, the length and exact position of a line segment in addition to its normal parameters are required for locating the line segment. Atiquzaman [2] in his paper gave a very efficient algorithm for complete line segment detection.

## 4.1 Related Works

Different algorithms have been studied for finding the length and end points of a line from the Hough accumulator array. Some algorithms are based on the projections of the line on the X or Y-axis. They suffer from the drawback of a single line segment being interpreted as multiple line segments. For our project purpose we have studied the overview of Hough transform, the work done by M. Atiquzzaman and M. W. Akhter [2] in general and the work proposed by Naveed [3] in particular. He used a different technique of least square error fit to more precisely estimate the line parameters. In addition to this we have thoroughly studied the MATLAB based Radon Transform and improved its performance by developing and applying new filters in its algorithm.

## 4.2 Line Definition

There are certain unavoidable problems that occur when detecting lines in an image. These problems stem from the fact that, even though a digital line is not the same as its mathematical counterpart, the mathematical definition is used for detection purposes. In order to take advantage of both concepts of a line, or more properly a line segment, it can be defined as by [3], “a set of pixels that fit a mathematical line model”.

## 4.3 Work done by Naveed on Hough Transform

Refer to Naveed [3] thesis, Chapter 4, page 25-33

## 4.4 Radon Transform(Hough in Matlab)

Radon transform is a MATLAB built-in function. We tried it for line detection. The Radon function computes the Radon transform, which is the projection of the image intensity along a radial line oriented at a specified angle. The Radon transform is closely related to a common computer vision operation known as the Hough transform. Radon function can be used to implement a form of the Hough transform used to detect straight lines. The steps are.

- Compute a binary edge image using the edge function either canny or Sobel
- Compute the Radon transform of the edge image.
- Find the locations of strong peaks in the Radon transform matrix.

The locations of these peaks correspond to the location of straight lines in the original image. In step three we used two different methods to get lines. These methods were entirely invented by us.

### 4.4.1 Regional Maxima Method

In this method the regional maxima of the Radon transformed image was taken. These local maxima were the accumulator locations indicating the presence of lines. Only regional maxima's were considered and rest all in the accumulator bin are made zero, then inverse Radon transform is carried out to show lines.
The Figure 4-1 explains the regional maxima's,

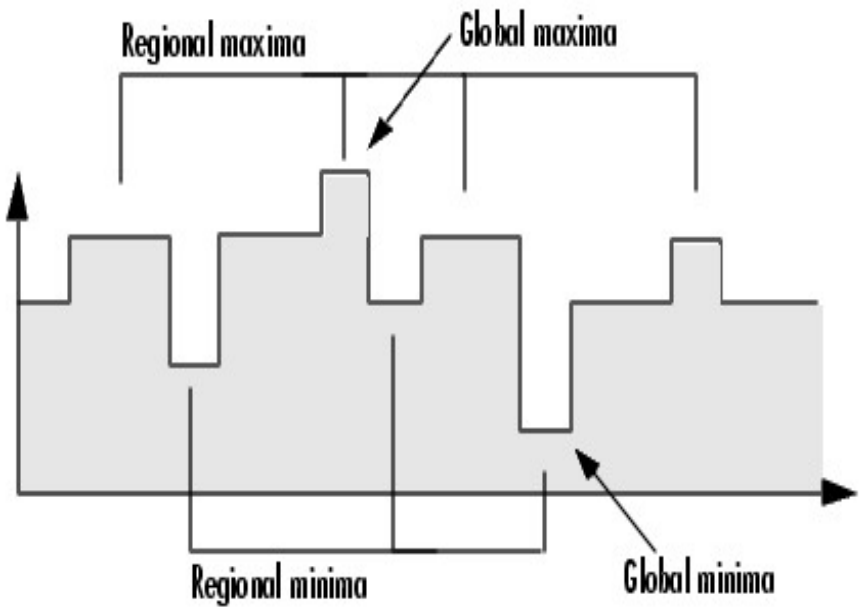



Figure 4-1

e.g. we have a portion of the image data of Figure 4-2 (L shape) after the radon transform in the accumulator bins Figure 4-3.

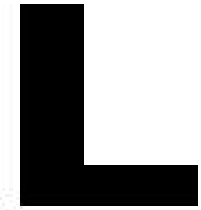


Figure 4-2 (L shape)

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 6 | 17 | 28 | 33 | 33 |
| 0 | 0 | 2 | 12 | 28 | 40 | 43 | 40 | 36 | 31 |
| 32 | 36 | 57 | 66 | 61 | 52 | 44 | 37 | 32 | 28 |
| 195 | 171 | 115 | 81 | 62 | 50 | 41 | 35 | 32 | 28 |
| 46 | 65 | 89 | 78 | 59 | 48 | 41 | 35 | 31 | 28 |
| 14 | 13 | 19 | 42 | 54 | 48 | 41 | 35 | 31 | 28 |
| 9 | 10 | 11 | 12 | 21 | 38 | 40 | 35 | 31 | 28 |
| 8 | 8 | 9 | 11 | 11 | 13 | 25 | 34 | 31 | 28 |
| 8 | 8 | 8 | 8 | 10 | 11 | 11 | 16 | 27 | 28 |
| 8 | 8 | 8 | 8 | 8 | 9 | 11 | 11 | 12 | 20 |
| 8 | 8 | 8 | 8 | 8 | 8 | 8 | 10 | 11 | 11 |

Figure 4-3 (accumulator bin)

After getting local maxima accumulator bin becomes.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 195 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure (accumulator bin) 4-4

### 4.4.2 Threshold Method

In this method the mean of all the values present in the bin is calculated. Then it is multiplied by a constant to get a threshold value. We have selected 6 as a threshold constant after testing it at no of different type of pictures. This threshold is than used to filter out the accumulator bins. All the bins above threshold are kept, rest made zero, then inverse Radon transform is carried out to get lines. The mean in this image is 6 and threshold ratio is 6.Considering again the accumulator bin of Figure 4-2 (L shape), the filtered data is in Figure 4-5.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 33 | 33 |
| 0 | 0 | 0 | 0 | 0 | 39 | 42 | 40 | 35 | 30 |
| 32 | 35 | 57 | 66 | 61 | 52 | 44 | 37 | 32 | 0 |
| 194 | 171 | 115 | 81 | 62 | 49 | 41 | 35 | 31 | 0 |
| 45 | 65 | 88 | 77 | 59 | 47 | 40 | 35 | 31 | 0 |
| 0 | 0 | 0 | 41 | 54 | 47 | 40 | 35 | 31 | 0 |
| 0 | 0 | 0 | 0 | 0 | 37 | 40 | 35 | 31 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 33 | 31 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure (accumulator bin) 4-5

### 4.4.3 Algorithm for threshold method

R=radon transformed image.

Mean=mean(R)  // mean of radon values

Thresh=mean * 5

RF=Filtered Radon data.

[x y]=size (RF)

```
For (p=1: x)
    For (q=1: y)
        If (RF (p, q) <Thresh)
            RF (p, q) =0;
        End
    End
End
```

### 4.4.4 Comparison of both methods

Here we will carry out comparison of both the methods. Three pictures have been selected; two are geometrical shapes while one is building figure. All the lines are detected by Local/regional maxima method in Figure 4-6 but with extended lengths. However all the lines are perfectly detected by threshold method with exact lengths. So the threshold method has a much better performance than regional maxima method.

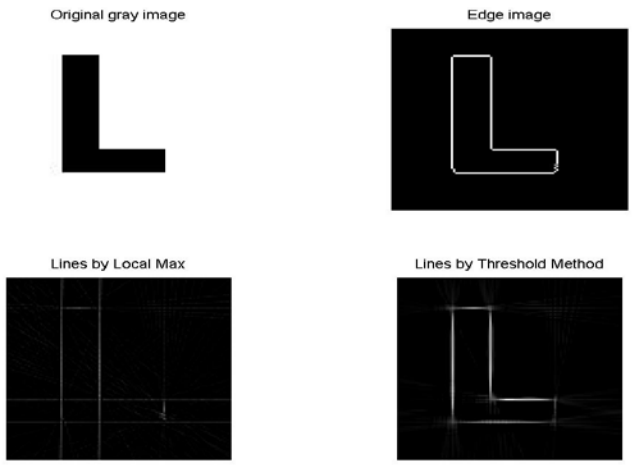



Figure 4-6

In Figure 4-7 (mosque), again almost all lines are detected by Local /Regional maxima method but with noise. The threshold method has detected the lines in a better way but it has missed out some detail like conical top shape of both small minarets is missing. Threshold method is also noise free. So in overall threshold method has performed better than local maxima.

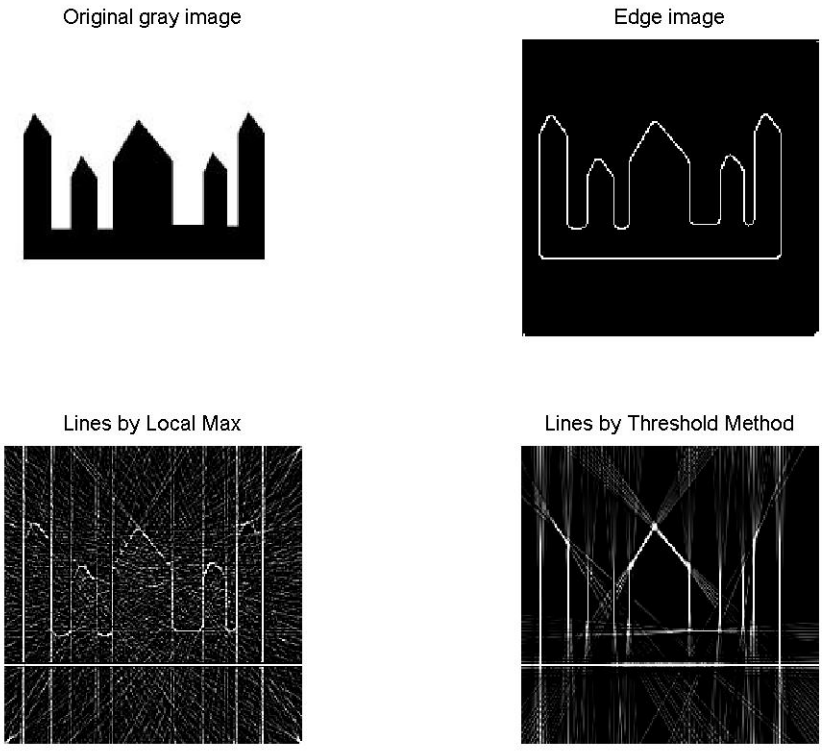



Figure 4-7

When Radon transform was applied on a building it failed as a whole to get proper legible lines. Although its performance till geometric shapes is quite fine but it is unsatisfactory on the manmade buildings which is our scope of project. So we can say that radon transform is only suitable for geometric shapes and not for detecting lines from manmade buildings

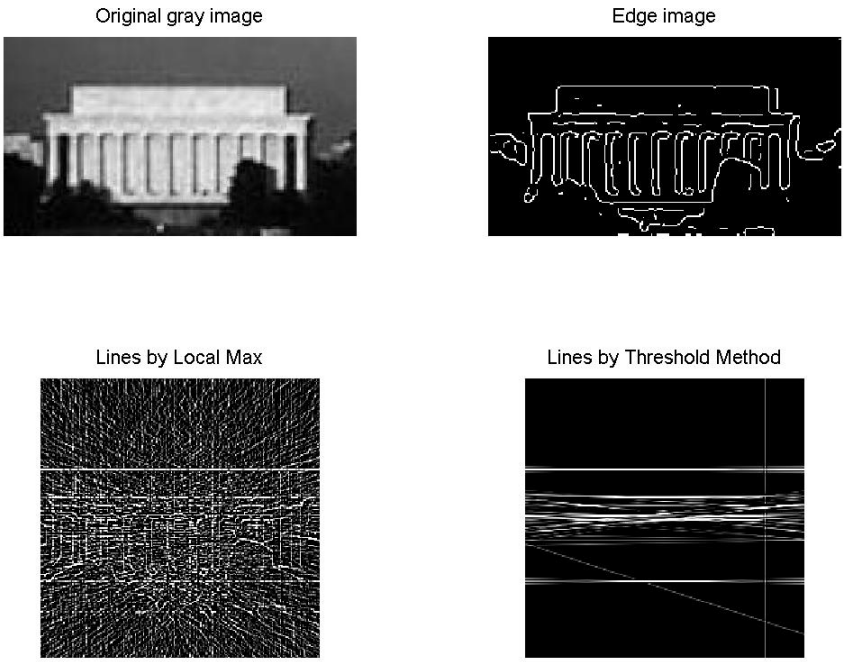



Figure 4-8

## 4.5 Hough transform

The Hough Transform is a global method for finding straight lines hidden in larger amounts of other data. It is an important technique in image processing. To understand the Hough Transform, it is important to know what the Hough space is. Each point (d, T) in Hough space corresponds to a line at angle T and distance d from the origin in the original data space. The value of a function in Hough space gives the point density along a line in the data space. The Hough Transform utilizes this by the following method.

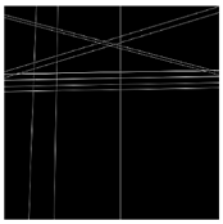
For each point in the original space consider all the lines which go through that point at a particular discrete set of angles, chosen a priori. For each angle T, calculate the distance to the line through the point at that angle and discretise that distance using an a priori chosen discretisation, giving value d. Make a corresponding discretisation of the Hough space - this will result in a set of boxes in Hough space.

These boxes are called the Hough accumulators. For each line we consider above, we increment a count (initialized at zero) in the Hough accumulator at point (d, T). After considering all the lines through all the points, a Hough accumulator with a high value will probably correspond to a line of points. In fact for uniformly distributed points, each Hough box should have a Poisson distributed count with mean given by the length of the line times discretisation width times uniform point density. Some Hough boxes will correspond to longer lines than other, resulting in the pattern seen in the star/galaxy data transform below. A count which is in the tail of the relevant Poisson distribution is unlikely to be the result of the underlying uniform data, and hence more likely to be the result of some line of points. Giving some prior model for the number of points in a line will allow a proper assessment of whether there is a line at the relevant angle and distance from the origin.

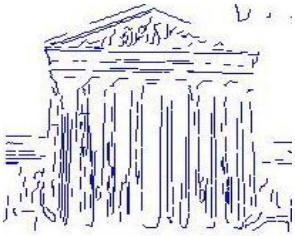
## 4.6 Line Detection Results

The results of the line detection procedure for all of the test images are shown in figures 4-9, 4-10 and 4-11. The first image we investigate is a building. Figure 4-9 (b), (c) and (d) shows the results of line detection by using three different techniques. Figure 4-9 (b) shows line detection results by Radon, as it is only effective for geometric shapes so the results are not good as required for the manmade objects because it can not detect the pillars. Figure 4-9 (c) is the result detected by Naveed's algorithm, it gives all the lines effectively but some un-necessary details are also included. In Figure 4-9 (d) we used the modified version of Hough transform in VC and the results are very good with detecting all the lines and processing time is also very less.

Figure 4-9(a) Pillars



(b)Radon



(c) Naveed, s

(d) VC HT

| Pillars | Processing Time | Detection Results | Noise Removal |
|---|---|---|---|
| (b)- Radon | 17 sec | Bad | No |
| (c)-Naveed's | 401 sec | Good | No |
| (d)- VC HT | Less than 3 sec | Good | Yes |

Table 4-1

Figures 4-10 (a), (b), (c) and (d) show the result on same pattern by using three different techniques. Figure 4-10 (b) shows very good result as it detects all the lines very effectively but with strong extended shadows, it shows that Radon is very good for geometric shape. Figure 4-10 (c) is the result by Naveed's algorithm and it gives complete line detection in all respect. Similarly Figure 4-10 (d) is the result by VC ( HT),

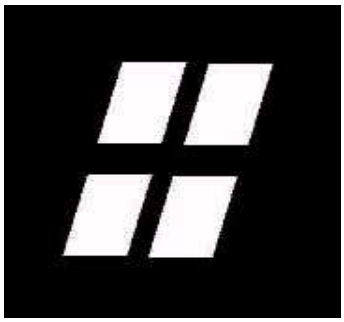
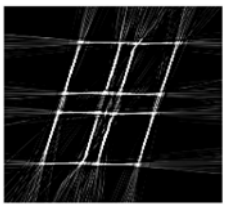
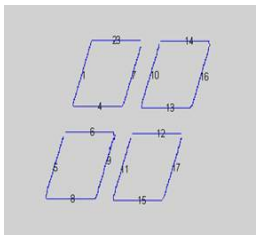


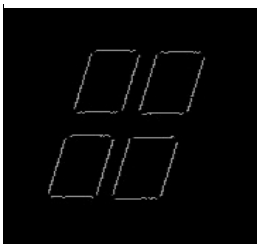
Figure 4-10 (a) Boxes



(b)Radon



(c) Naveed, s



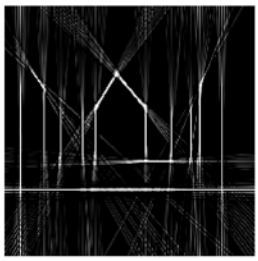
(d) VC HT

| Boxes | Processing Time | Detection Results | Noise Removal |
|---|---|---|---|
| (b)- Radon | 4.6 sec | Good | No |
| (c)-Naveed's | 7.9 sec | Good | Yes |
| (d)- VC HT | Less than 1.5 sec | Good | Yes |

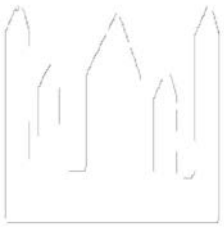
Table 4-2

Figures 4-11(a), (b), (c) and (d) represent results with above three algorithms. Figure 4-11(b) gives the result by Radon and it is good as it has detected all the lines very effectively but with some un-necessary details due to strong shadows. Figure 4-11 (c) is the result by Naveed‘s algorithm and it misses some of its inner lines due to their individual bins did not receive enough votes to warrant line fitting , similarly Figure 4-11 (d) represents the result by VC (HT) and its result is also good.

Figure 4-11 (a) Mosque



(b)Radon



(c) Naveed, s

(d) VC HT

| Mosque | Processing Time | Detection Results | Noise Removal |
|---|---|---|---|
| (b)- Radon | 4.68 sec | Good | No |
| (c)-Naveed’s | 28 sec | Good | Yes |
| (d)- VC HT | Less than 1 sec | Good | Yes |

Table 4-3

# 5. Corner and Ribbon Detection

## 5.1 Corner Definition and Detection

A corner is defined to be the point of intersection of two distinct proximal line segments making a shape of "L". Here we will not consider corners ("⊥"). The attributes of a corner consist of an angle and a location. Since lines are fit in a non-quantized sense, it is possible to define the location of a corner with sub-pixel accuracy. In order to detect corners, the line lists found in the line detection stage must be processed [3].

$$\kappa(L_i, L_j) < \delta_n \qquad 5.1$$

$\kappa$ is function that finds the corner between pair of lines [$L_i$ , $L_j$] if the threshold is less than $\delta_n$ , where i and j are line labels and $i \neq j$ and $j \in [i - 10, i + 10]$.

$m = (y - y_1) / (x - x_1)$ is the slope of the line $L_i$ with two end points (y, x), ($y_1$, $x_1$). First, the slope of each line $L_i$ is computed with equation 4.2 and then compared against the slope of *Lj* (where $i \neq j$ and $j \in [i + 10, i - 10]$). If the slopes are commensurate, the point of intersection is computed. Because the points of interest occur as the intersection of line segments rather than lines, the computed intersection point must be compared against the initial and terminal points of each line segment. If the point is within a certain distance, the corner threshold, of either the initial or the terminal point of both segments, or if the point lies on both segments, the pair is classified as a corner. The default distance for the corner threshold is ten pixels. Essentially, the corner threshold is the distance that each line segment is extended in both directions before testing for intersections.

After detecting the corner point, the measure of the corner angle is computed.

$$\text{Angle}_{LiLj} = \tan^{-1}((m_j - m_i)/(1 + m_i * m_j)) \qquad 5.2$$

Where $m_i$ is slope of $_{Li}$ and $m_j$ is slope of $_{Lj}$, and $i \neq j$

## 5.2 Corner Detection Observations

Since corner detection is dependent upon both the edge detection and line detection, there are three parameters that affect its performance, the Gaussian spread, the line length tolerance, and the corner threshold. Because the number of parameters is three, the reasons for incorrect corner detection may be difficult to pinpoint, and it may be difficult to find the best configuration of these parameters for a particular image, the reader can verify that, for the majority of the examples, both false positives and false negatives (non-detection of corners) can be attributed to faulty line detection. In figure 5-1 two different types of Corner Detectors are applied and their results are shown in table 5-1.The results show that no corner has been detected in the figure of monitor. This no detection of even a single corner in monitor is a serious drawback in Naveed corner detector. This no detection is primarily because of the poor edge detection by the edge detector. So here we can say that very good edge detection is pivotal in our project. It also justifies our efforts on a good edge detector. More so all the corners are detected at their right places and no false corner has been detected and all the detected corners have been rightly placed.

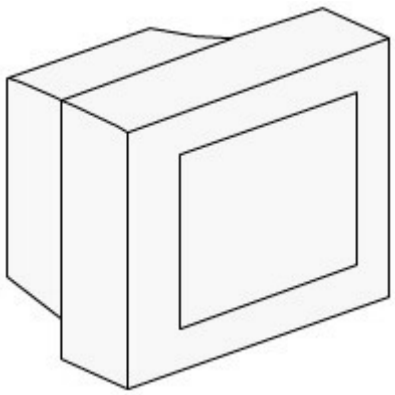
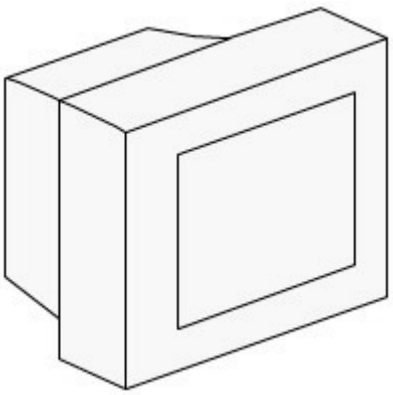
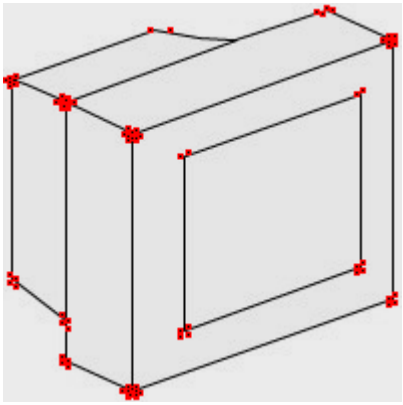


Figure 5-1 (a) Monitor Test image for corner detection



(b) Naveed corner detection



(c) VC corner detection

| Monitor | Processing Time | False positives | False negatives | Real |
|---|---|---|---|---|
| (b)-Naveed's corner detection | 6 sec | - | - | no |
| (c)-VC corner detection | 2 sec | No | No | All |

Table 5-1

In figure 5-2 two different types of Corner Detectors are applied and their results are shown in table, 5-2. When we observe closely, we see that the corner detection in VC function is much better that the Naveed's corner detection. In Naveed's corner detection almost no corner is detected on the pillars of mosque and few corners are detected on the grass. However in VC corner detection a number of corners are detected at the pillars and detection of false corners is very low.

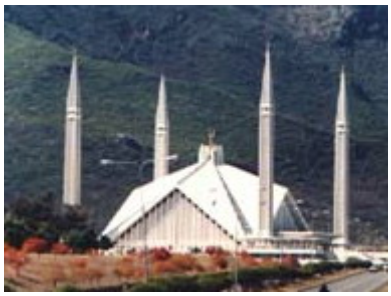
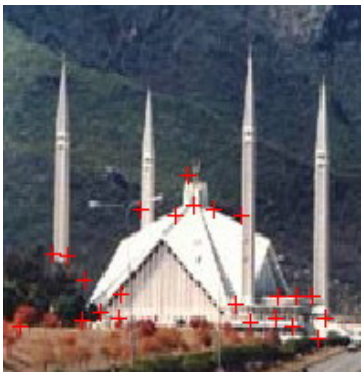


Figure 5-2 (a) Mosque Test image for corner detection



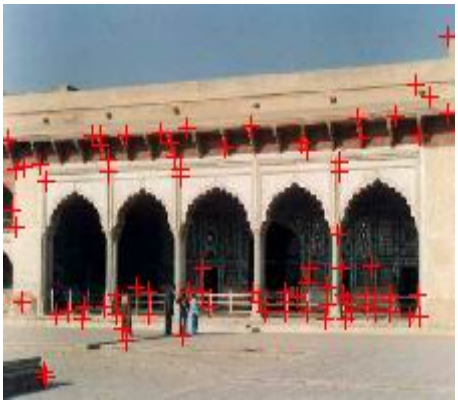
(b) Naveed corner detection (c) VC corner detection

| Mosque | Processing Time | False positives | False negatives | Real |
|---|---|---|---|---|
| (b)-Naveed's corner detection | 3 sec | Yes | Yes | Yes |
| (c)-VC corner detection | 1 sec | Yes | Yes | Yes |

Table 5-2

In this picture of a building the corner detected by naveed detector are not well defined. When we see as a whole, the VC corner detection is much better and finely oriented.

Figure 5-3 (a) Doors Test image for corner detection



(b) Naveed corner detection

(c) VC corner detection

| Mosque | Processing Time | False positives | False negatives | Real |
|---|---|---|---|---|
| (b)-Naveed corner detection | 713 sec | Yes | Yes | Yes |
| (c)-VC corner detection | 18 sec | Yes | Yes | Yes |

Table 5-3

## 5.3 Ribbon Detection

Stockman and Shapiro [1] define a ribbon as "an elongated region that is approximately symmetrical about its major axis". This definition encompasses many varied objects, including picture frames; bottles, columns, and any 2-dimensional object that possesses symmetry. When the images of such objects are projected onto a plane, the symmetry is translated into curves with reflective symmetry about an axis. When an actual cylindrical object is projected onto a plane, its symmetry is translated into parallel lines. The ribbon detection algorithm will only identify those ribbons that are the result of projected cylinders and rectangular prisms, namely straight ribbons.

## 5.4 Ribbon Definition

Naveed [3] for the purpose of identifying straight ribbons, a more practical definition is, a straight ribbon to be two lines whose edge gradients differ by $180^o \pm 10^o$, have approximately the same length, approximately the same slopes and the width should be ≤ 1.5 times the line length. To be more precise, given two lines, $L_1$ and $L_2$, they form a ribbon if and only if the following conditions are satisfied:

- $170^o \leq |\ \theta_1 - \theta_2\ | \geq 190^o$ where $\theta_1$ and $\theta_2$ are the respective average edge gradient of $L_1$ and $L_2$.
- $l(L_i) \geq l(L_j)/2$ For $1 \leq i, j \leq 2$ where $l(L)$ is the length of line segment L.
- The projected line $L_1$ should overlap $L_2$ and vise versa.

A detected ribbon then inherits the attributes of each of its lines, as well as a width, length, and one of two possible types. The length of the ribbon is defined to be the average of the lengths of the two lines. The width of the ribbon is the distance between the lines.

### 5.4.1 Type of Ribbons

Ribbons are divided into two different types: Type-1 and Type-2[3]. If the edge gradient $\theta_1$ at line $L_1$ is greater than the edge gradient $\theta_2$ at line $L_2$, the ribbon is Type-1, otherwise it is Type 2. Essentially, a Type-1 ribbon is the one in which the gradient vectors of the lines pointing away from each other, and a Type-2 ribbon is one with gradient vectors pointing towards each other. Figure 5-4 gives examples of the two different types.

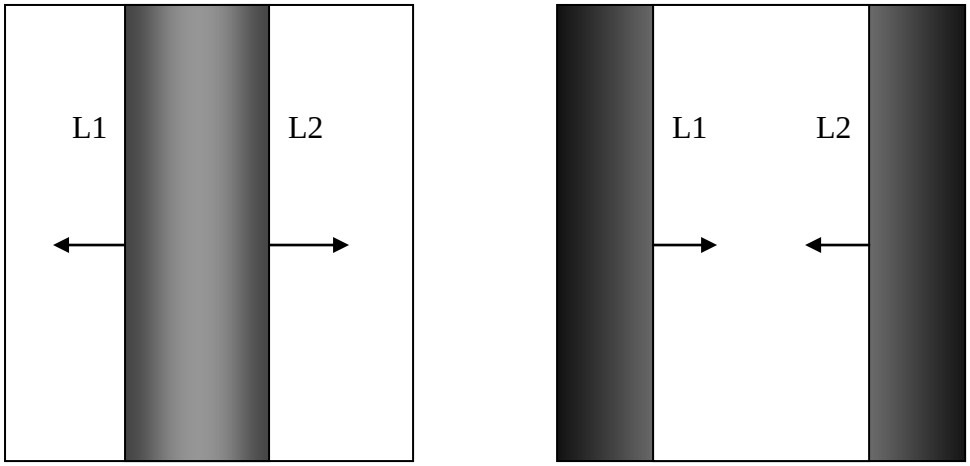



Figure 5-4 (a) is Type-1 ribbon and (b) is Type-2 ribbon (L1 must be left of L2)

### 5.4.2 Ribbon Detection Results

Since ribbon detection is dependent upon both the edge detection and line detection, there are three parameters that affect its performance, the Gaussian spread, the line length tolerance, and the corner threshold. Because the number of parameters is three, the reasons for incorrect ribbon detection may be difficult to pinpoint, and it may be difficult to find the best configuration of these parameters for a particular image. We will carry out comparison of both the methods. Three pictures have been selected; two are geometrical shapes while one is building figure. All the ribbons are missing in Figure 5-5(b) by Naveed's algo and all the ribbons are detected by VC. This show a major shortcoming of Naveed's ribbon detection algorithm. It also proves again that our ribbon detection is much robust and covers a variety of man made buildings etc.

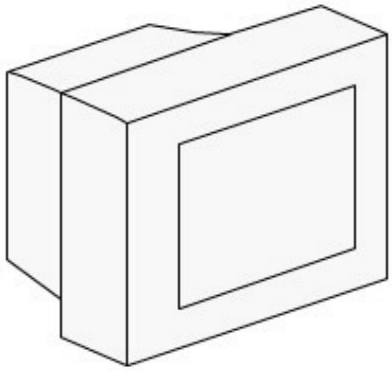
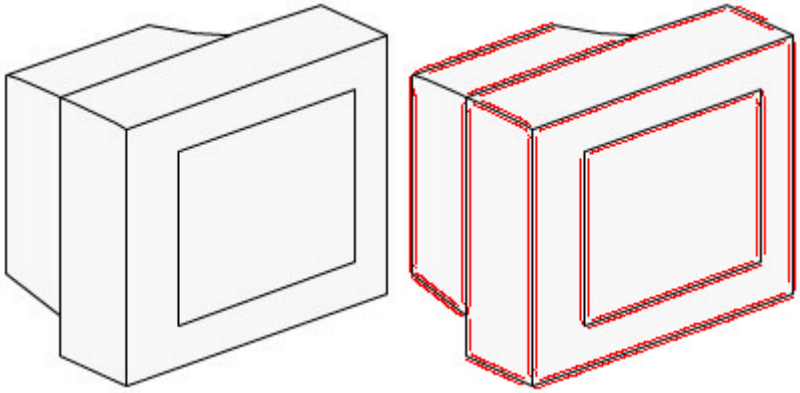


Figure 5-5Monitor



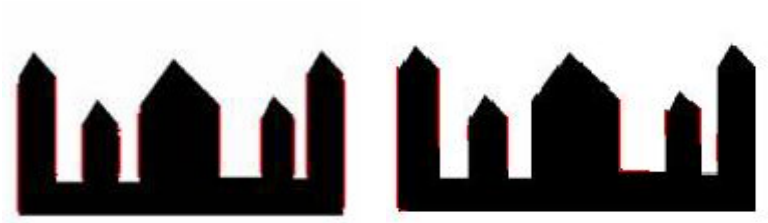
(b) Naveed, s ribbon detection (d) VC ribbon detection

| Monitor | Processing Time | Ribbon detection |
|---|---|---|
| (b)-Naveed's ribbon detection | - | No |
| (c)-VC ribbon detection | 2 sec | Yes |

Table 5-4

In this geometrical shape, although all the ribbons are detected by both the ribbon detectors but once we observe closely we see that the ribbons of VC detector is more finely defined, full in their length as compared to Naveed's ribbon detector which are half of the original length of theoretical ribbons..

Figure 5-6 Mosque



(b) Naveed, s ribbon detection (c) VC ribbon detection

| Mosque | Processing Time | Ribbon detection |
|---|---|---|
| (b)-Naveed's ribbon detection | 27 sec | Yes |
| (c)-VC ribbon detection | 3 sec | Yes |

Table 5-5

In this figure of Faisal mosque, we observe that extra ribbons are detected by VC ribbon detector. A ribbon has been detected at front right roof wall which is missing in Naveed's ribbon detector. More so ribbons have also been detected in the right bottom of the Mosque. Some ribbons have been detected by VC detector in the grass as well which shows its limitation.

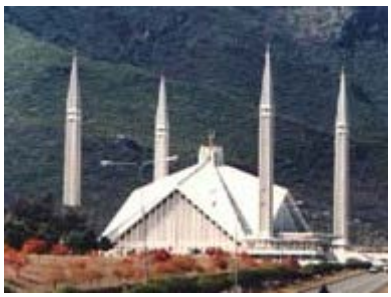
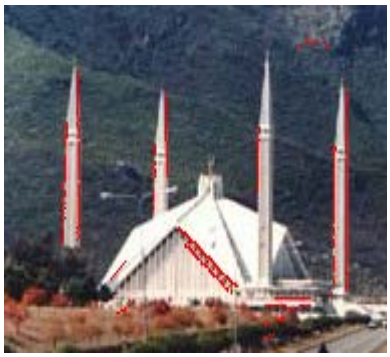


Figure 5-7 Faisal Mosque



(b) Naveed, s ribbon detection (c) VC ribbon detection

| Faisal Mosque | Processing Time | Ribbon detection |
|---|---|---|
| (b)-Naveed's ribbon detection | 111 sec | Yes |
| (c)-VC ribbon detection | 3 sec | Yes |

Table 5-5

## 6. Image Representations and Graph Matching

There are many types of features such as global, local and relational features that can be used for object recognition and matching. Global features are region based and can be obtained either by considering all points inside the region, or only those points on the boundary of the region. In any of the case, the purpose is to find the locations, intensity characteristics, color, and spatial relations of these points. Local features usually represent a small area of a region. Some local features are corners, and boundary segments. Relational features are based on relative position of different entities, such as distance between features, relative angle, orientations and precincts. These features are very important in object recognition as even change in lighting conditions, view angle and noise will not greatly influence these features [3]. The algorithms used in image representation are also designed by Naveed and we are just giving an overview of the work as it is very important for graph matching purpose in image retrieval. We have also proposed two new representations.

### 6.1 Line Structure Representation

Line structure is divided in to two groups depending upon its properties, descriptor and relationship. Descriptors of a line represent different properties of line such as line length, line slope and edge gradient, and relationships represent distance between line segments and line orientations [3].

### 6.2 Corner Structure Representation

A geometric descriptor of the corner structure is ‘angle of a corner’, and relations that a corner makes with other corners are:-

- Distance between Corners.
- Corner’s Orientation.
- Corner’s Precincts.

### 6.2.1 Proposed Corner Descriptor

We have also proposed a new corner descriptor which is not found in the literature previously. It is that if we consider a corner a triangle assuming a perfect angle of 90 degree than we can calculate the area of that triangle. This area can be used for matching the image data base with the other images. The variations in area with the change in the angle of corner can be made and thus two areas can be compared within 10 percentage of their resemblance.

## 6.3 Ribbon Structure Representation.

The detection of ribbons is an important step towards the ultimate goal of image matching using the graph-matching algorithm. The ribbon structure required by the matching algorithm mainly uses descriptors of the line structure because a ribbon is defined as parallel lines, whose edge gradients differ by $180^{o}$, are in the same relative position and at least some of the portion of lines overlaps each other. The descriptors used for ribbon structure are:-

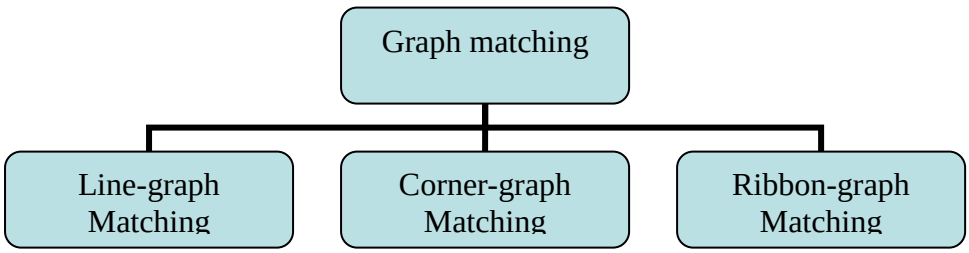
- Width of a Ribbon.
- Line Slope.
- Edge Gradient.
- Type of Ribbon.

### 6.3.1 Proposed Ribbon Descriptor

We have proposed a new ribbon descriptor as well. The two lines of ribbon enclose certain area in it. We have proposed that the area between two overlapping lines of the ribbon can be used for the image matching purpose. This area can be matched with the other image data with the certain tolerance e.g. 10% plus minus. It is expected that this matching would be quite robust as area enclosed by a ribbon is usually same so it will provide a much better authenticated performance than other descriptors.

## 6.4 Graph matching technique

In most of the image retrieval algorithms, researchers have been using only one of these image features in isolation such as lines and corners with only two attributes. For example line is used with two of its attributes, line length and gradient for sensitivity analysis for the problem of recognizing line patterns from large structural libraries. For object based image retrieval a single feature (corner) with two attributes, location and colour histogram at the centre of mass is used. The algorithm which we have studied and which is very successful in graph matching is designed by Naveed. This graph-matching algorithm which will be using three structures, (1) lines with four attributes (2) corners with five attributes and (3) ribbons with six attributes. Although graph matching is comparatively slower than other techniques used for matching, at the same time the accuracy in graph matching is superior and with such a large number of feature points, their attributes and relationships graph matching is still robust. The matching algorithm is divided into three routines; each routine is used for matching one of the features.



# 7. Graphical User Interface

## 7.1 How to Use

Figure 7-1 is the main window showing all the options to run our algorithm and to extract different feature according to the configuration selected. Menu bar contains File, Filter, View, Window and Help. Through this window we can convert color image into grey level image. We can also configure, detect and extract the features from the image and can use for different applications.

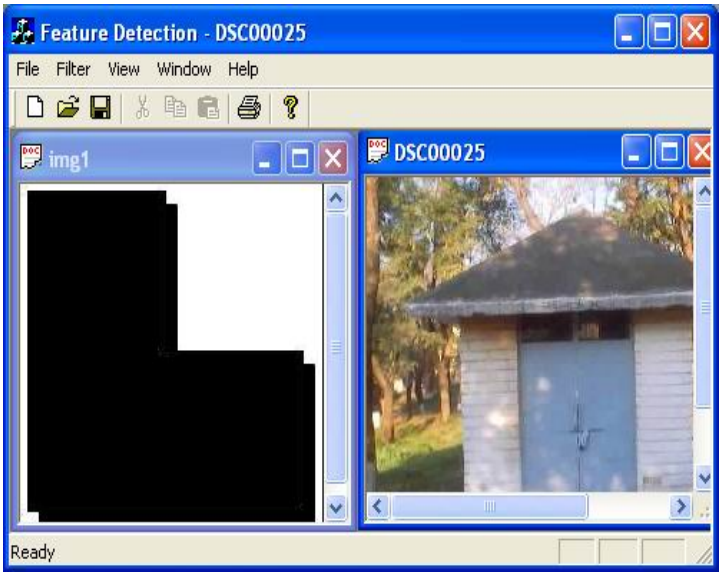



Figure 7-1

Figure 7-2 and Figure 7-3 represent the windows to open up any image file for input purpose.

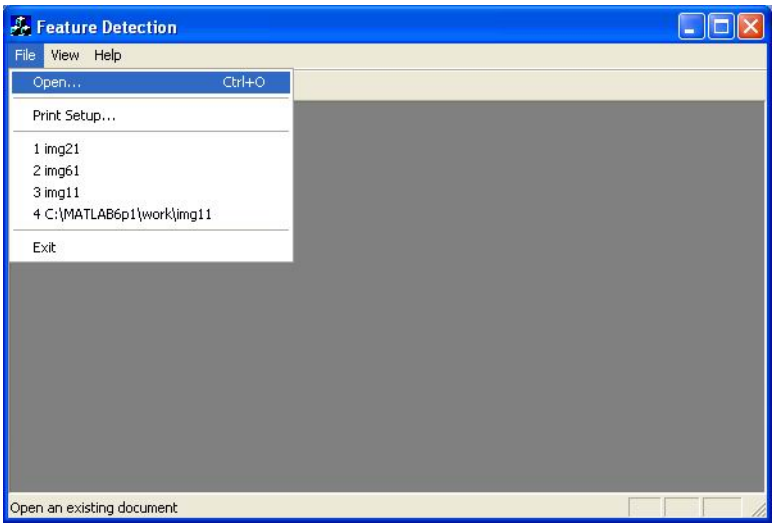



Figure 7-2

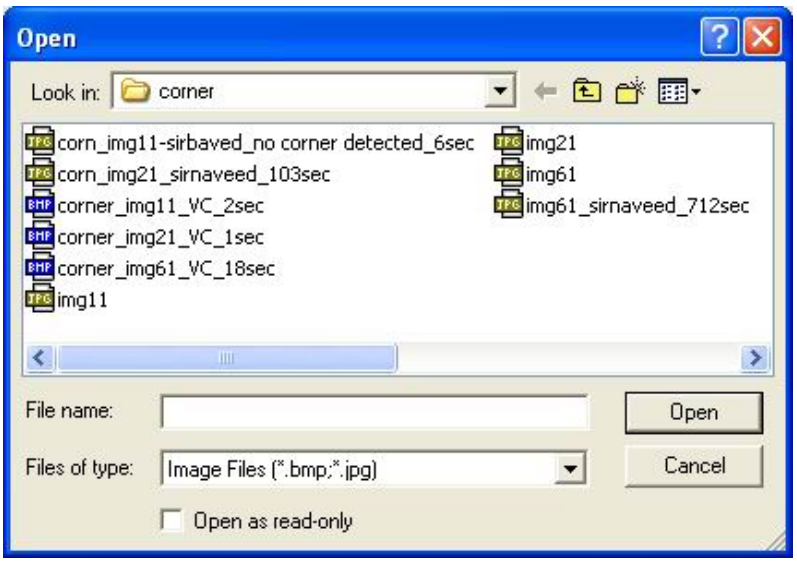



Figure 7-3

Figure 7-4 represents the window for conversion of any color image into grey scale if it is already colored, but as we have already a grey scale image of a monitor so there is no change

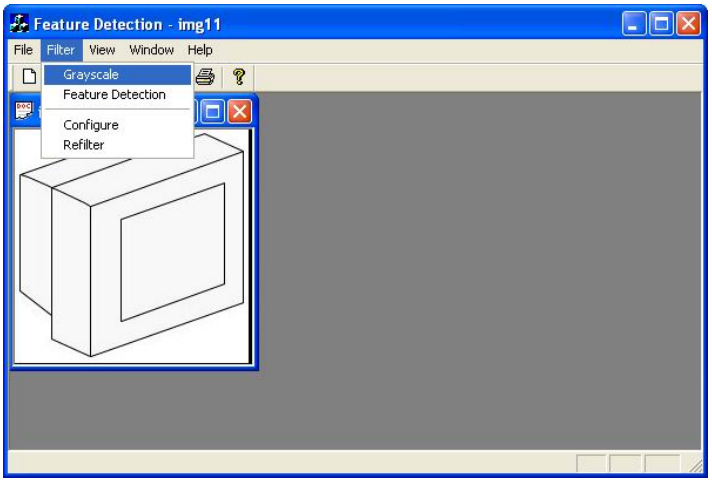



Figure 7-4

Figure 7-5 represents a window used for feature detection .It will detect all the image features as shown in Figure 7-6

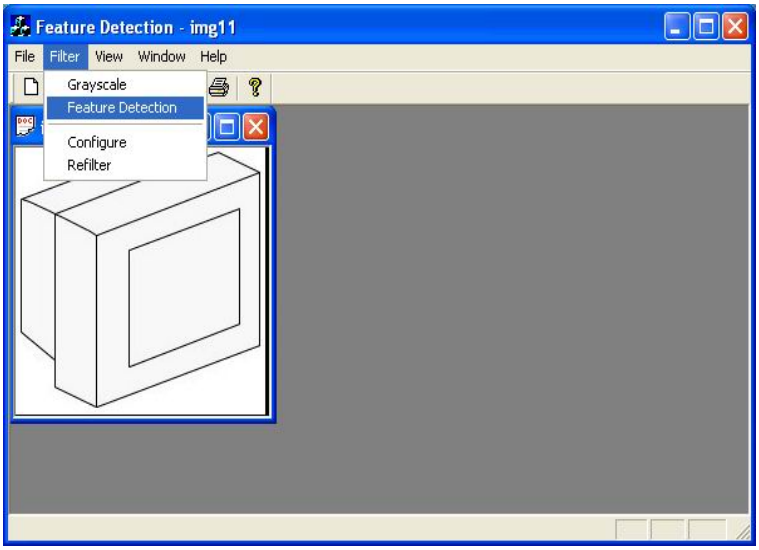



Figure 7-5

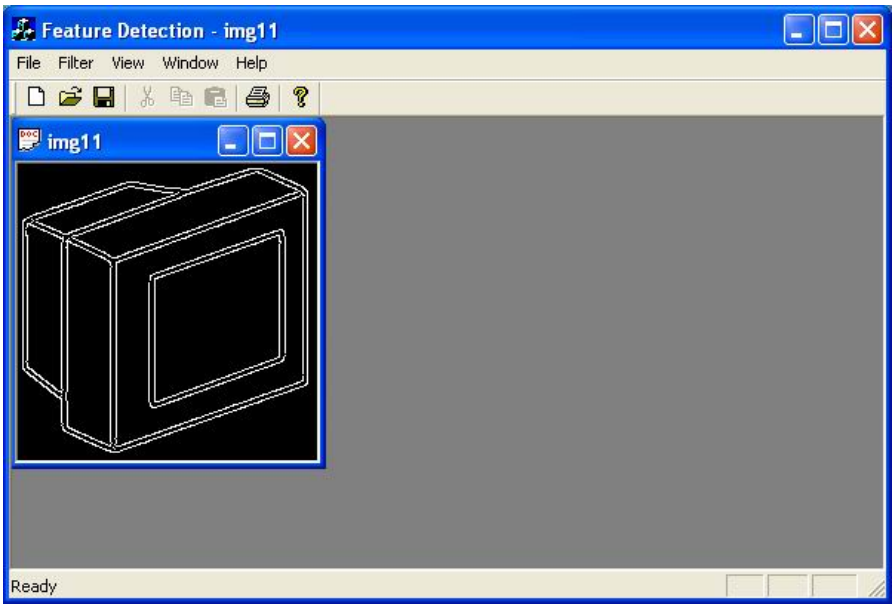



Fig 7-6

Figure 7-7 represents the window which is used for configuring the extraction technique.

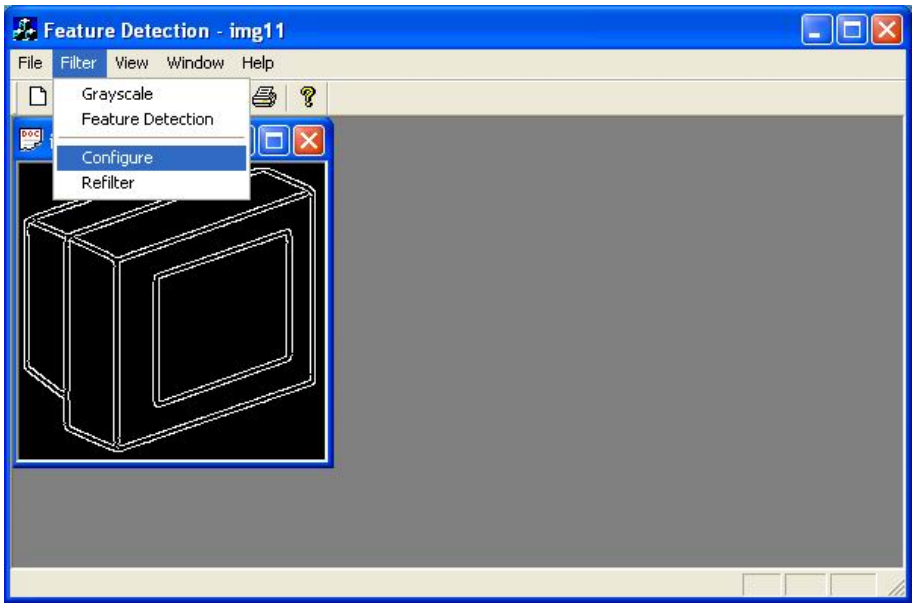



Figure 7-7

After selecting the configure icon the next window will be as shown in Figure7-8. In this window we have different options available with different smoothing levels and with different threshold level.

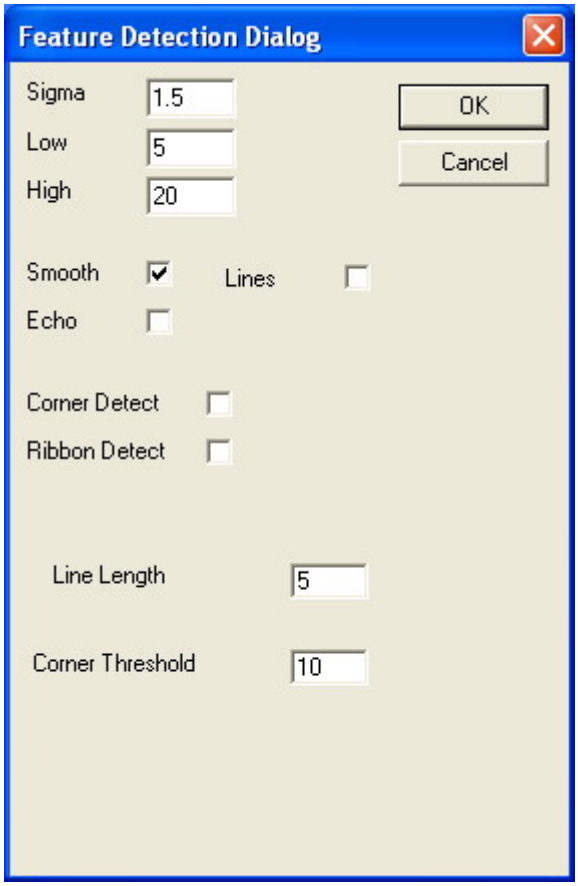



Figure 7-8

Figure 7-9 represents the window which is used for configuration for line detecting purpose. In this window we can configure different values as per our own choice. Similarly Figure 7-10 is showing the line detection result for the input image.

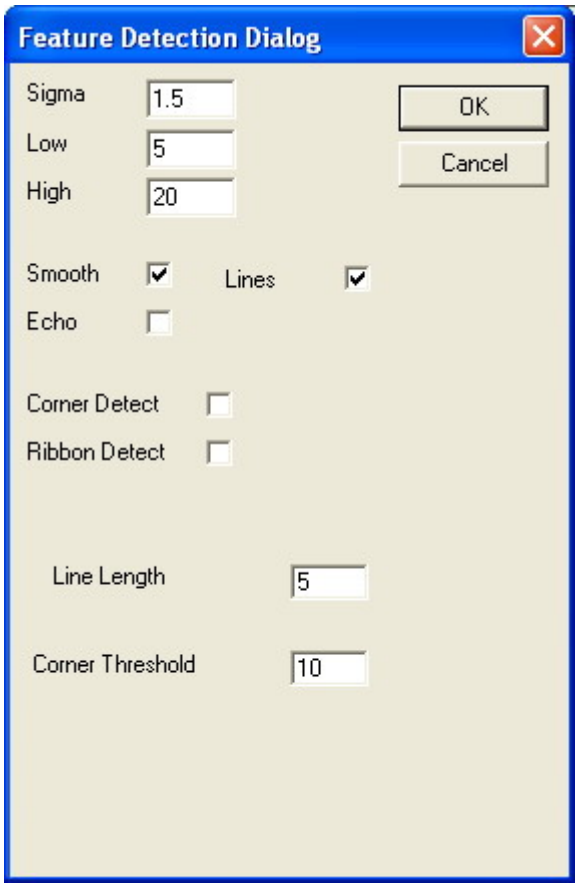



Figure 7-9

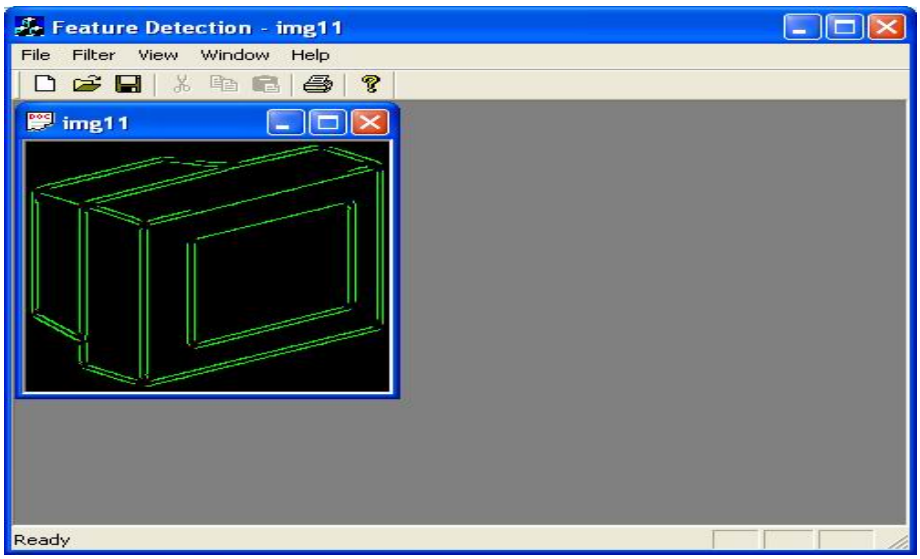



Figure 7-10

Similarly Figure 7-11 shows the window which is used for configuring for corner detection. For corner detection the corner box has to check while keeping the line box on check position. Where as Figure 7-12 shows the result of corner detection.

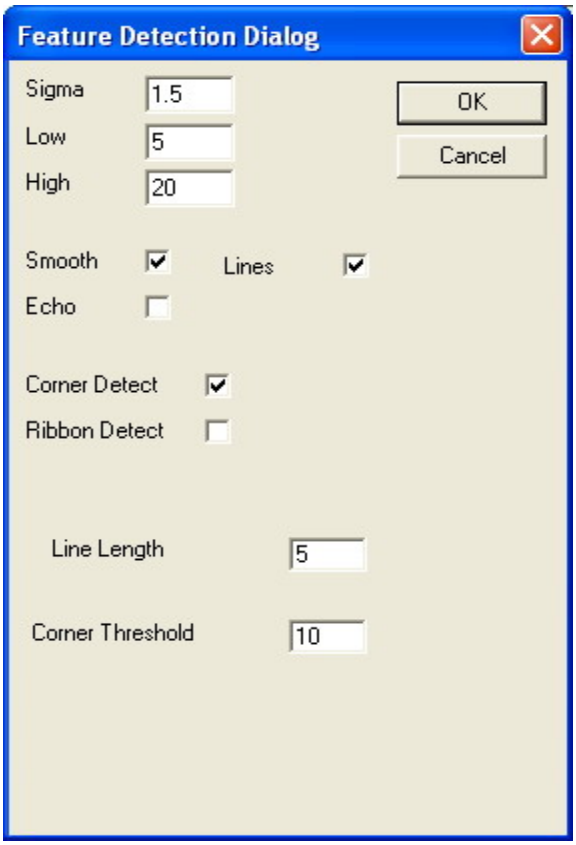



Figure 7-11

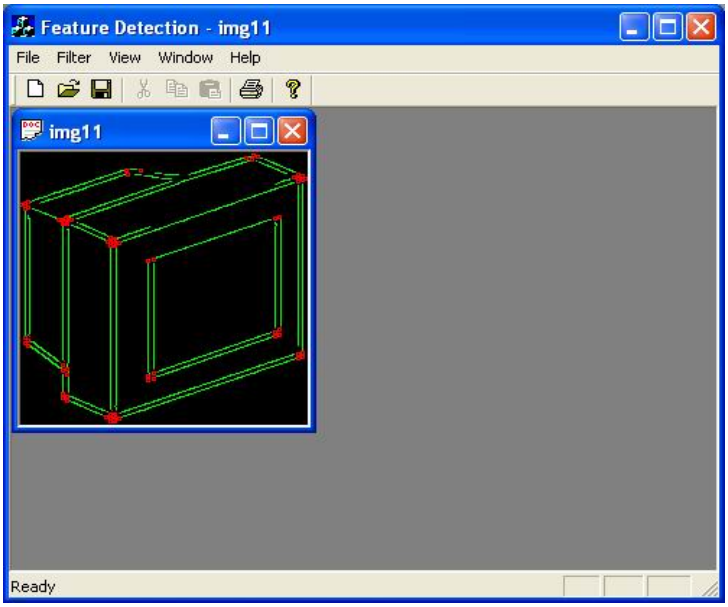



Figure 7-12

For Ribbon detection purpose the configuring window will give the option as shown in Figure 7-13

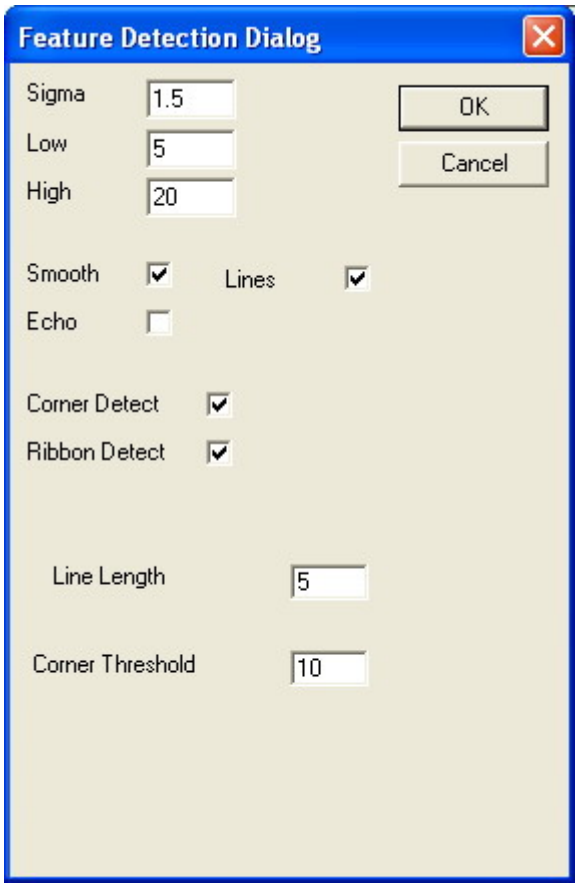



Figure 7-13

Fig 7.14 shows the ribbon detection results.

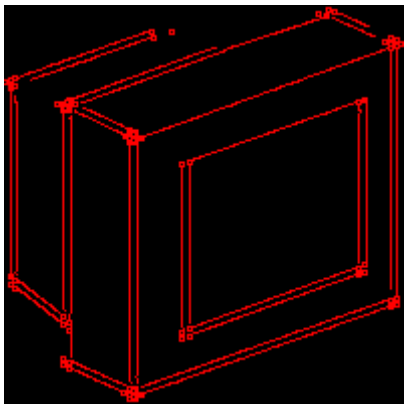


Fig 7.14

# 8. Discussion and Future Work

## 8.1 Analysis

Our main area of interest was to achieve image extraction, detection and retrieval Although it was assumed that the performance of the proposed image retrieval algorithm would solely depend on the feature detection process, meaning that feature extraction would be of vital importance, so we gave our maximum intentions towards this field. In our project maximum we have achieved and less in left which is regarding graph matching and image retrieval. Although it was our aim to develop and improve the work related to this field but a large number of techniques are already available and time available to us was short too. In our project we modified the existing methodology of Naveed and converted his work into VC and found that results are more accurate with speedy processing. We also added different levels of noise to some of the images, it negatively affected the feature extraction results but still correct images features were extracted. To further test our algorithms and the effect of applying smoothing factors we used the different value of smoothing parameters and acquired the desired results.

## 8.2 Future Work

Although all the desired goals of this work have not been met, several new issues are noticed, which can be dealt with in future research e.g. the area of the corner and the area of the ribbon for Image representation purpose. Since the work done by Naveed is very advanced and new to us as image processing was not taught in degree course. So lot of time was consumed to under stand the subject as a result the work related to image matching and retrieval is left, and we expect that new researchers can carry this work. It is recommended that subject related to this field must be taught at under grade levels due to its emerging importance.

## 8.3 Conclusions

Content Based Image Retrieval (CBIR) is an important problem for computer vision and engineering sciences. This project proposes a new geometric method for a Content Based Image Retrieval system for image identification and retrieval. Edges, lines, corners and ribbons are extracted by using algorithms in VC++ to reduce the processing time in order to ensure its practical implementation. We have shown the performance of our algorithms against Naveed's and matlab functions and verified the improved results. Our software extracts the line, Corner and ribbons information and stores them in text files in the same path where pictures are stored. These are the important features used for matching. We expect that our work will inspire the researchers and encourage them to excel in this field.