Project Report

on

Detection all horizontal and vertical lines in a image

(Image Processing Project)

Community College of Skill Development

J. C. BOSE UNIVERSITY OF SCIENCE & TECHNOLOGY, YMCA
FARIDABAD-121006

May 2020

Submitted by:

VRINDA
19021271013

CANDIDATE'S DECLARATION

I hereby certify that the work which is being carried out in this Mini Project title "Detect all horizontal and vertical lines in a image" in fulfillment of the requirement for the award of Post Graduate Diploma in Data Science and Analytics and submitted to "**J. C. Bose University of Science and Technology, YMCA, Faridabad**", is an authentic record of my own work carried out under the supervision of Shailender Gupta Sir.

The work contained in this project has not been submitted to any other University or Institute for the award of any other degree or diploma by me.

VRINDA

CERTIFICATE

This is to certify that the work carried out in this project titled **Detect all horizontal and vertical lines in a image** submitted by Vrinda to "**J. C. Bose University of Science and Technology, YMCA, Faridabad**" for the award of the Post Graduate Diploma in Data Science and Analytics is a record of bonafide work carried out by her under my supervision. In my opinion, the submitted report has reached the standards of fulfilling the requirements of the regulations to the degree.

Shailender Gupta
(Supervisor)
J. C. Bose University of Science and Technology, YMCA, Faridabad

Dr. Sanjeev Goyal
Principal,
Community College of Skill Development,
J. C. Bose University of Science and Technology, YMCA, Faridabad

TABLE OF CONTENTS

CHAPTER 1:
PROBLEM DEFINITON AND OBEJCETIVE

Determining the location and orientation of straight lines in images is a problem of great importance in the field of image processing. The Hough transform, has been widely used to solve this problem for binary images, and is still by far dominating the scene when it comes to line detection and its concept was further developed to detect other shapes like circles. In this paper, a new method for straight line detection is proposed. The formulation of this new method is based on the use of the coordinates of pixels in the digital image, which relates the parameters determining the location and orientation of the lines in the image. The advantage of this method is that it provides an approach to the problem of line detection within a framework that enhances the detection of lines by incorporation of prior information of the images nature. This approach proves useful and efficient in computational schemes where it is needed to detect lines in 0, 45, 90, and 135 degree directions, and results in optimized performance. Experimental results of this new approach shows how the image coordinates method can be viewed as an alternate optimal method for noisy images and dotted lines in terms of detection accuracy and accumulator size reduction.

CHAPTER 2:
INTRODUCTION

Image processing is a method of analyzing and manipulating the digital images with the computer using mathematical operators. In image processing, the input is an image and outcome may be either set of characteristics or set of the parameter of image or an image. An image comprises various information like contour of the object, its orientation, size and color. Most of the shape information of an image is enclosed in edges. So first detect these edges in an image and by using these filters and then by enhancing those areas of image which contains edges, sharpness of the image will increase and image.

Line detection

Line detection is a very basic, yet important problem in image processing. Line detection is an algorithm that takes a collection of an edge points and finds all the lines on which these edge points lie.

It is an approach based on whether sets of pixels lie on curves of a specified shape. Once detected, these curves form the edges of interest. Most of the shape information of an image is enclosed in edges. So first detect these edges in an image and by using these filters and then by enhancing those areas of image.

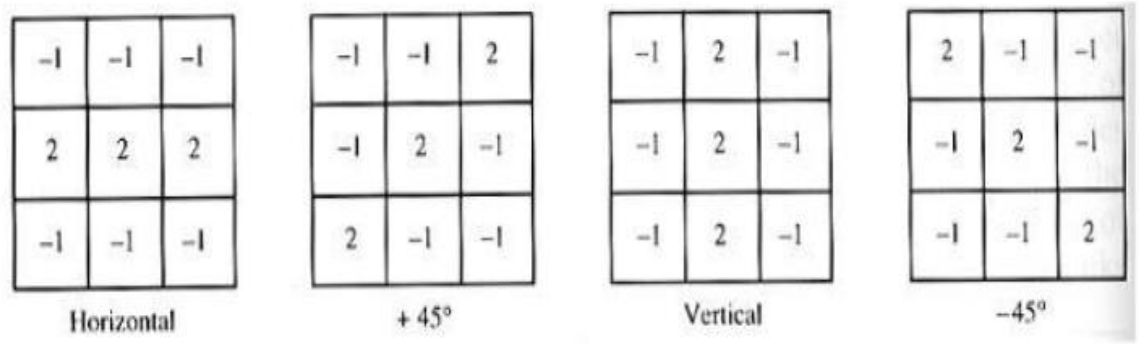
Which contains edges, sharpness of the image will increase and image will become clearer. However the output from an edge detector is still an image described by its pixels. If lines, ellipses and so forth could be defined by their characteristic equations, the amount of data would be reduced even more.

I. CONVOLUTION BASE TECHNIQUE

Convolution is a simple mathematical operation which is fundamental to many common image processing operators. Convolution provides a way of 'multiplying together' two arrays of numbers, generally of different sizes, but of the same dimensionality, to produce a third array of numbers of the same dimensionality. This can be used in image processing to implement operators whose output pixel values are simple linear combinations of certain input pixel values.

In an image processing context, one of the input arrays is normally just a gray level image. The second array is usually much smaller, and is also two dimensional (although it may be just a single pixel thick), and is known as the kernel. The convolution is performed by sliding the kernel over the image, generally starting at the top left corner, so as to move the kernel through all the positions where the kernel fits entirely within the boundaries of the image. (Note that implementations differ in what they do at the edges of images, as explained below.) Each kernel position corresponds to a single output pixel, the value of which is calculated by multiplying together the kernel value and the underlying image pixel value for each of the cells in the kernel, and then adding all these numbers together.

The line detection operator consists of a convolution kernel tuned to detect the presence of lines of a particular width n, at a particular orientation θ. Figure 1 shows a collection of four such kernels, which each respond to lines of single pixel width at the particular orientation shown.



Usually the black box (system) used for image processing is an LTI system or linear time invariant system. By linear we mean that such a system where output is always linear, neither log nor exponent or any other. And by time invariant we means that a system which remains same during time.

Mathematically represented as

g(x,y) = h(x,y) * f(x,y)

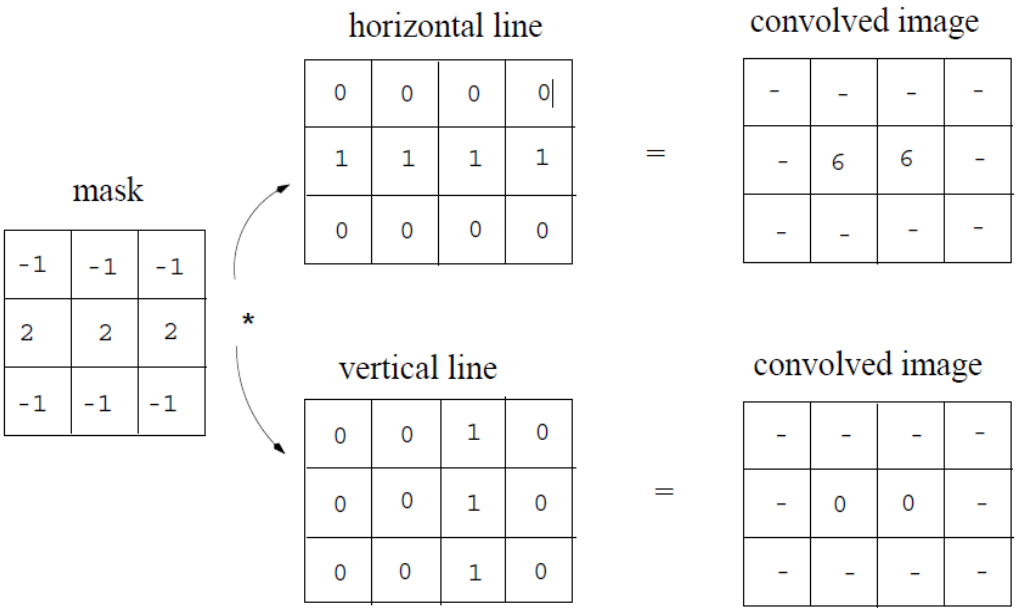
It can be explained as the “mask convolved with an image”.

What is mask?

Mask is also a signal. It can be represented by a two dimensional matrix. The mask is usually of the order of 1x1, 3x3, 5x5, 7x7. A mask should always be in odd number, because otherwise you cannot find the mid of the mask. Why do we need to find the mid of the mask.

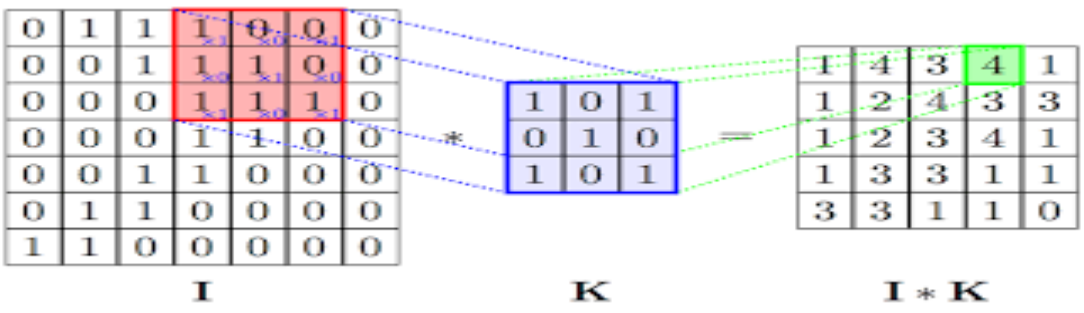
Why Convolution

Convolution can achieve something, that the previous two methods of manipulating images can't achieve. Those include the blurring, sharpening, edge detection, noise reduction etc.



Below image will explain the convolution operation, "I" is the image and "K" is the filter and "I*K" is the result of convolution.

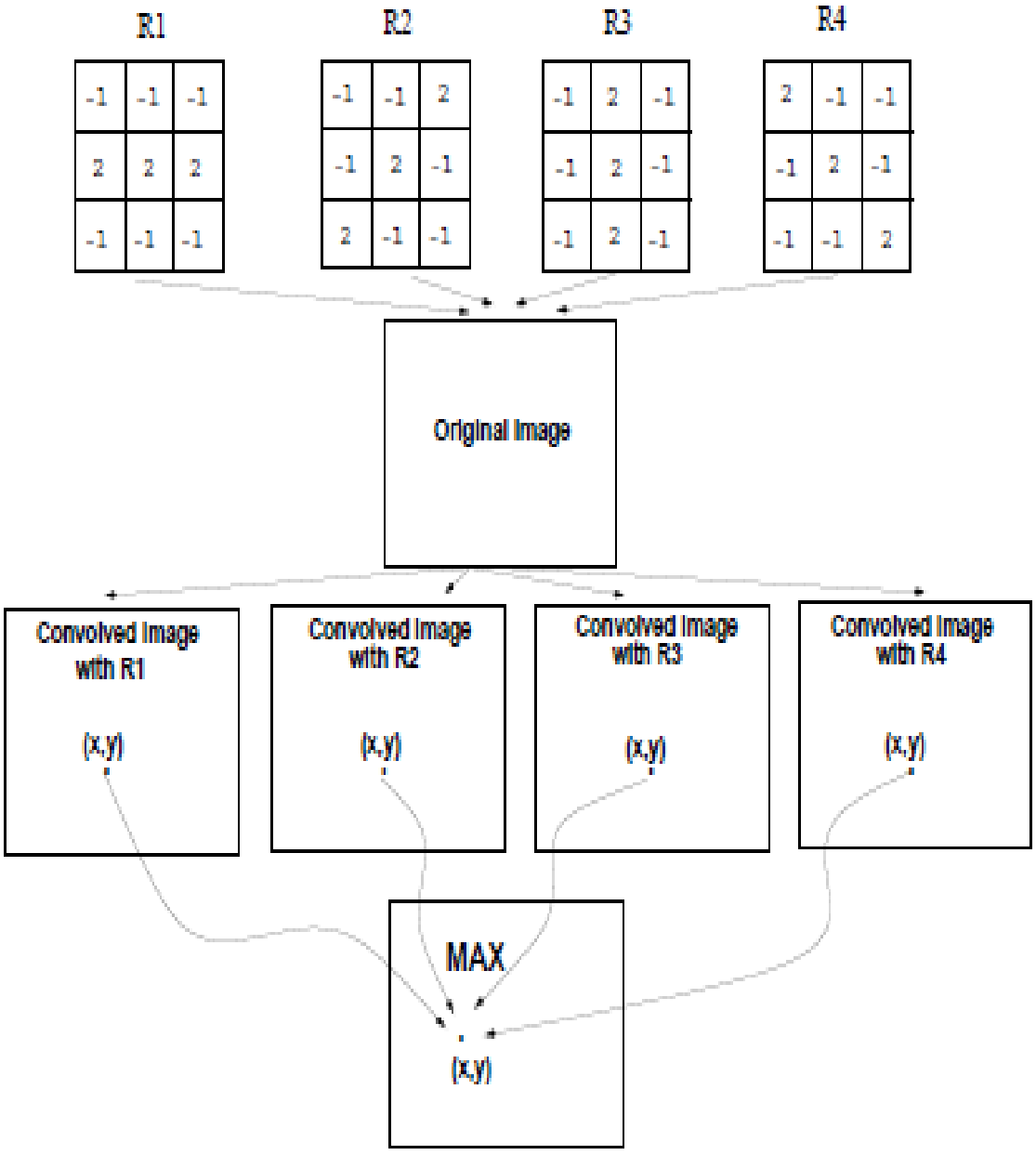
Image is of size 7*7 and the filter size is 3*3 with no padding and stride value as 1. filter "K" will be moving across "I" one step at a time and for each step that part of "I" will be multiplies with "K", this process will be done over and over until the filter cover all the pixels in "I" and form the resultant image "I*K".



- In practice, we run every mask over the image and we combine the responses:

$$R(x,y) = max(|R_1(x,y)|, |R_2(x,y)|, |R_3(x,y)|, |R_4(x,y)|)$$

If $R(x,y) > T$, then discontinuity

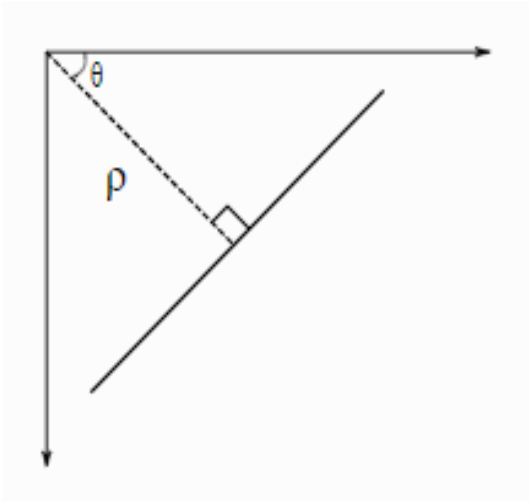


II. HOUGH TRANSFORM

The Hough Transform is a method that is used in image processing to detect any shape, given a binary image. It can detect the shape even if it is broken or distorted a little bit.

A “simple” shape is one that can be represented by only a few parameters. For example, a line can be represented by two parameters only (slope, intercept) and a circle has three parameters – the coordinates of the center and the radius (x, y, r).

A line can be represented as $y = mx + c$ or in parametric form $\rho = x\cos\theta + y\sin\theta$, as where ρ is the perpendicular distance from origin to the line, and θ is the angle formed by this perpendicular line and horizontal axis measured in counter-clockwise (That direction varies on how you represent the coordinate system. This representation is used in OpenCV). Check below image:



So if line is passing below the origin, it will have a positive rho and angle less than 180. If it is going above the origin, instead of taking angle greater than 180, angle is taken less than 180, and rho is taken negative. Any vertical line will have 0 degree and horizontal lines will have 90 degree.

Now let’s see how Hough Transform works for lines. Any line can be represented in these two terms, (ρ, θ). So first it creates a 2D array or accumulator (to hold values of two parameters) and it is set to 0 initially. Let rows denote the ρ and columns

denote the θ. Size of array depends on the accuracy you need. Suppose you want the accuracy of angles to be 1 degree, you need 180 columns. For ρ,, the maximum distance possible is the diagonal length of the image. So taking one pixel accuracy, number of rows can be diagonal length of the image.

Consider a 100x100 image with a horizontal line at the middle. Take the first point of the line. You know its (x,y) values.

Now in the line equation, put the values $\theta = 0, 1, 2, ..., 180$ and check the ρ, you get. For every (ρ, θ) pair, you increment value by one in our accumulator in its corresponding (ρ, θ) cells. So now in accumulator, the cell (50,90) = 1 along with some other cells.

Now take the second point on the line. Do the same as above. Increment the the values in the cells corresponding to (ρ, θ) you got. This time, the cell (50,90) = 2. What you actually do is voting the (ρ, θ) values. You continue this process for every point on the line. At each point, the cell (50,90) will be incremented or voted up, while other cells may or may not be voted up. This way, at the end, the cell (50,90) will have maximum votes. So if you search the accumulator for maximum votes, you get the value (50,90) which says, there is a line in this image at distance 50 from origin and at angle 90 degrees.

Working of Houghline method:

- First it creates a 2D array or accumulator (to hold values of two parameters) and it is set to zero initially.
- Let rows denote the r and columns denote the (θ)theta.
- Size of array depends on the accuracy you need. Suppose you want the accuracy of angles to be 1 degree, you need 180 columns(Maximum degree for a straight line is 180).
- For r, the maximum distance possible is the diagonal length of the image. So taking one pixel accuracy, number of rows can be diagonal length of the image.

Summarizing the process

- In an image analysis context, the coordinates of the point(s) of edge segments (i.e. X,Y) in the image are known and therefore serve as constants in the parametric line equation, while R(rho) and Theta(θ) are the unknown variables we seek.
- If we plot the possible (r) values defined by each (theta), points in cartesian image space map to curves (i.e. sinusoids) in the polar Hough parameter space. This point-to-curve transformation is the Hough transformation for straight lines.
- The transform is implemented by quantizing the Hough parameter space into finite intervals or accumulator cells. As the algorithm runs, each (X,Y) is transformed into a discretized (r,0) curve and the accumulator(2D array) cells which lie along this curve are incremented.
- Resulting peaks in the accumulator array represent strong evidence that a corresponding straight line exists in the image.

The Hough Space Accumulator

To determine the areas where most Hough space lines intersect, an accumulator covering the Hough space is used. When an edge point is transformed, bins in the accumulator is incremented for all lines that could pass through that point. The resolution of the accumulator determines the precision with which lines can be detected. In this worksheet a resolution of 1 pixel for r and 1 degree for θ has been used. In general, the number of dimensions of the accumulator corresponds to the number of unknown parameters in the Hough transform problem. Thus, for ellipses a 5-dimensional space is required (the coordinates of its center, the length of its major and minor axis, and its angle). For lines 2 dimensions suffice (r and θ). This is why it is possible to visualize the content of the accumulator.

HOUGH TRANSFORM APPLICATIONS

Due to the merits of the Hough Transform mentioned above e.g. Noise immunity, it has been widely used in many applications. For example, 3D applications, detection of objects and shapes, lane and road sign recognition, industrial and medical applications, pipe and cable inspection, and underwater tracking. Some of these applications are illustrated below:

***A. Traffic and Transport Applications*.**

A system for vehicle license plate (VLP) detection. The system combines the HT and a contour detecting algorithm to improve the speed. Although the input images can be taken from various distances with inclined angles, the sys- tem is able to detect multiple VLPs. A perspective centroid-based HT for Skew correction of license plate recognition is shown. It gets first the centroid of each char- actor on the plate, and then the HT is used to estimate a line passing by the centroids. Knowing the skew angle, the plate position can be corrected for a more accurate recognition. A project for rail road defect inspection is proposed in. It finds the defects of the railroad tracks with a notification to the inspector of the type and location of the defect. For example, the CHT is used to detect missing bolts, defects, and cracks in ties.

***B. Biometrics and Man-Machine Interaction*.**

A real- time human-robot interaction system based on hand gestures and tracking. It combines the connected component label ling (CCL), and the HT. Detecting the skin color, it has the ability to extract the center of the hand, the directions and the fingertip positions of all outstretched fingers. On the other hand, the HT is applied to finger print matching. The algorithm uses a robust alignment method (descriptor – based HT). Additionally, it measures the similarity between fingerprints by considering both minutiae and orientation field information. The CHT is used to locate the eyes and the heads for the people.

C. Object Recognition

An approach for object detection using the GHT with the color similarity between homogeneous segments of the object. The input of the algorithm is previously segmented regions with homogeneous color. According to this work, the approach is robust to illumination changes, occlusion and distortion of the segmentation output. It is able to recognize objects in spite of being translated, rotated, scaled and even located in a complex environment. The work proposes a robust method for recognizing objects among clutter and in the presence of occlusion. A close to real-time performance is achieved. The recognition of the objects depends mainly on matching individual features stored in a database of known objects using a fast nearest neighbor algorithm. The HT is then used to identify clusters belonging to a single object. In addition to this, recognition of multiple instances of an object is proposed.

D. Object Tracking

A shape based tracking is per- formed using a mixture of uniform and Gaussian Hough (MOUGH). This approach is able to locate and track objects even against complex backgrounds such as dense foliage while the camera is moving. An object tracking in a sequence of sparse range images is suggested using the bounded Hough transform (BHT). This is a variation of the GHT which exploits the coherence across image frames. A method for rectangular object tracking. First, it finds the edges using the canny edge detector, and then applies the HT to find all lines in a predefined window surrounding the tracked object. A spatial color histogram is proposed in [58] for segmentation and object tracking. The spatial color histogram model encodes the color distribution and the spatial information. The GHT is used as a location estimator, which estimates the object location from a frame to another using its voting capabilities.

***E. Underwater Applications*.**

A method for geometrical shape recognition for an underwater robot images. The recognition problem is transformed into a bounded error estimation problem compared to the classical GHT. An underwater system is developed to carry out visually guided tasks on an autonomous underwater vehicle (AUV). Underwater pipelines are detected using the HT.

Limitations

The Hough transform is only efficient if a high number of votes fall in the right bin, so that the bin can be easily detected amid the background noise. This means that the bin must not be too small, or else some votes will fall in the neighboring bins, thus reducing the visibility of the main bin.

Also, when the number of parameters is large (that is, when we are using the Hough transform with typically more than three parameters), the average number of votes cast in a single bin is very low, and those bins corresponding to a real figure in the image do not necessarily appear to have a much higher number of votes than their neighbors.

The complexity increases at a rate of $O(A^{m-2})$ with each additional parameter, where A is the size of the image space and m is the number of parameters. (Shapiro and Stockman, 310) Thus, the Hough transform must be used with great care to detect anything other than lines or circles.

Finally, much of the efficiency of the Hough transform is dependent on the quality of the input data: the edges must be detected well for the Hough transform to be efficient. Use of the Hough transform on noisy images is a very delicate matter and generally, a denoising stage must be used before. In the case where the image is corrupted by speckle,.

CHAPTER 3 :

HARDWARE AND SOFTWARE REQUIREMENTS

Operating System Requirement

Computers running Microsoft Windows must meet the following minimal hardware and software requirements.

Hardware Requirements

- Dual-core 64-bit processor
- Min 4 GB of memory
- Up to 24 GB of internal storage
- Network interface card
- Windows 10, Windows 8.1 Update, Windows 8, or Windows 7.

Software Requirements

- Jupyter Notebook (or any other Python IDE)
- Python 3.0 or Later (OpenCV, Matplotlib and Numpy installed)

CHAPTER 4:

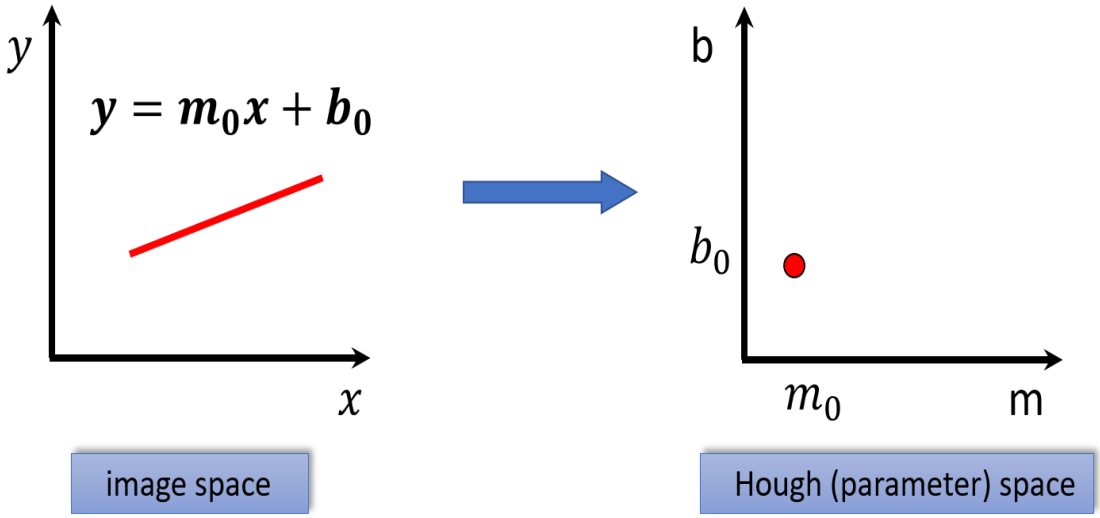
LINE DETECTION BY HOUGH TRANSFORMATION

1. Line Detection

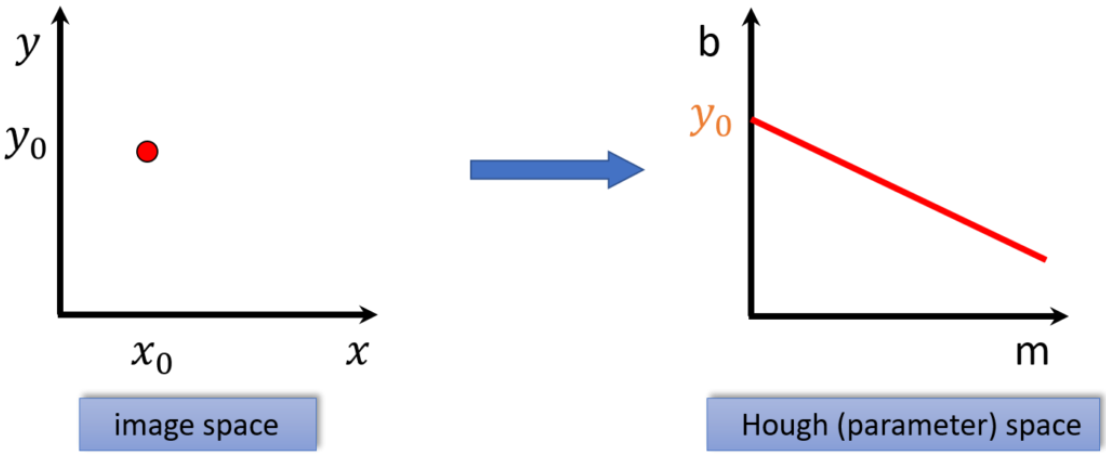
Our first approach in line detection can be the basic brute search method. We can scan a large number of pixels and ask if they belong to the same line. You might be thinking is this not computationally expensive? Yeah, you are right. Even with very fast computers checking for every possible line on an image requires a lot of time. We want to somehow allow the data to decide where the line is.

2. Hough Space

We are going to take you through the main idea of line detection. The main concept of the Hough transform is an understanding of Hough space.

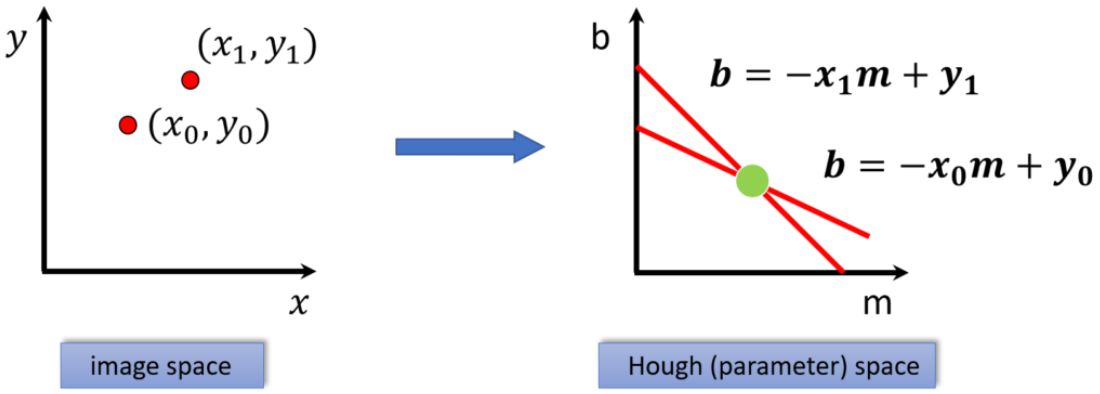


Let's say we have a line in an image space represented by the equation $y=m_0x+b_0$. Given this line, we want to represent it as a point in Hough space with the parameters m and b. **In other words, the line in the image space should correspond to a point in Hough space.**

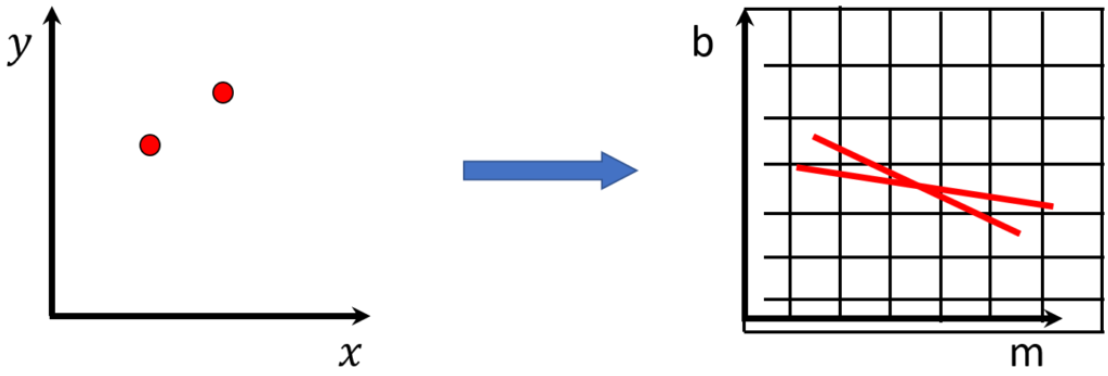


$$y = mx_0 + b \qquad b = -x_0m + y_0$$

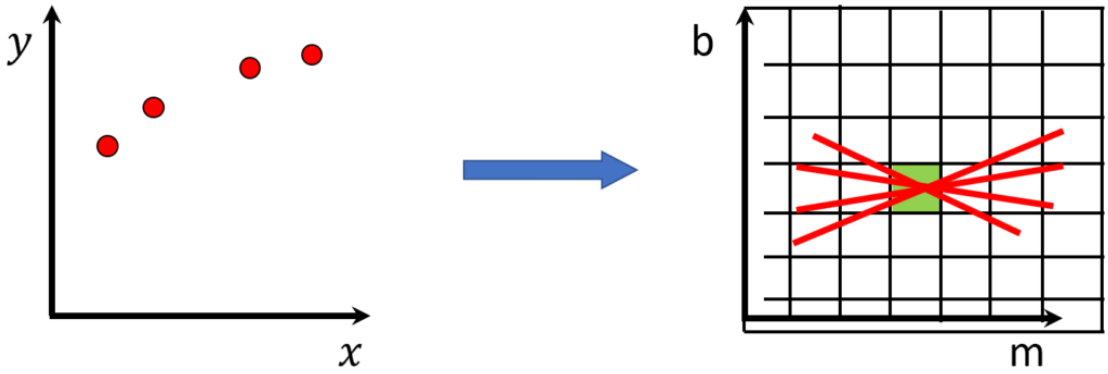
With that being said, we are going to do something different. Imagine that we have a point represented as x_0, y_0 in the image space. We are going to derive the equation of that line that goes through the point in order to satisfy the equation $y_0=m_0x+b_0$. As an illustration, we put in another point in the graph x_1, y_1. Yet, it is simply another line in Hough space.



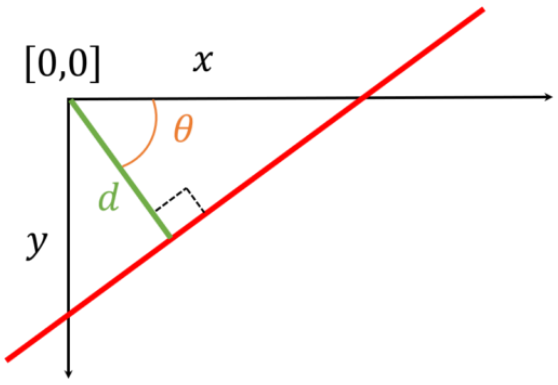
Now, the question is, what line is consistent with both points? Certainly, it has to be the line that corresponds to a point of intersection in Hough space. As shown above, that is how we are going to identify lines from points. To put it differently, every point in the image space gives us a line in Hough space. Let's view another image below.



A grid in m and b space is being created. Once every point casts a vote, every vote are taken into consideration and whatever bin has the most votes, that is our line.



3. Polar representation of lines



d: perpendicular distance from line to origin

θ: angle the perpendicular makes with the x-axis

$$x\cos\theta + y\sin\theta = d$$

Before continuing, if we carefully analysed our representation of lines, we detect some flaws using the m and b representation of lines. To clarify, vertical line, concerning m=infinity. With this in mind, have this perception of an infinite slope can be very painful, which leads us to come up with a much powerful representation of line called the “Polar representation for lines”. Not to mention, we do not have any bad numerical problems. In this representation, the redline is going to be defined by two quantities.

One of them is the perpendicular distance. This is the distance to the closest point on the line to the origin, d. Along with the second parameter theta, which is the angle, the perpendicular makes with the x axis.

4. Implementation

The *HoughLineP()* function finds circles on grayscale images using a Hough Transform.

- image – The output from the edge detector. This is a grayscale image.
- rho – Distance resolution in pixels of the Hough grid which is the parameter d.
- theta – Angular resolution in radians of the Hough grid which in our case is theta.
- threshold – Minimum number of votes (intersections in Hough grid cell)
- minLineLength – Minimum number of pixels making up a line.
- maxLineLength – Maximum gap in pixels between connectable line segments.

Algorithm

The algorithm for detecting straight lines can be divided into the following steps:

1. Edge detection, e.g. using the Canny edge detector.

2. Mapping of edge points to the Hough space and storage in an accumulator.

3. Interpretation of the accumulator to yield lines of infinite length. The interpretation is done by thresholding and possibly other constraints.

4. Conversion of infinite lines to finite lines.

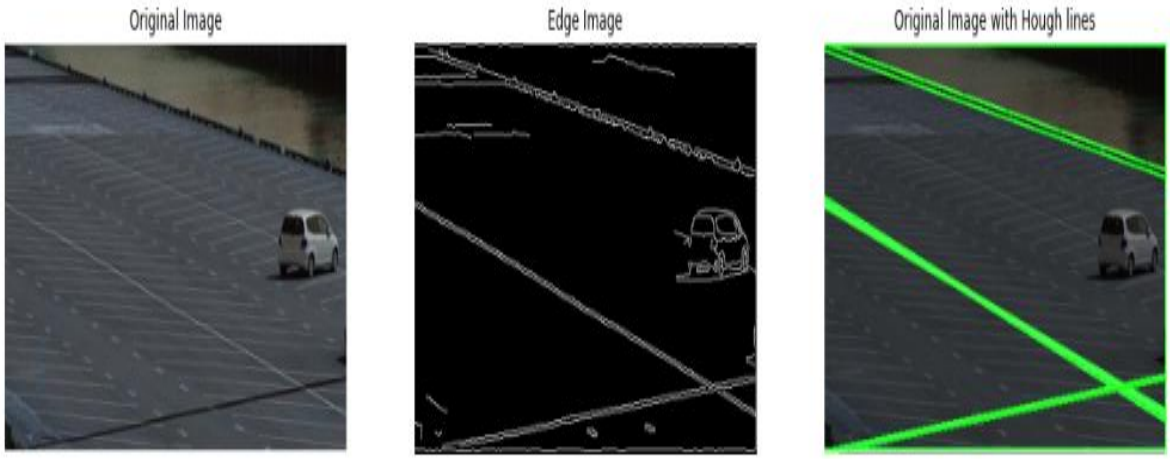
CODE

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

#function # Draw the lines represented in the hough accumulator on the original image
def drawhoughLinesOnImage(image, houghLines):
   for line in houghLines:
      for rho,theta in line:
         a = np.cos(theta)
         b = np.sin(theta)
         x0 = a*rho
         y0 = b*rho
         x1 = int(x0 + 1000*(-b))
         y1 = int(y0 + 1000*(a))
         x2 = int(x0 - 1000*(-b))
         y2 = int(y0 - 1000*(a))
         cv2.line(image,(x1,y1),(x2,y2),(0,255,0), 2)

# Different weights are added to the image to give a feeling of blending
def blend_images(image, final_image, alpha=0.7, beta=1., gamma=0.):
   return cv2.addWeighted(final_image, alpha, image, beta,gamma)

image = cv2.imread("road.jpg") # load image in grayscale
gray_image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
blurredImage = cv2.GaussianBlur(gray_image, (5, 5), 0)
edgeImage = cv2.Canny(blurredImage, 50, 120)

# Detect points that form a line
dis_reso = 1 # Distance resolution in pixels of the Hough grid
theta = np.pi /180 # Angular resolution in radians of the Hough grid
threshold = 170 # minimum no of votes

houghLines = cv2.HoughLines(edgeImage, dis_reso, theta, threshold)

houghLinesImage = np.zeros_like(image) # create and empty image

drawhoughLinesOnImage(houghLinesImage, houghLines) # draw the lines on the empty image
orginalImageWithHoughLines = blend_images(houghLinesImage,image) # add two images together,
using image blending

fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3, figsize=(20, 20))
ax1.imshow(image)
ax1.set_title('Original Image')
ax1.axis('off')

ax2.imshow(edgeImage, cmap='gray')
ax2.set_title('Edge Image')
ax2.axis('off')

ax3.imshow(orginalImageWithHoughLines, cmap='gray')
ax3.set_title("Original Image with Hough lines")
ax3.axis('off')
```

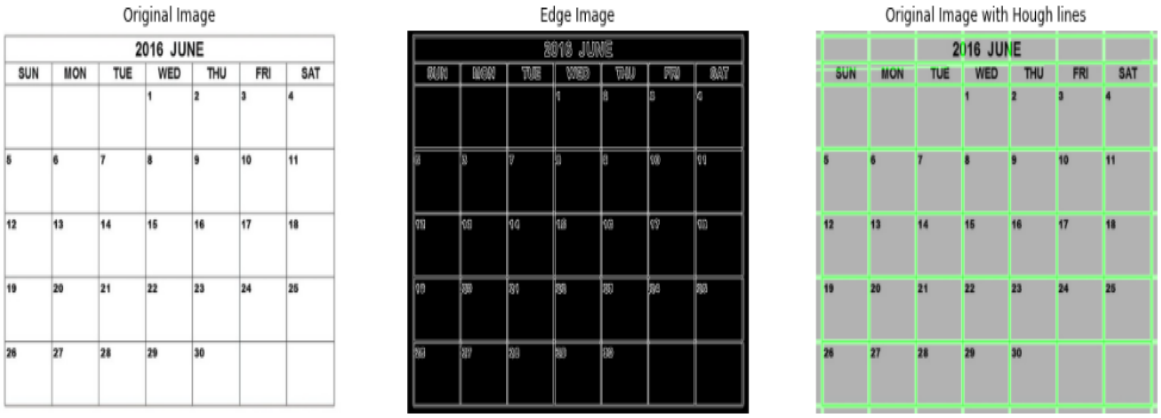
CHAPTER 5:

RESULTS (SCREENSHOTS, GRAPHS Etc.)

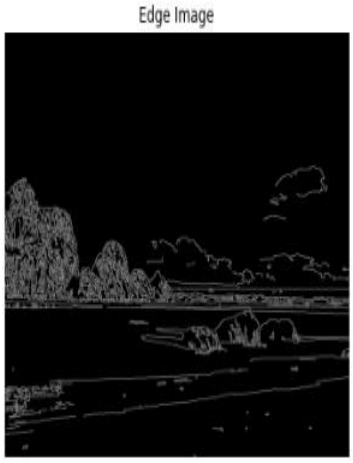
INPUT 1



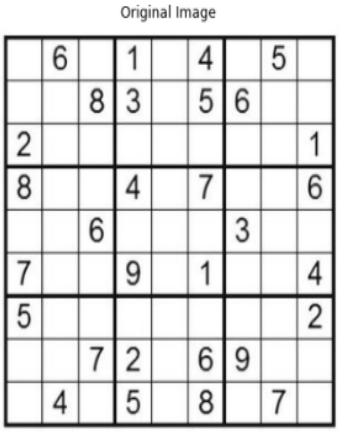
INPUT 2

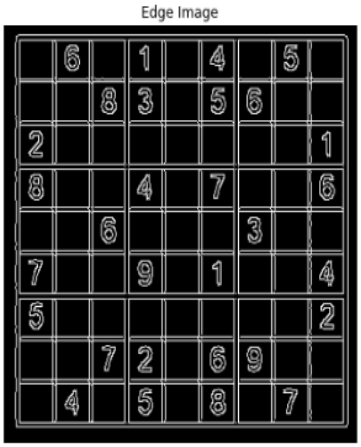


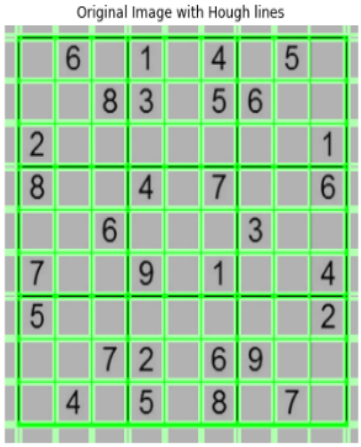
INPUT 3



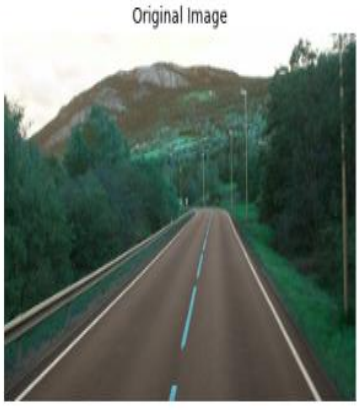
INPUT 4

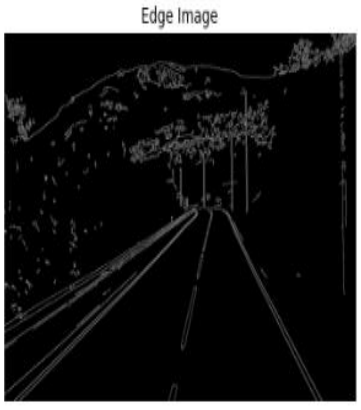


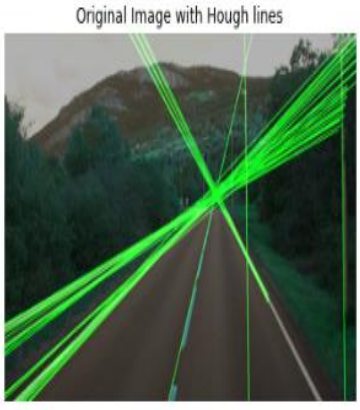




INPUT 5







REFERENCES

1. Wikipedia – Line Detection , Hough Line Transformation

2. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html#additional-resources

3. http://web.ipac.caltech.edu/staff/fmasci/home/astro_refs/HoughTrans_lines_09