IMAGE SEGMENTATION USING HOUGH TRASFORM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS & INSTRUMENTATION ENGINEERING

BY

RABINDRA KUMAR MURMU (10507010**)**

MEENA JHANIYA (10507012**)**

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

ROURKELA-769008

2009

IMAGE SEGMENTATION USING HOUGH TRASFORM

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS & INSTRUMENTATION ENGINEERING

BY

RABINDRA KUMAR MURMU (10507010**)**

MEENA JHANIYA (10507012**)**

under the guidance of

Prof. U. C. PATI

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

ROURKELA-769008

2009

National Institute of Technology, Rourkela

CERTIFICATE

This is to certify that the thesis entitled "**IMAGE SEGMENTATION USING HOUGH TRASFORM"** submitted by Sri Rabindra Kumar Murmu and Miss Meena Jhaniya in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Electronics & Instrumentation Engineering at the National Institute of Technology, Rourkela (Deemed University) is an authentic work carried out by them under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University/Institute for the award of any Degree or Diploma.

Date:

Prof. U. C. Pati
Dept. of Electronics & Instrumentation Engineering
National Institute of Technology
Rourkela, Orissa-769008

ACKNOWLEDGEMENT

I wish to express my profound gratitude and indebtedness to **Prof. U. C. Pati** Department of Electronics & Instrumentation Engineering, N.I.T Rourkela and, Department of Electronics & Communication Engineering, N.I.T. Rourkela for introducing the present topic and for their inspiring guidance, constructive criticism and valuable suggestion throughout this project work.

I am also thankful to **Mr. S. K. Patra** and other staffs in Department of Electronics & Communication Engineering for providing all joyful environments in the lab and helping me out in different ways.

Last but not least, my sincere thanks to all our friends who have patiently extended all sorts of help for accomplishing this undertaking.

DATE:

RABINDRA KUMAR MURMU(10507010**)**

MEENA JHANIYA(10507012**)**

Dept. of Electronics & Instrumentation Engineering

National Institute of Technology
Rourkela – 769008

ABSTRACT

Image segmentation, a way of extracting and representing information from an image is to group pixels together into regions of similarity. We would group pixels together according to the rate of change of their intensity over a region or the rate of change of depth in the image, corresponding to pixels lying on the same surface such as a plane, cylinder, sphere etc.

Hough has proposed an interesting and computationally efficient procedure for detecting lines in pictures. In this paper we point out that the use of angle- radius rather than s lope-intercept parameters simplifies the computation further. We also show how the method can be used for circle detection.

The objective of this application is the recognition of different shapes in an image. This task can be subdivided into following procedures. First, an image is converted into gray scale. For detecting lines in images, we used histogram for the intensity related information and threshold. Then, an edge recognition procedure is implemented. We use the first-derivative Sobel detector to determine the edges and edge directions. Then, a Hough transform is accomplished on the threshold edge map for linking the edges. This is the procedure to detect straight line. For circle detection we wrote a function called “houghcircle”.

CONTENTS

List of figures

Chapter 1

INTRODUCTION

INTRODUCTION

Natural images consist of an overwhelming number of visual patterns generated by very diverse stochastic processes in nature. The objective of image understanding is to parse an input image into its constituent patterns. Depending on the type of patterns that a task is interested in, the parsing problem is called respectively

1) Image segmentation --- for homogeneous grey/color/texture region processes.

2) Perceptual grouping --- for point, curve, and general graph processes

3) Object recognition --- for text and objects.

1.1 SEGMENTATION

In computer vision, **segmentation** refers to the process of partitioning a digital image into multiple regions (sets of pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images.

The result of image segmentation is a set of regions that collectively cover the entire image, or a set of contours extracted from the image. Each of the pixels in a region is similar with respect to some characteristic or computed property, such as colour, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristic(s).

Segmentation algorithms generally are based on one of 2 basis properties of intensity values

Discontinuity : to partition an image based on abrupt changes in intensity (such as edges)

Similarity : to partition an image into regions that is similar according to a set of predefined criteria.

For intensity images (i.e. those represented by point-wise intensity levels) four popular approaches are: threshold techniques, edge-based methods, region-based techniques, and connectivity-preserving relaxation methods.

Threshold techniques, which make decisions based on local pixel information, are effective when the intensity levels of the objects fall squarely outside the range of levels in the background. Because spatial information is ignored, however, blurred region boundaries can create havoc. Edge-based methods center around contour detection: their weakness in connecting together broken contour lines make them, too, prone to failure in the presence of blurring.

A region-based method usually proceeds as follows: the image is partitioned into connected regions by grouping neighboring pixels of similar intensity levels. Adjacent regions are then merged under some criterion involving perhaps homogeneity or sharpness of region boundaries. Over stringent criteria create fragmentation; lenient ones overlook blurred boundaries and over merge. Hybrid techniques using a mix of the methods above are also popular.

A connectivity-preserving relaxation-based segmentation method, usually referred to as the *active contour model*, was proposed recently. The main idea is to start with some initial boundary shape represented in the form of spline curves, and iteratively modifies it by applying various shrink/expansion operations according to some energy function. Although the energy-minimizing model is not new, coupling it with the maintenance of an ``elastic" contour model gives it an interesting new twist. As usual with such methods, getting trapped into a local minimum is a risk against which one must guard; this is no easy task.

1.1.1 Application of Image Segmentation:

- Identifying objects in a scene for object-based measurements such as size and shape
- Identifying objects in a moving scene for *object-based video compression (MPEG4)*
- Locate tumors and other pathologies
- Measure tissue volumes
- Computer-guided surgery
- Diagnosis
- Treatment planning
- Study of anatomical structure
- Locate objects in satellite images (roads, forests, etc.)
- Face recognition

- Fingerprint recognition
- Traffic control systems
- Brake light detection
- Machine vision

1.2 EDGE:

An edge is seen at a place where an image has a strong intensity contrast. Edges could also be represented by a difference in color, without any difference in intensity. Detecting such edges goes beyond the scope of this introduction.

Of course there are exceptions where a strong intensity contrast does not embody an edge. Therefore a zero crossing detector is also thought of as a feature detector rather than a specific edge detector.

1.2.1 Edge Detector:

Edge detection is a very important area in the field of Computer Vision. Edges define the boundaries between regions in an image, which helps with segmentation and object recognition. They can show where shadows fall in an image or any other distinct change in the intensity of an image. Edge detection is a fundamental of low-level image processing and good edges are necessary for higher level processing.

The problem is that in general edge detectors behave very poorly. While their behavior may fall within tolerances in specific situations, in general edge detectors have difficulty adapting to different situations. The quality of edge detection is highly dependent on lighting conditions, the presence of objects of similar intensities, density of edges in the scene, and noise. While each of these problems can be handled by adjusting certain values in the edge detector and changing the threshold value for what is considered an edge, no good method has been determined for automatically setting these values, so they must be manually changed by an operator each time the detector is run with a different set of data. Since different edge detectors work better under different conditions, it would be ideal to have an algorithm that makes use of multiple edge detectors, applying each one when the scene conditions are most ideal for its method of detection. In order to create this system, you must first know which edge detectors perform better under which conditions. That is the goal of our project. We tested four edge detectors that use different methods for detecting edges and compared their results under a variety of situations to determine which detector was preferable under different sets of conditions. This data could then be used to create a multi-edge-detector system, which analyzes the scene and runs the edge detector best suited for the current set of data. For one of the edge detectors we considered two different ways of implementation, one using intensity only and the other using color information.

1.3 OBJECTIVE:

The basic aim of the project is to detect different shapes like triangle, rectangle, rhombus and circle etc. in an image using hough transform.

1.4 LAYOUT:

This project report consists of a total of five chapters each dealing with a specific part. Chapter 1 gives the general outline of the problem, aim and specific objectives of the study, and methodology adopted. Chapter 2 gives a thorough review of the available literature collected over the period. Chapter 3 consists of program we used in the experiment. After that the results obtained in the corresponding experiments are discussed in the chapter 4. At last the important conclusions drawn from the various results are discussed in the last chapter i.e chapter 5

Chapter 2

LITERATURE REVIEW

LITERATURE REVIEW

2.1 HISTOGRAM

The histogram of a digital image with gray levels in the range [0, L-1] is a discrete function h (r_k) = $n_{k,}$ where r_k is the kth gray level and n_k is the number of pixel in the image having gray level r_k. It is common practice to normalize a histogram by dividing each of its values by the total number of pixel in the image, denoted by n. Thus, a normalize histogram is given by p (r_k) = n_k/n, for k = 0, 1……, L-1. Loosely speaking, p(r_k) gives an estimate of probability of occurrence of gray level r_k. Note that the sum all components of a normalized histogram is equal to 1.

Histograms are the basic for numerous spatial domains processing technique. Histogram manipulation can be used effectively for image enhancement. In addition to providing useful image statistics, the information inherent in histograms is quite useful in image compression and segmentation. Histograms are simple to calculate in software and also lent themselves to economic hardware implementations, thus them a popular tool for real- time image processing. We can consider the histograms of our images.For the noise free image, it’s simply two spikes at i=100, i=150. For the low noise image, there are two clear peaks centred on i=100,i=150. For the high noise image, there is a single peak – two grey level populations corresponding to object and background have merged. We can define the input image *signal-to-noise ratio* in terms of the mean grey level value of the object pixels and background pixels and the additive noise standard deviation

$$S / N = \frac{|\mu_b - \mu_o|}{\sigma} \quad \text{--} \quad 2.1$$

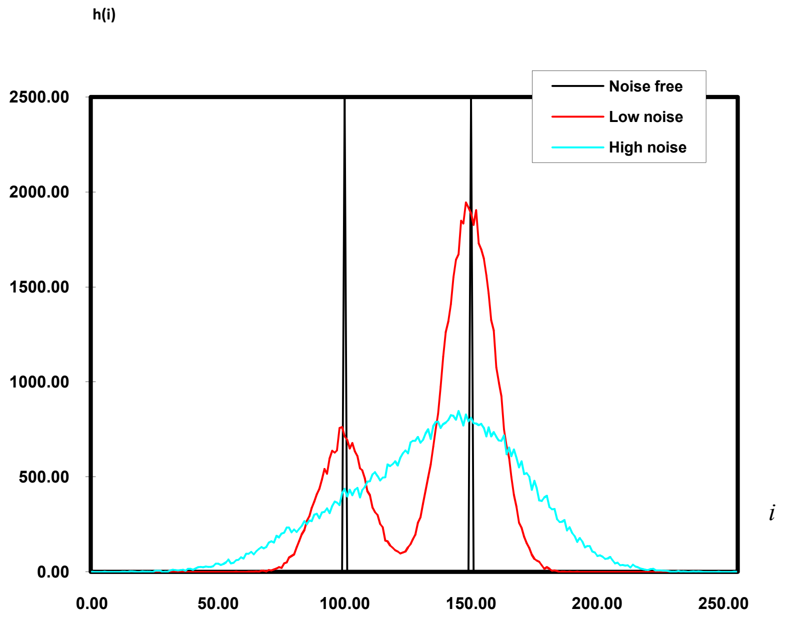


Fig 2.1 Signal to noise ratio of a signal

2.1.1 THRESHOLDING:

Thresholding is the simplest method of image segmentation. During the thresholding process, individual pixels in an image are marked as “object” pixels if their value is greater than some threshold value (assuming an object to be brighter than the background) and as “background” pixels otherwise. This convention is known as threshold above. Variants include threshold below, which is opposite of threshold above; threshold inside, where a pixel is labeled "object" if its value is between two thresholds; and threshold outside, which is the opposite of threshold inside. Typically, an object pixel is given a value of “1” while a background pixel is given a value of “0.” Finally, a binary image is created by coloring each pixel white or black, depending on a pixel's label.

We can easily understand segmentation based on thresholding by looking at the histogram of the low noise object/background image. There is a clear ‘valley’ between to two peaks. We can define the grey level thresholding algorithm as follows:

If the grey level of pixel p <=T then pixel p is an object pixel

Else

Pixel p is a background pixel

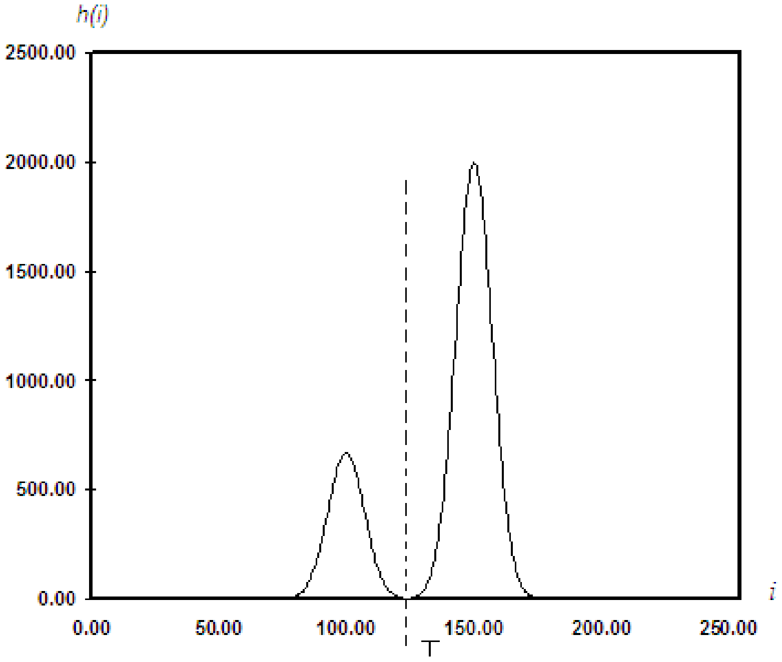


Fig 2.2 Thresholding of a signal

2.2 EDGE DETECTION TECHNIQUES

There are many ways to perform edge detection. However, the majority of different methods may be grouped into two categories:

- **Gradient:** The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image.
- **Laplacian:** The Laplacian method searches for zero crossings in the second derivative of the image to find edges. An edge has the one-dimensional shape of a ramp and calculating the derivative of the image can highlight its location. Suppose we have the following signal, with an edge shown by the jump in intensity below:

Suppose we have the following signal, with an edge shown by the jump in intensity below:

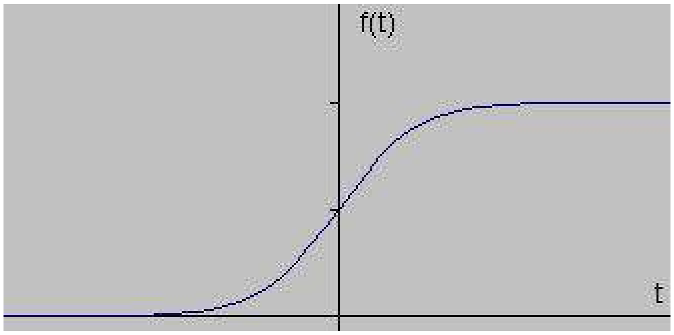


Fig 2.3 Intensity graph of a signal

If we take the gradient of this signal (which, in one dimension, is just the first derivative with respect to t) we get the following:

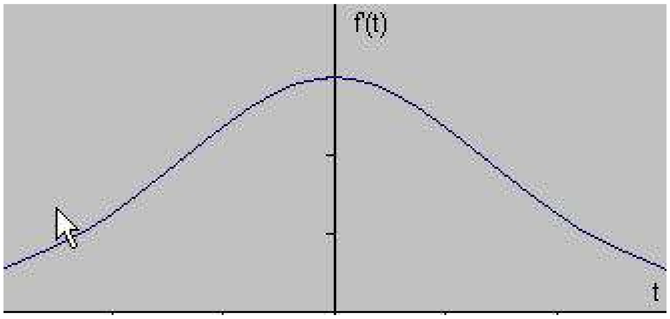


Fig 2.4 First derivative of the signal

Clearly, the derivative shows a maximum located at the center of the edge in the original signal. This method of locating an edge is characteristic of the “gradient filter” family of edge detection filters and includes the Sobel method. A pixel location is declared an edge location if the value of the gradient exceeds some threshold. As mentioned before, edges will have higher pixel intensity values than those surrounding it. So once a threshold is set, you can compare the gradient value to the threshold value and detect an edge whenever the threshold is exceeded. Furthermore, when the first derivative is at a maximum, the second derivative is zero. As a result, another alternative to finding the location of an edge is to locate the zeros in the second derivative. This method is known as the Laplacian and the second derivative of the signal is shown below:

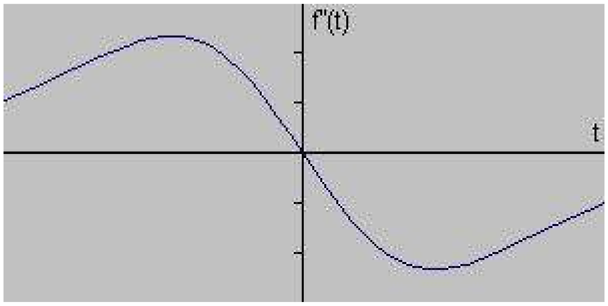


Fig 2.5 Second derivative of the signal

2.2.1 SOBEL OPERATOR :

The Sobel operator is used in image processing, particularly within edge detection algorithms. Technically, it is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. At each point in the image, the result of the Sobel operator is either the corresponding gradient vector or the norm of this vector. The Sobel operator is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical direction and is therefore relatively inexpensive in terms of computations. On the other hand, the gradient approximation which it produces is relatively crude, in particular for high frequency variations in the image.

In simple terms, the operator calculates the *gradient* of the image intensity at each point, giving the direction of the largest possible increase from light to dark and the rate of change in that direction. The result therefore shows how "abruptly" or "smoothly" the image changes at that point and therefore how likely it is that that part of the image represents an *edge*, as well as how that edge is likely to be oriented. In practice, the magnitude (likelihood of an edge) calculation is more reliable and easier to interpret than the direction calculation.

Mathematically, the gradient of a two-variable function (here the image intensity function) is at each image point a 2D vector with the components given by the derivatives in the horizontal and vertical directions. At each image point, the gradient vector points in the direction of largest possible intensity increase, and the length of the gradient vector corresponds to the rate of change in that direction. This implies that the result of the Sobel operator at an image point which is in a region of constant image intensity is a zero vector and at a point on an edge is a vector which points across the edge, from darker to brighter values.

The Sobel Edge Detector uses a simple *convolution kernel* to create a series of *gradient magnitudes*. For those you of mathematically inclined, applying convolution *K* to pixel group *p* can be represented as:

$$N(x,y) = \sum_{k=-1}^{1} \sum_{j=-1}^{1} K(j,k)p(x-j,y-k) \text{ -------}\mathbf{2.2}$$

So the Sobel Edge Detector uses two convolution kernels, one to detect changes in vertical contrast (h_x) and another to detect horizontal contrast (h_y).

$$h_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad h_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Fig 2.6 Sobel edge detector mask

The amazing thing is is that this data can now be represented as a vector (gradient vector). The two gradients computed using h_x and h_y can be regarded as the x and y components of the vector. Therefore we have a gradient magnitude and direction:

$$\mathbf{g} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} \qquad \text{---}\mathbf{2.3}$$

$g = \sqrt{g_x^2 + g_y^2}$ ---2.4

$\theta = \tan^{-1}\left(\frac{g_y}{g_x}\right)$ ---2.5

Where **g** is the gradient vector, g is the gradient magnitude and θ is the gradient direction.

2.2.2 ROBERT'S CROSS OPERATOR:

The Roberts Cross operator performs a simple, quick to compute, 2-D spatial gradient measurement on an image. Pixel values at each point in the output represent the estimated absolute magnitude of the spatial gradient of the input image at that point.

The operator consists of a pair of 2×2 convolution kernels as shown in Figure. One kernel is simply the other rotated by 90°. This is very similar to the Sobel operator.

+1	0
0	-1

Gx

0	+1
-1	0

Gy

Fig 2.7 Roberts edge detector mask

These kernels are designed to respond maximally to edges running at 45° to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (call these *Gx* and *Gy*). These can then be224 combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient.

The gradient magnitude is given by:

$$|G| = \sqrt{Gx^2 + Gy^2} \tag{2.6}$$

although typically, an approximate magnitude is computed using:

$$|G| = |Gx| + |Gy| \tag{2.7}$$

which is much faster to compute.

The angle of orientation of the edge giving rise to the spatial gradient (relative to the pixel grid orientation) is given by:

$$\theta = \arctan(Gy/Gx) - 3\pi/4 \tag{2.8}$$

2.2.3 PREWITT'S OPERATOR:

Prewitt operator is similar to the Sobel operator and is used for detecting vertical and horizontal edges in images.

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad h_3 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Fig 2.8 Priwitt edge detector mask

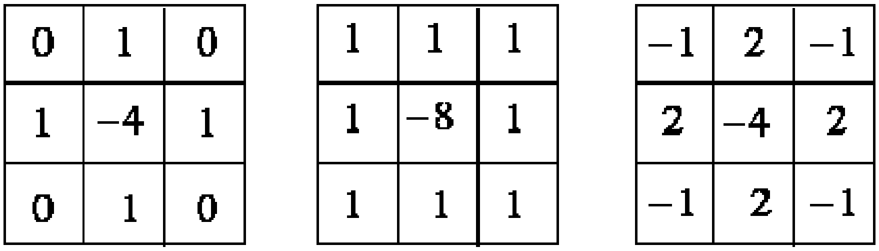
2.2.4 LAPLACIAN OF GAUSSIAN:

The Laplacian is a 2-D isotropic measure of the 2nd spatial derivative of an image. The Laplacian of an image highlights regions of rapid intensity change and is therefore often used for edge detection. The Laplacian is often applied to an image that has first been smoothed with something approximating a Gaussian Smoothing filter in order to reduce its sensitivity to noise. The operator normally takes a single gray level image as input and produces another gray level image as output.

The Laplacian *L(x,y)* of an image with pixel intensity values *I(x,y)* is given by:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \quad \text{---2.9}$$

Since the input image is represented as a set of discrete pixels, we have to find a discrete convolution kernel that can approximate the second derivatives in the definition of the Laplacian. Three commonly used small kernels are shown in Figure2.9.



0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

−1	2	−1
2	−4	2
−1	2	−1

Fig 2.9 Laplacian edge detector mask

Because these kernels are approximating a second derivative measurement on the image, they are very sensitive to noise. To counter this, the image is often Gaussian Smoothed before applying the Laplacian filter. This pre-processing step reduces the high frequency noise components prior to the differentiation step.

In fact, since the convolution operation is associative, we can convolve the Gaussian smoothing filter with the Laplacian filter first of all, and then convolve this hybrid filter with the image to achieve the required result. Doing things this way has two advantages:

Since both the Gaussian and the Laplacian kernels are usually much smaller than the image, this method usually requires far fewer arithmetic operations.

The LoG (`Laplacian of Gaussian') kernel can be precalculated in advance so only one convolution needs to be performed at run-time on the image.

The 2-D LoG function centered on zero and with Gaussian standard deviation σ has the form:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \quad 2.10$$

and is shown in Figure 2.10.

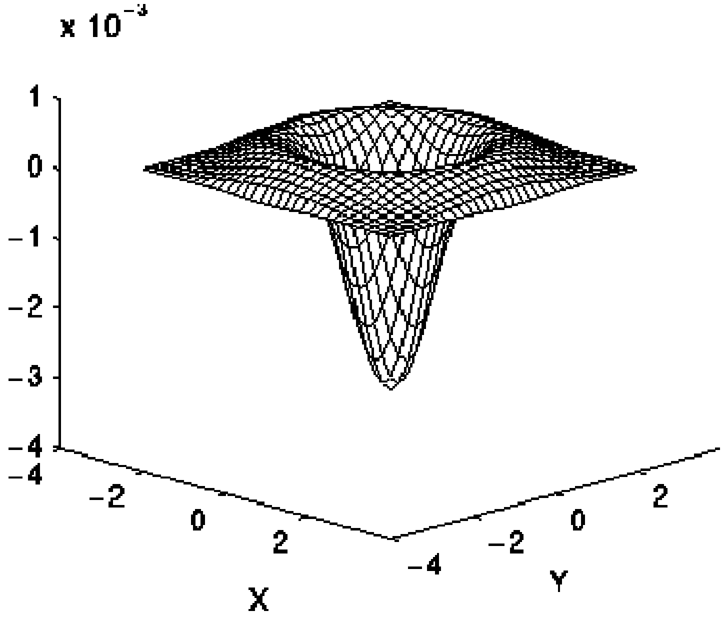
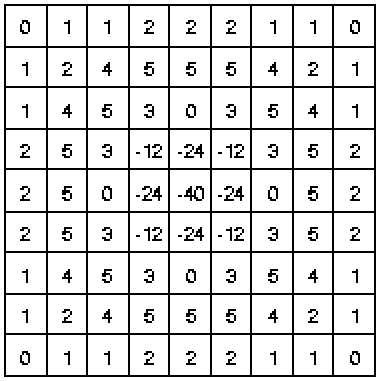


Fig 2.10 Gaussian Surface



0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

Fig 2.11 Discrete approximation to LoG function with Gaussian σ = 1.4

Note that as the Gaussian is made increasingly narrow, the LoG kernel becomes the same as the simple Laplacian kernels shown in Figure 2.11. This is because smoothing with a very narrow Gaussian (σ < 0.5 pixels) on a discrete grid has no effect. Hence on a discrete grid, the simple Laplacian can be seen as a limiting case of the LoG for narrow Gaussians.

2.2.5 CANNY'S EDGE DETECTOR:

The canny edge detector first smoothes the image to eliminate noise. It then finds the image gradient to highlight regions with high spatial derivatives. The algorithm then tracks along these regions and suppresses any pixel that is not at the maximum (non maximum suppression). The gradient array is now further reduced by hysteresis. Hysteresis is used to track along the remaining pixels that have not been suppressed. Hysteresis uses two thresholds and if the magnitude is below the first threshold, it is set to zero (made a non edge). If the magnitude is above the high threshold, it is made an edge. And if the magnitude is between the 2 thresholds, then it is set to zero unless there is a path from this pixel to a pixel with a gradient above second threshold.

2.3 HOUGH TRANSFORM:

In Hough transform, the points are linked by determining first if they lie on the curve of specified shape. Unlike the local analysis method, where given n points of an image are taken into consideration. Suppose we want to find the subset of these points that lie on the straight lines. One possible solution is to find all the lines determine by every pair of points and then find all subsets of points that are close to particular lines. The problem with this procedure is that it involves finding n (n-1)/2 ~ n^2 lines and then performing (n) (n (n-1)) ~ n^3 comparisons of every point to all lines. This approach is computationally prohibitive in all but in most the trivial applications.

The Hough transform is a method that, in theory, can be used to find features of any shape in an image. In practice it is only generally used for finding straight lines or circles. The computational complexity of the method grows rapidly with more complex shapes.

Assume we have some data points in an image which are perhaps the result of an edge detection process, or boundary points of a binary blob. We wish to recognize the points that form a straight line.

Consider a point (x_i, y_i) in the image. The general equation of a line is

$y = ax + b$. ---2.11

There are infinitely many lines that pass through this point, but they all satisfy the condition

$y_i = ax_i + b$-- 2.12

For varying a and b.

We can rewrite this equation as

$b = -x_i a + y_i$, --- 2.13

And plot the variation of a and b.

If we divide parameter space into a number of discrete accumulator cells we can collect `votes' in a b space from each data point in x y space. Peaks in a b space will mark the equations of lines of co-linear points in x y space.

However, we have a problem with using $y = ax + b$ to represent lines when the line is vertical. In that case $a = \infty$ and our parameter space is unbounded (we would need a very large computer to store our parameter accumulator array!)

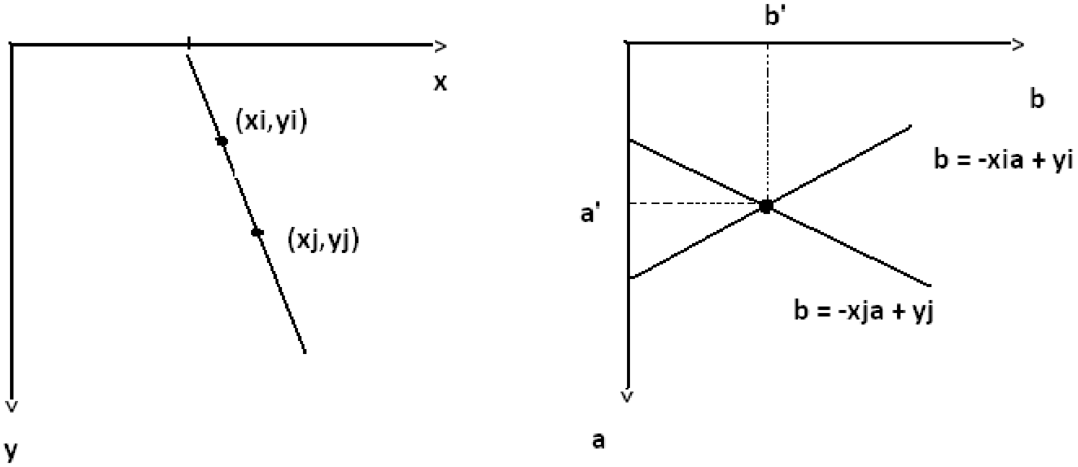


Fig 2.12 xy-plane and Parameter space

An alternative representation of a line is given by

$x\cos\theta \;- y\sin\theta = \rho$---2.14

Where *r* is the distance of the line from the origin and *theta* is the angle between this perpendicular and the x-axis. Our parameter space is now in θ and *r*, where $0 \leq \theta \leq 2\pi$

And *r* is limited by the size of the image.

As before, peaks in the accumulator array mark the equations of significant lines.

2.3.1 ACCUMULATOR:

Finding the dominant lines in the image can now be reformulated as finding all the locations in parameter space where significant number of lines intersect. This is basically the goal of Hough Transform. In order to compute the Hough Transform, we must decide on a discrete representation of the continuous parameter space by selecting an appropriate step size for the k and d axes. Once we have selected step sizes for the coordinates, we can represent the space naturally using an array. Since the lines intersect, it is called an accumulator array. Each parameter space line is painted into the accumulator array and the cells through which it passes through are incremented so that ultimately each cell accumulates the total number of lines that intersect at each cell(Fig. 2.13)

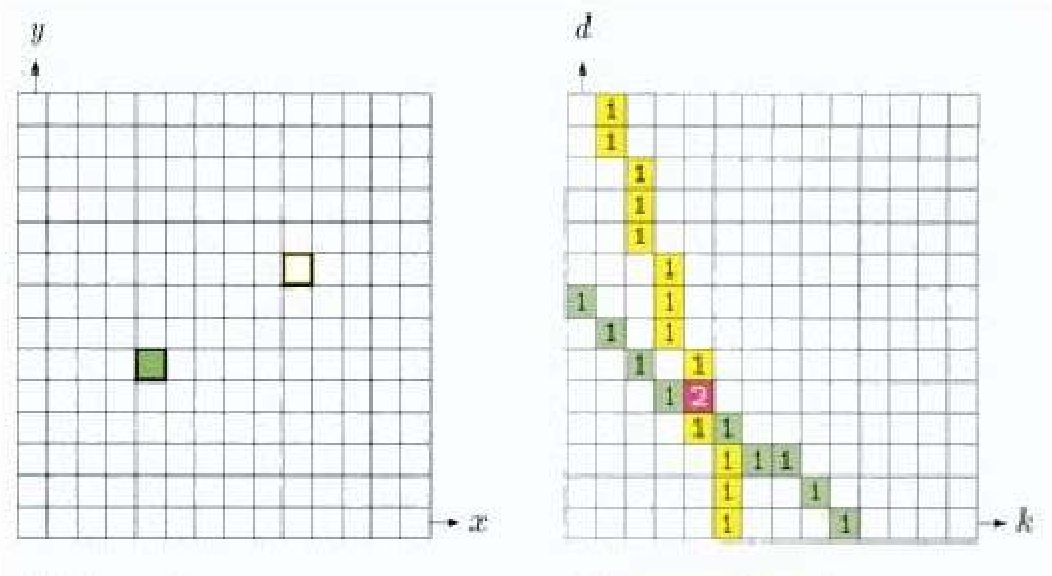


Fig 2.13 Image space and parameter space

2.3.2 BASIC ALGORITHMS:

- Compute the gradient of an image and threshold it to obtain a binary image.
- Specify subdivisions in the ρθ-plane.
- Examine the counts of the accumulator cells for high pixel concentrations.
- Examine the relationship (principally for continuity) between pixels in a chosen cell.
  - Based on computing the distance between disconnected pixels identified during traversal of the set of pixels corresponding to a given accumulator cell.
  - A gap at any point is significant if the distance between that point and its closet neighbor exceeds a certain threshold.

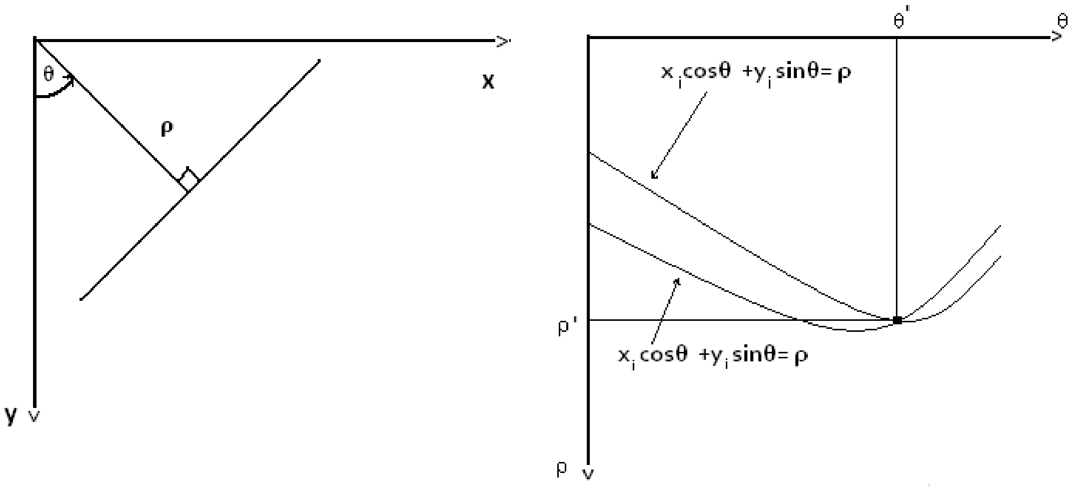

θ
x
ρ
y
θ'
θ
$x_i\cos\theta + y_i\sin\theta = \rho$
ρ'
$x_i\cos\theta + y_i\sin\theta = \rho$
ρ

Fig 2.14 (ρ,θ) parameterization of line in the xy-plane and Sinusoidal curve in the ρθ-plane.

2.4 CIRCULAR HOUGH TRANSFORM:

The general Hough transform can be used on any kind of shape although the complexity of the transformation increase with the number of parameters needed to describing the shape. In the following we will look at the Circular Hough transform (CHT) .

2.4.1 PARAMETEER REPRESENTATION:

The general Hough transform can be described as a transformation of a point in the x-y plane to the parameter space. The parameter space is defined according to the space of the object of interest.

The circle is simpler to represent in parameter space, compared to the line, since the parameter of the circle can be directly transfer to the parameter space.

The equation of the circle is :

$$r^2 = (x-a)^2 + (y-b)^2 \text{--}2.15$$

As it can be seen the circle to get three parameter r, a & b, where a & b are the centre of the circle in the direction x & y respectively and r is the radius .

The parameter representation of the circle is:

$$x=a + r\cos(\theta) \text{---} 2.16$$

$$y=b + r\sin(\theta) \text{---} 2.17$$

Thus the parameter space for a circle will belong to R^3 whereas the line only belonged to R^2.As the number of parameter needed to describe the shape increase as well as the dimension of the parameter space R increase so do the complexity of the Hough transform .There for simple shape with parameter belonging to R^2 or at most R^3 .In order to simplicity the parametric representation of the circle ,the radius can be held as a constant or limited number of known radii.

2.4.2 ACCUMULATOR:

The process of finding circles in the image using Circular Hough Transform is:

First we find all the edge in the image .This step has nothing to do with Hough transform and any edge detection technique can be used .It could be canny or sobel or prewitt. Here we used sobel edge detection technique.

At each edge point we draw a circle with centre in the point with the desired radius .This circle is drawn in the parameter space, such that our x-axis is the a- value and y-axis in the b-value and z-axis is the radii. At the coordinates which belongs to the parameter of the drawn circle .We increment the value in our accumulator matrix which essentially has same size as parameter space .In this way we sweep over energy edge point in the input image drawing circle with the desired circle with desired radii and incrementing the value in our accumulator. When every edge point and every desired radius is used ,we can turn our attention to accumulator will now contain numbers corresponding to the number of circles passing through the individual coordinate .Thus the highest number correspond to the circle of the circle in the image.

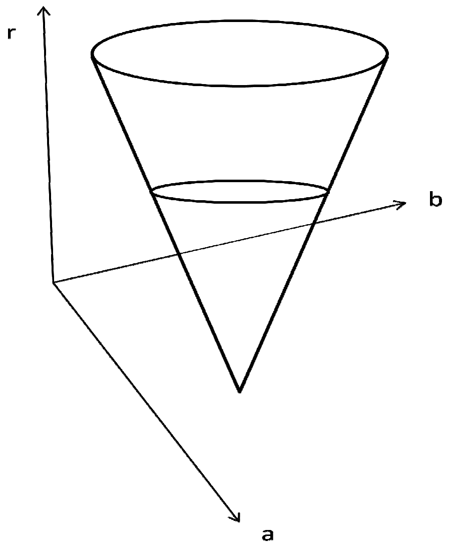


Fig 2.15 The parameter space used for CHT

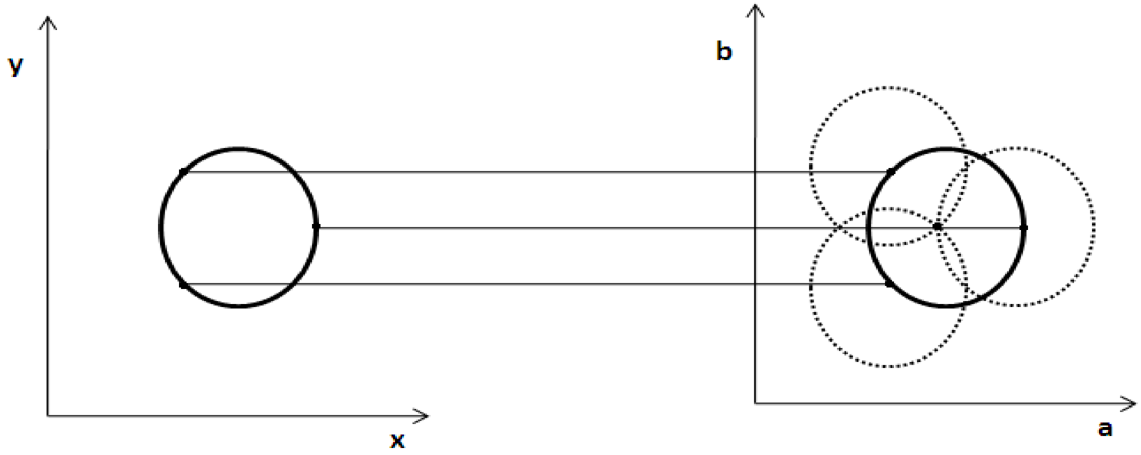


Fig 2.16 A Circular Hough transform from the x,y-space (left) to the parameterspace (right), this example is for a constant radius

2.5 ADVANTAGES:

One important difference between the Hugh Transform and other approaches is resistance of the former to noise in the image and its tolerance towards holes in the boundary line. Figures 1 and 2 compare the Hugh transform of a plain straight line with a dotted one. As can be seen there is almost no differences in the results.

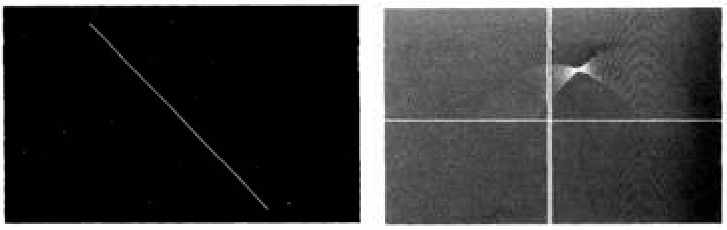


Fig 2.17 Straight line and its Hough transform.

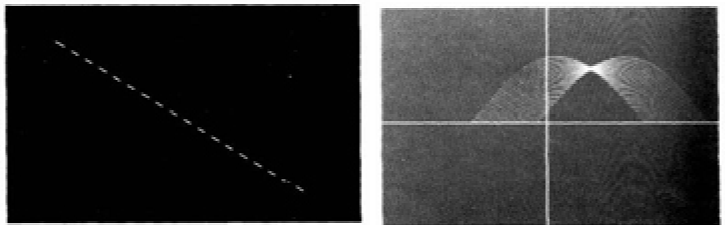


Fig 2.18 Straight dashed line and its Hough transform.

2.6 DISADVANTAGES:

Since the Hugh Transform is a kind of Brute-Force method it is very complex in computation. It has two main drawbacks: large memory requirement and slowness. In order to find the plane parameters accurately, parameter space must be divided finely in all three directions, and an accumulator assigned to each block. Also it takes a long time to fill the accumulators when there are so many. The Fast Hough Transform gives considerable speed up and reduces memory requirement. Instead of dividing parameter space uniformly into blocks, the FHT homes in on the solution, ignoring areas in parameter space relatively devoid of votes. The relative speed advantage of the FHT increases for higher dimensional parameter spaces. Another weakness of the Hough Transform is that it often recognizes many similar lines instead of the one correct one. The algorithm only returns a line without a starting and ending point. For that different post processing algorithms have to be used.

Chapter 3

METHODODOLOGY

METHODOLOGY

3.1 FOR LINE DETECTION

We are working on MATLAB environment. The method for detecting a line using hough transform is as follows.

- First take the image to read it.
- It has to convert to gray if the image is colour one to reduce the computational complexity.
- Draw the histogram of the image. It will give the information about the intensity.
- Edge detection

  Use any of the edge detector to detect the edge and adjust the parameter such that it will show all the edge in the image.
- The output image of the detector is binary one and it is not in continuous manner.
- So edge detection algorithm typically is followed by linking procedure to assemble edge pixel to meaningful edge.
- One of the linking method is Hough Transform which works on voting scheme.
- Hough Transform

  To implement the hough transform, **“hough”** function is used.

  Syntax:

  [h,theta,rho] = hough(f,dtheta,drho)

  Inputs:

  f is output image of the edge detector.

  dtheta specifies the spacing oh hough transform bins along the **theta** axis.

  drho specifies the spacing of the hough transform bins along the **rho** axis.

  drho and **dtheta** are the optional parameter.

  Outputs:

  This function converts space of representation from xy-plane to ρθ-plane.

  The output **h** is the Hough Transform matrix.

  It is **nrho-**by-**ntheta**

  Where nrho = 2*ceil(norm(size(f))/drho) -1,

  and ntheta = 2*ceil(90/dtheta)

theta is an **ntheta**-element vector containing the angle in degrees, corresponding to each column of **h**.

rho is **nrho**-element vector containing the value of **rho** corresponding to each row of **h**.

- All the collinear point in the xy-plane will intersect at a point in ρθ-plane.
- Hough Transform Peak Detection
    1. Find the Hough Transform cell containing the highest value and record its location.
    2. Suppress (set to zero) Hough Transform cells in the immediate neighborhood of the maximum found in step 1.
    3. Repeat until desired no of peaks has been found, or until specified threshold has been reached.

This procedure is implemented by **“houghpeaks”** function.

Syntax :

[r,c,hnew] = houghpeaks(h,numpeaks,threshold,nhood)

Inputs:

This function detects the peaks in the Hough Transform matrix **h**.

numpeaks specifies the maximum number of peak location to look for.

Value of **h** below **threshold** will not be considered to be peak.

nhood is a two-element vector specifying the size of suppression neighborhood. This is the neighborhood around each peak that isset to zero after the peak is identified.

Output:

r and **c** are the row and column coordinates of the identified peaks.

hnew is the Hough Transform with peak neighborhood suppressed.

- Hough Transform Line Detection And Linking

  Once a set of peaks has been identified in the Hough Transform, it remains to be determined if there are line segments associated with those peaks, as well as where they start and end. For each peak, the first step is to find the location of all nonzero pixels in the image that contributed to that peak

  For this purpose, we write function **“houghpixels”**

  Syntax:

  [r,c] = houghpixels(f,theta,rho,rbin,cbin)

  This function computes the row-column indices (**r,c**) for nonzero pixels in image **f** that maps to particular Hough Transform bin, (**rbin,cbin**). **rbin** and **cbin** are scalars indicating the row-column bin location in the Hough Transform matrix return function **hough. theta** and **rho** are the second and third out arguments from the **hough** function.

- The pixel associated with the locations found using **houghpixels** must be grouped into line segment. Function **houghlines uses the following strategy:**

  1. Rotate the pixel by 90° - Ө so that they lie approximately along a vertical line.
  2. Sort the pixel location by their rotated *x*-value.
  3. Use the function **diff** to locate gaps. Ignore small gaps; this has effect of merging adjacent line segment that are separated by a small space.
  4. Return information about line segment that are longer than minimum length threshold.

     Syntax:

     Lines = houghlines(f,theta,rho,rr,cc,fillgap,minlength)

     This function extract line segments in the image **f** associated with particular bins in a Hough Transform. **theta** and **rho** are vector return by the function **hough.** Vector **rr** and **cc** specifies the row and column of the Hough Transform bins to use in searching for line segment. If **houghlines** finds two line segment associated with the same Hough Transform bin that are separated by less than **fillgap pixel, houghline** merges them into single line segment.merged line segment less than **minlength** pixels along are discarded.

3.2 FOR CIRCLE DETECTION

We wrote a function called “houghcircle” .This function detects multiple circles in an image using Hough transform. The image contains disks whose centers can be in or out of the image.

Syntax:

circles = houghcircle(im, minR, maxR);

or

circles = houghcircle(im, minR, maxR, ratio);

or

circles = houghcircle(im, minR, maxR, ratio, delta);

Inputs:

im: input image (gray or color)

minR: minimal radius (unit: pixels)

maxR: maximal radius (unit: pixels)

ratio: minimal number of detected edge pixels of a circle to the circle perimeter (0<ratio<1, default: 0.3)

delta: the maximal difference between two circles for them to be considered as the same one (default: 12); e.g., circle1=[x1 y1 R1],circle2=[x2 y2 R2], delta = |x1-x2|+|y1-y2|+|R1-R2|.

Output:

circles: n-by-4 array of n circles; each circle is represented by [cx cy R c], where (cx cy), R, and c are the center coordinate, radius, and pixel count, respectively.

Steps:

- First of all we check the input parameter.
- If the number of input is ‘4’, in that case , delta=12, because Each element in [cx cy R] may deviate 4 pixels approx.
- If the number of input is ‘3’, then ratio=0.3 ,which is the 1/3 of the perimeter and delta=12.
- If the number of input is not ‘5’ ,at that time program will show a message that” Require at least 3 input arguments”
- If minR<0 or maxR<0 or minR>maxR or ratio<0 or ratio>1 or delta<0, the it will show this message

  'Required: 0<minR, 0<maxR, minR<=maxR, 0<ratio<1, and 0<delta'

- Turn the colour image into gray image.

- Create a 3D Hough array. The first two dimensions specify the coordinates of the circle centers (x,y), and the third specifies the radii(r).

- Detect edge pixels using Sobel edge detector. Reset the lower and/or upper thresholds to balance between the performance and detection quality.

- For an edge pixel (px,py), the locations of its corresponding possible circle centers are within the area (px-maxR:px+maxR, py-maxR:py+maxR).

- Create the grid [0:maxR2, 0:maxR2], and then compute the distances between the center and all the grid points to form a radius map (Rmap), followed by clearing out-of-range radii

  . Where : maxR2= 2 * max R.

- For each edge pixel, increment the corresponding elements in the Hough array.

- Collect circles

  Format: [cx cy R c]

  Clear pixel count < 2*pi*R*ratio

- Delete similar circles.

- Draw circles on the original image.

Chapter 4

RESULTS and DISCUSSIONS

4.1 COMPARISON OF DIFFERENT EDGE DETECTOR

We are working on MATLAB environment. Different type of edge detectors are there such as Sobel, Prewitt, Roberts, LoG, Zero crossing, Canny etc. using different technique. Here we compared some of the detector and chosen the one of these for our edge detection which met our requirement.

The general calling syntax is

[g,t] = edge(f,'method',parameter)

4.1.1 SOBEL EDGE DETECTOR:

We are using the following syntax with parameter

[g,t] = edge(f,'sobel', [ ], both)

The output is shown below

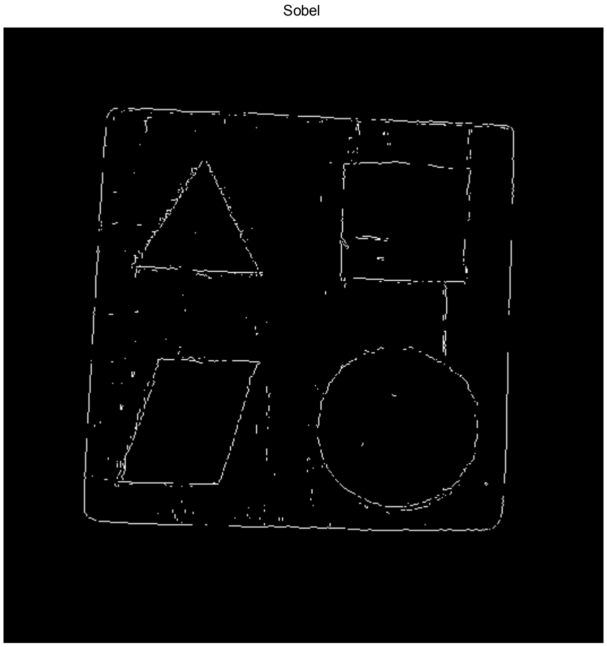


Fig 4.1 Sobel image

4.1.2 PREWITT EDGE DETECTOR:

We are using the following syntax with parameter

[g,t] = edge(f,’prewitt’, [ ], both)

The output is shown below

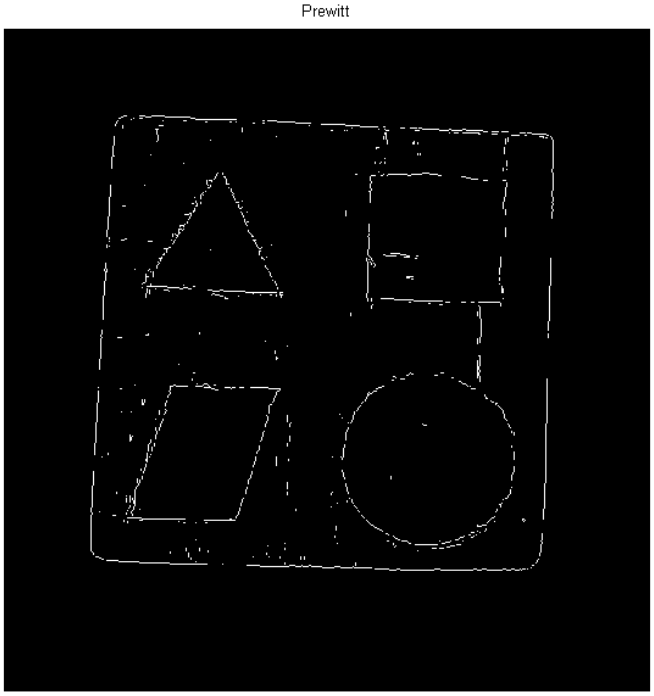


Fig 4.2 Prewitt image

4.1.3 ROBERTS EDGE DETECTOR:

We are using the following syntax with parameter

[g,t] = edge(f,’roberts’, [ ], both)

The output is shown below

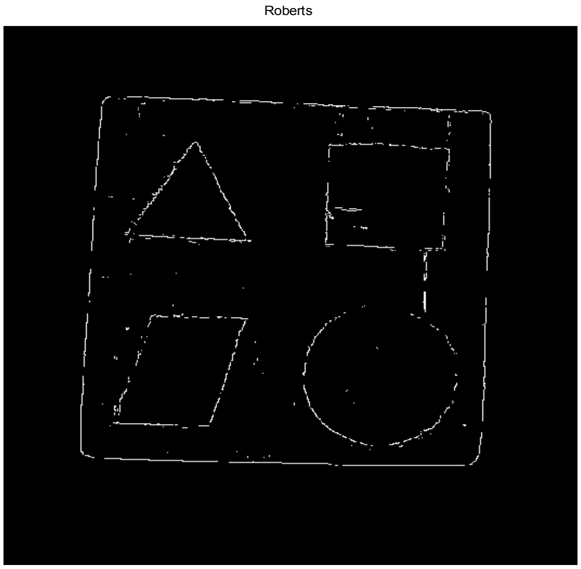


Fig 4.3 Roberts image

4.1.4 LAPLACIAN OFA GAUSSIAN(LoG) EDGE DETECTOR:

We are using the following syntax with parameter

[g,t] = edge(f,'log', [ ], 0.4)

The output is shown below

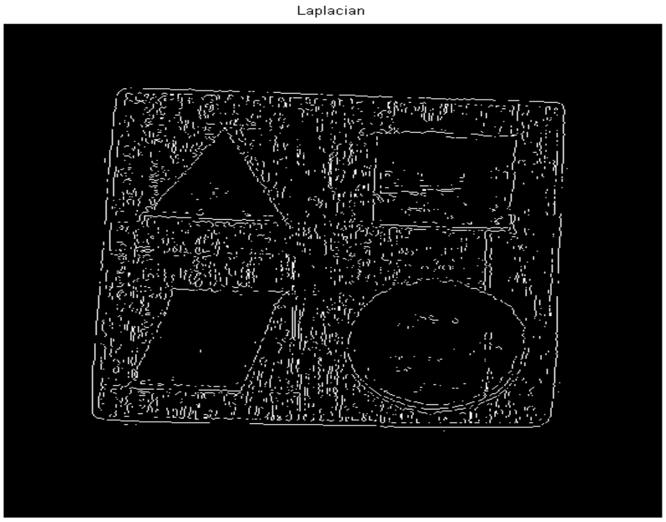


Fig 4.4 Laplacian image

4.1.5 CANNY EDGE DETECTOR:

We are using the following syntax with parameter

```
[g,t] = edge(f,’canny’, [ ], 0.4)
```

The output is shown below

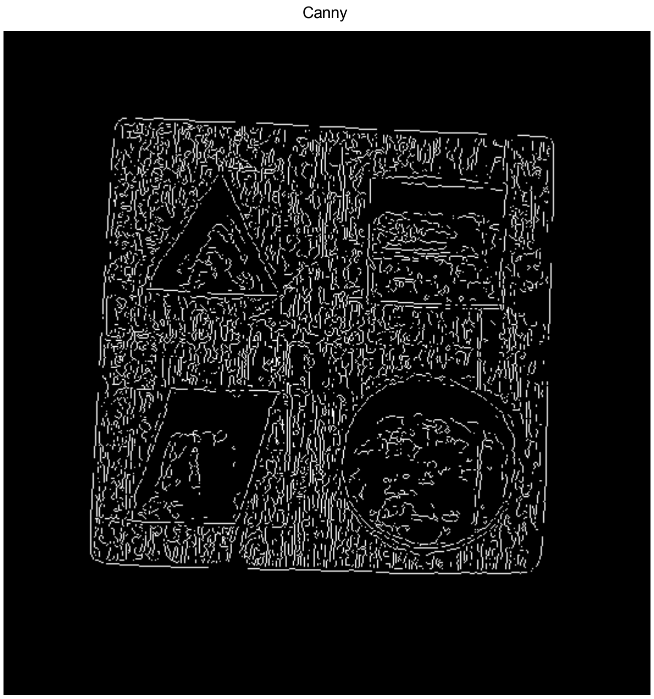


Fig 4.5 Canny image

From all the edge detector operator output, we concluded that sobel operator is suitable for our experiment.

4.2 DETECTION PROCESS

First, we made a object in wood which contained triangle, square, parallelogram & circle. The inner side of these shape are painted in black colour so that the shadow will not come in the picture. For better quality and to reduce the other effect, the image has been taken by a digital camera with white back ground. The image is shown in fig 4.6.

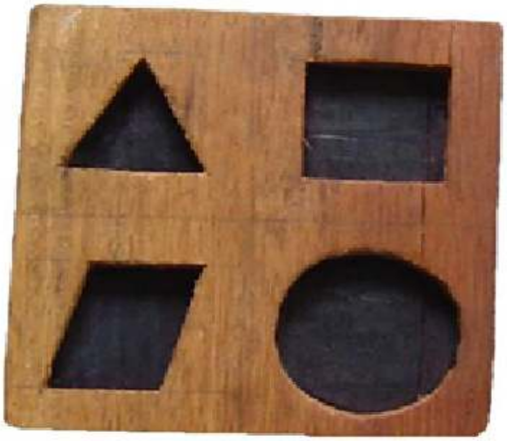


Fig 4.6 Original color image

The image is colour one so we have to convert it to gray (fig 4.7)

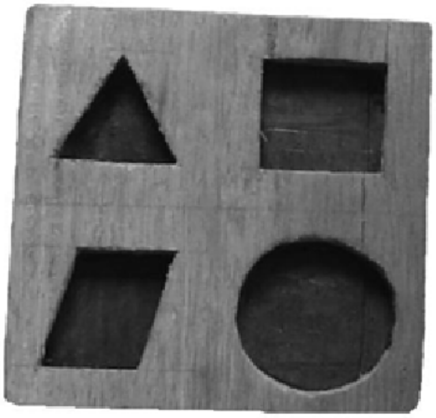


Fig 4.7 Gray image

To know the intensity level of the image, the histogram in bar form of the image has been found (fig 4.8). It gives the information about the back ground and object and how pixels have gray levels grouped into two dominant modes.

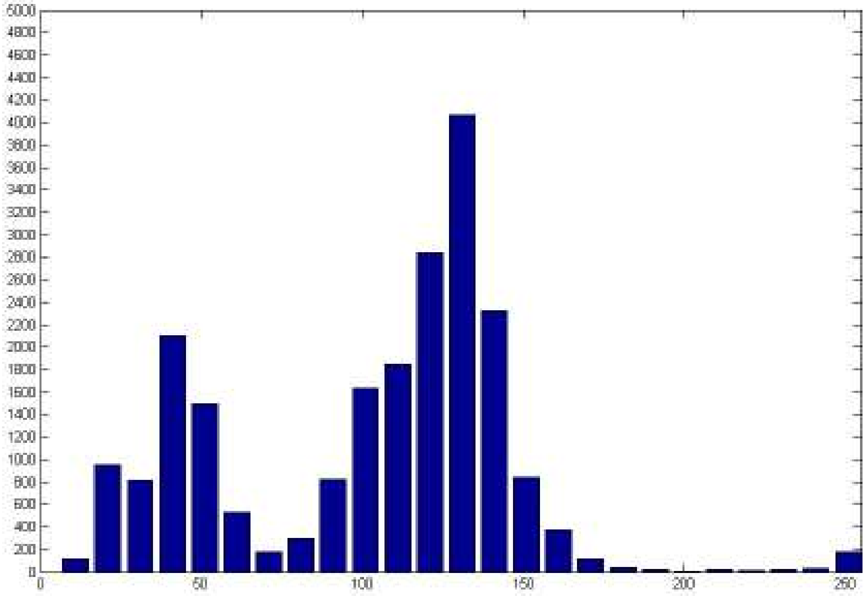


Fig 4.8 Histogram of the image

For recognizing edge and edge direction, ‘sobel’ function is used (fig 4.9). It gives the binary image with discrete point at the edges and where the intensity level changes. Points are present on every where including the edges. To clear the edges, other points should be minimizing. To minimize the point the threshold value is kept [0.9 0.9].

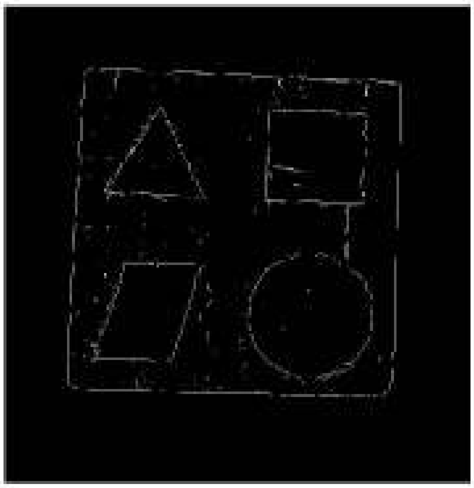


Fig 4.9 Sobel image

In binary image is the points are discrete. So the points may or may not be in the same line. To detect the co-linearity of the points the ‘Hough’ functionin in MATLAB is made and implemented (fig 4.10). It gives the information that how many points are co-linear to each other.

We are using the following syntax with parameter

[H,theta,rho]=hough(e,0.5)

The angular is 0.5 i.e. Δθ = 0.5.

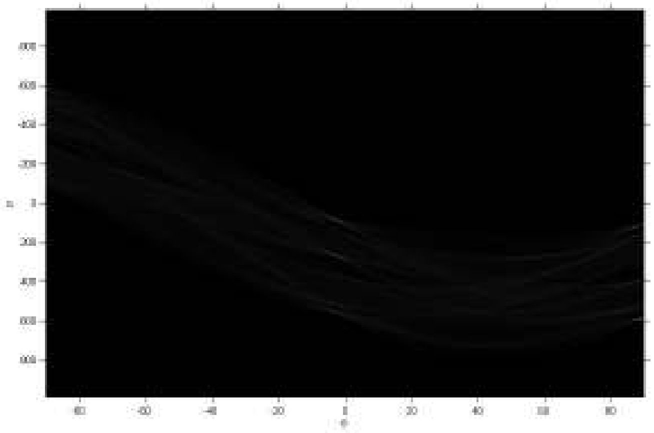


Fig 4.10 Hough transform of the image

The graph is in ρ (rho) and θ(theta) space. The sinusoidal lines are points in X-Y plane.

The numbers of point in a line vary from two to some unknown value. But we take only these lines into account which have some minimum number of points. To detect these line a function is made, called 'Houghpeaks' and implemented (fig 4.11). It shows the peak points in the figure which contain more than minimum number of line.

We are using the following syntax with parameter

[r,c]=houghpeaks(H,100,50,41)

The maximum number of peak location to be specified is 100. Value of H below 50 will not be consider to be peaks.41 specifies the value of the neighborhood around each peak that is set to zero after the peak is identified.

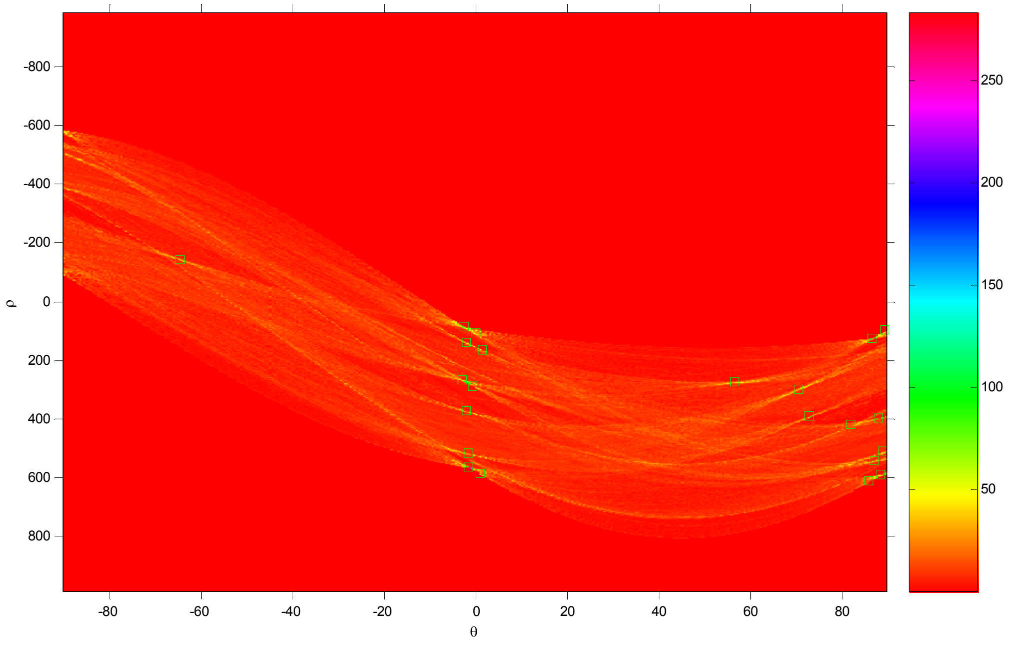


Fig 4.11 Peak point in the hough transform

The green colour square is the peak points.

To obtain the continuous line of the edge of the image, it is being required to link the points. Once a set of candidate peaks has been indentified in the Hough transform, it remains to be determine if there are line segment associate those peaks ,as well as where they start and end. For each peaks, the first step is to find the location of all nonzero pixel in the image that contributed to that peak. For this purpose, we write a function 'huoghpixels'. This pixel associate with the location found using 'houghpixel' must be grouped into line segments. To achieve this a function 'huoghlines' is made and implemented (fig 4.12).

We are using the following syntax with parameter

lines=houghlines(e,theta,rho,r,c,30,1)

The function merges the two line if they are separated by less than 30 pixel. Merged line segment less than 1 pixels long are discarded.

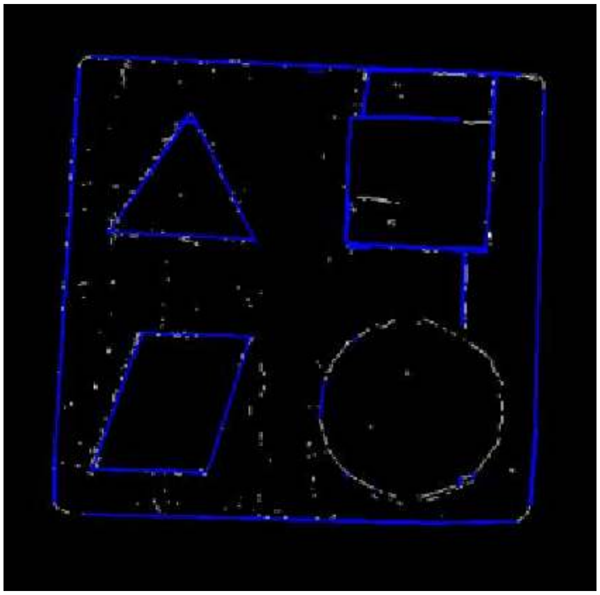


Fig 4.12 Detected edge on sobel image

The blue lines are the linking lines. In figure it is clearly visible that only the straight lines of the triangle, square and parallelogram but not the line of the circle fully.

The lines are plotted on the original image as below.

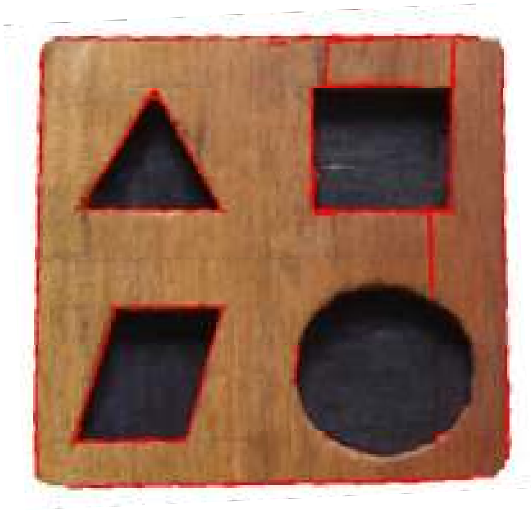


Fig 4.13 Detected edge on original image

The red colour lines show the out boundary of the image and shape inside it.

The circle in the image is obtained by joining all the point of the circle. For this purpose, we wrote a fuction called “houghcircles”.

We are using the following syntax with parameter

A= houghcircle(f1,50,100)

Here f1 is the image where we want to see the circle. The minimum radius of the circle is 50 and maximum is 100. This gives the range of the radius of the circles that can be detected. The red colour circle is plotted on the sobel image and original image as below.

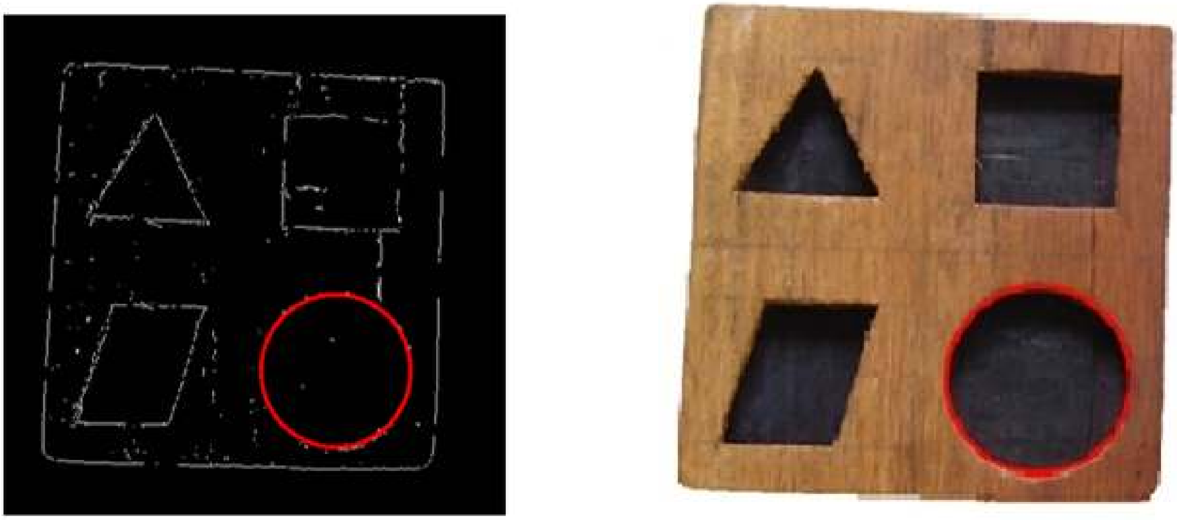


Fig 4.14 Detected circle on sobel image and Detected circle on original image

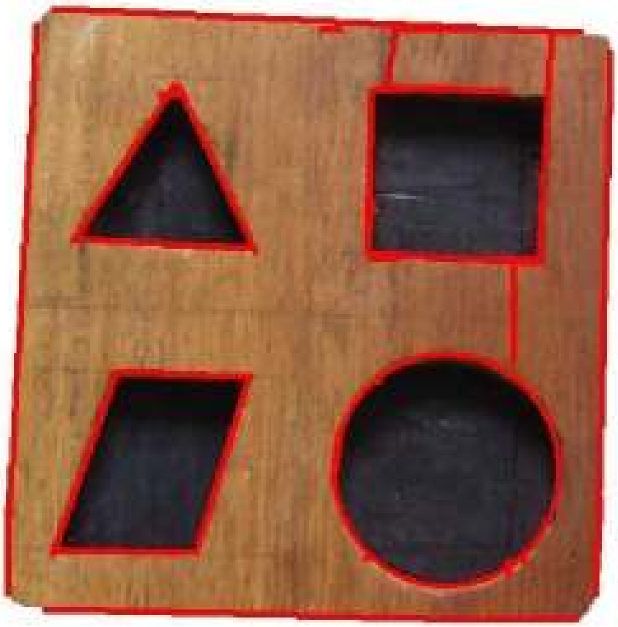


Fig 4.15 Detected edge and circle on originaL image

Chapter 5

CONCLUSIONS and SCOPE

5. CONCLUSION:

Image segmentation for a sample image is presented based on the Hough transform. This method detects both straight line and circle. When we are detecting circle, it is necessary to give the minimum and maximum value of the radius of the circle i.e. range of the circle. So it is required to have some knowledge about the radius. This method cannot be applied with unknown radius. One more problem arises when it detects the corner portion of an image if the corner is semi-circle. It detects semi-circle as circle. Though is circle are small, this can be avoided by taking the radius value large.

FUTURE SCOPE:

From the above analysis, it can be suggested this method is not efficient for circle detection where there is no information about the radius of the circle. The future work will be to develop a algorithm using hough transform for circle detection with unknown radius and detect only the full circle.

REFERENCES:

1. [WIK07] Wikipedia contributors (2007). Hough transform. In Wikipedia,The Free Encyclopedia. Retrieved 18:06, January 28, 2008
2. Chester F. Carlson. Lecture 10: Hough circle transform. Rochester Institute of Technology: Lecture notes, October 11, 2005.
3. Rafael C. Gonzalez and Richard E. Woods. Digital Image Processing. Prentice Hall, 2007. ISBN 0-13-168728-x.
4. Linda G. Shapiro and George C. Stockman (2001): “Computer Vision”, pp 279-325, New Jersey, Prentice-Hall
5. H.Chidiac, D.Ziou, “Classification of Image Edges”,Vision Interface’99, Troise-Rivieres, Canada, 1999.pp. 17-24.
6. Q.Ji, R.M.Haralick, “Quantitative Evaluation of Edge Detectors using the Minimum Kernel Variance Criterion”, ICIP 99. IEEE International Conference on Image Processing volume: 2, 1999, pp.705-709.
7. Albovik,“Handbook of Image and Video Processing”, Academic Press, 2000.
8. M.Woodhall, C.Linquist, “ New Edge Detection Algorithms Based on Adaptive Estimation Filters”, Conference Record of the 31st Asilomar IEEE Conference on Signals Systems & Computers, volume: 2, 1997, pp. 1695-1699.
9. P.V.C. Hough, Method and Means for Recognizing Complex Patterns, U.S. Patent 3069654 (1962).
10. R. K. K. Yip, “Line patterns Hough transform for line segment detection,” *IEEE Transactions on Image Processing,* pp. 319-323, 1995.
11. V. Leavers, *Shape Detection in Computer Vision Using the Hough Transform*, New York, Springer-Verlag, 1992.
12. Ioannou, D., W. Duda and F. Laine, 1999. Circle recognition through a 2D Hough Transform and radius histogramming. Image and Vision Computing, 17: 15-26.□

WEBSITES :

- http://en.wikipedia.org/w/index.php?title=Hough transform & oldid=179352381
- http://www.altera.com/literature/cp/gspx/edge-detection.pdf
- http://www.pages.drexel.edu/~weg22/edge.html
- http://www.personal.psu.edu/users/b/r/brv106/cse485/project2.htm
- http://www.pages.drexel.edu/~weg22/can_tut.html

APPENDIX

A. PROGRAM:

A1. FOR LINE DETECTION

```
clc;

close all;

clear all;

%%%%%%%%%%%%%%%%%%%%%%%

f1=imread('C:\MATLAB7\work\2.jpg');

figure; imshow(f1);

%CONVERTION FROM ORIGINAL IMAGE TO GRAY IMAGE

f=rgb2gray(f1);

figure;imshow(f);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%HISTOGRAM DETECTION

T=graythresh(f);

h0=imhist(f);

h1=h0(1:10:255);

horz=1:10:255;

figure;

bar(horz,h1);

axis([0 255 0 5000]);
```

```
set(gca,'xtick',0:50:255)

set(gca,'ytick',0:200:5000)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%HOUGH TRASFORM

e = edge(f, 'sobel', [  ],'both' );

figure,imshow(e);

[H,theta,rho]=hough(e,0.5);

figure,imshow(theta,rho,H,[   ],'notruesize'),axis on,axis normal;

xlabel('\theta'),ylabel('\rho');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%PEAKS DETECTION

[r,c]=houghpeaks(H,100,50,41);

hold on;

plot(theta(c),rho(r),'linestyle','none','marker','s','color','g');

% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%LINKING THE LINES

lines=houghlines(e,theta,rho,r,c,30,1);

figure,imshow(e),hold on;

for k=1:length(lines)

   xy=[lines(k).point1;lines(k).point2];

   plot(xy(:,2),xy(:,1),'Linewidth',2,'color',[0 0 1]);
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%END OF THE PROGRAM
```

A.2 FOR CIRCLE DETECTION

```
function circles = houghcircle(im, minR, maxR, ratio, delta)



% Check inupt arguments

if nargin==3

  ratio = 0.3;   % 1/3 of the perimeter

  delta = 12;    % Each element in [cx cy R] may deviate 4 pixels approx.

elseif nargin==4

  delta = 12;

elseif nargin~=5

  disp('Require at least 3 input arguments');

  return;

end

if minR<0 || maxR<0 || minR>maxR || ratio<0 || ratio>1 || delta<0

  disp('Required: 0<minR, 0<maxR, minR<=maxR, 0<ratio<1, and 0<delta');

  return;

end

% Turn a color image into gray

origim = im;

if length(size(im))>2
```

```
  im = rgb2gray(im);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Create a 3D Hough array. The first two dimensions specify the
% coordinates of the circle centers, and the third specifies the radii.
maxR2 = 2*maxR;
sy = size(im,1)+maxR2;
sx = size(im,2)+maxR2;
hough = zeros(sy, sx, maxR-minR+1);
edgeim = edge(im, 'canny', [0.15 0.2]);
[X Y] = meshgrid(0:maxR2, 0:maxR2);
Rmap = round(sqrt((X-maxR).^2 + (Y-maxR).^2));
Rmap(Rmap<minR | Rmap>maxR) = 0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% For each edge pixel, increment the corresponding elements in the Hough
% array.
[py px] = find(edgeim);
[y x R] = find(Rmap);
ind1 = y-1 + (x-2)*sy + (R-minR)*sx*sy;
for i = 1:length(px);
  index = (y+py(i)-1) + (x+px(i)-2)*sy + (R-minR)*sx*sy;   % Orginal formula
%   index = ind1 + py(i) + px(i)*sy;
  hough(index) = hough(index)+1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Collect circles
circles = zeros(0,4);   % Format: [cx cy R c]
rate = 2*pi*ratio;
for R = minR:maxR   % Loop from minimal to maximal radius
  slice = hough(:,:,R-minR+1);
  slice(slice<rate*R) = 0;   % Clear pixel count < 2*pi*R*ratio
  [y x count] = find(slice);
  circles = [circles; [x-maxR, y-maxR, R*ones(length(x),1), count]];
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Delete similar circles
circles = sortrows(circles,-4);  % Sort according to pixel count
i = 1;
while i<size(circles,1)
  j = i+1;
  while j<=size(circles,1)
    if sum(abs(circles(i,1:3)-circles(j,1:3))) <= delta
      circles(j,:) = [];
    else
      j = j+1;
    end
  end
  i = i+1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Draw circles on the original image
figure, imshow(origim), hold on;
for i = 1:size(circles,1)
  x = circles(i,1)-circles(i,3);
  y = circles(i,2)-circles(i,3);
  w = 2*circles(i,3);
  rectangle('Position', [x y w w], 'EdgeColor', 'red', 'Curvature', [1 1],'LineWidth',3);
end
hold off;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%END OF THE PROGRAM
```

A.3 MAIN PROGRAM

```
clc;
close all;
clear all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Using the “circles” fuction
A= houghcircle(e,50,100);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%END OF THE PROGRAM
```

IMAGE SEGMENTATION USING HOUGH TRASFORM

Author: Rabindra Kumar Murmu

Meena Jhaniya

Supervisors: U.C. Pati

Abstract: The objective of this application is the recognition of different shapes in an image. This task can be subdivided into following procedures. First, an image is converted into gray scale. For detecting lines in images, we used histogram for the intensity related information and threshold. Then, an edge recognition procedure is implemented. We use the first-derivative Sobel detector to determine the edges and edge directions. Then, a Hough transform is accomplished on the threshold edge map for linking the edges. This is the procedure to detect straight line. For circle detection we wrote a function called "houghcircle".

Introduction: In computer vision, **segmentation** refers to the process of partitioning a digital image into multiple regions (sets of pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. Edge detection is a very important area in the field of Computer Vision. Edges define the boundaries between regions in an image, which helps with segmentation and object recognition. They can show where shadows fall in an image or any other distinct change in the intensity of an image. Edge detection is a fundamental of low-level image processing and good edges are necessary for higher level processing.

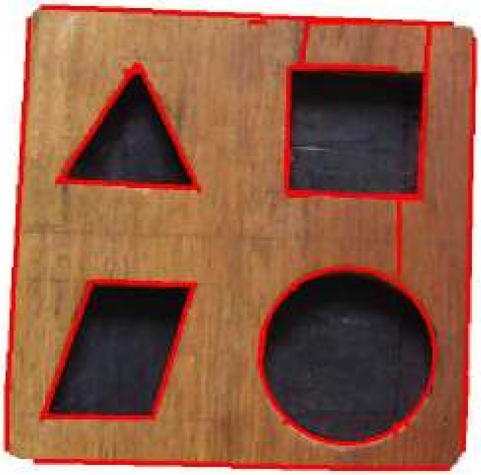
Objective: The basic aim of the project is to detect different shapes like triangle, rectangle, rhombus and circle etc. in an image using hough transform.

Methodology:

- Compute the gradient of an image and threshold it to obtain a binary image.
- Specify subdivisions in the ρθ-plane.
- Examine the counts of the accumulator cells for high pixel concentrations.
- Examine the relationship (principally for continuity) between pixels in a chosen cell.
  - Based on computing the distance between disconnected pixels identified during traversal of the set of pixels corresponding to a given accumulator cell.

- A gap at any point is significant if the distance between that point and its closet neighbor exceeds a certain threshold.

Results: The edge of the sample image is detected using Hough Transform. The figure below is the result of the experiment.



Conclusion: Image segmentation for a sample image is presented based on the Hough transform. This method detects both straight line and circle. When we are detecting circle, it is necessary to give the minimum and maximum value of the radius of the circle i.e. range of the circle. So it is required to have some knowledge about the radius. This method cannot be applied with unknown radius. One more problem arises when it detects the corner portion of an image if the corner is semi-circle. It detects semi-circle as circle. Though is circle are small, this can be avoided by taking the radius value large.

References:

1. [WIK07] Wikipedia contributors (2007). Hough transform. In Wikipedia,The Free Encyclopedia. Retrieved 18:06, January 28, 2008
2. Chester F. Carlson. Lecture 10: Hough circle transform. Rochester Institute of Technology: Lecture notes, October 11, 2005.
3. Rafael C. Gonzalez and Richard E. Woods. Digital Image Processing. Prentice Hall, 2007. ISBN 0-13-168728-x.