CSE 573 - CVIP

Final Project Report

# Hough Transform to Detect Circles

By

Amaan Akhtarali Modak – amaanakh - 50206525

Leila Talebpour – leilatal - 5010523

Project Submitted on: 12/16/2016

# Table of Contents

## A. Literature Review

The Hough Transform is a feature extraction technique used in image analysis, computer vision and digital image processing[1]. This method uses voting procedure to find the instances in the objects within many types of shapes, even though the instances are not perfect.

Hough Transform was initially invented to determine the set of parameters that would describe a trajectory of the particles obtained in experimental research conducted at CERN. At first, HT was only used to as a method for detecting straight lines in an image. After that the transform for analytical curves was used. Then the Hough Transform was developed in terms of its application for solving many problems that appeared in computational technology. as a result of studies a number of variations of computational technology have been invented, they are: Fast Hough Transform, Fuzzy Hough Transform, Hough Transform with Varying resolution, Hierarchical and randomized Hough transform[2] but the Hough Transform is most commonly used in image processing as a method for extracting circles and semi-circles which we are mostly focusing on, in this project.

The Hough transform can be used to determine the parameters of a circle when a number of points that fall on the parameter are known[3]. The main advantage of Hough transform is its capability to detect shape of the objects, even if they are occluded. There are also some weaknesses in the existing Hough transform methods regarding circle and semi-circle detection which are: Large memory space used, slow speed processing when voting is executed in every pixel, high computation and storage requirement, the detection is easy to be missed.

## B. Technologies Related to This Project

To perform Hough transform for circle detection in this project, there were a number of pre-processing steps we needed to take, which requires using the following image processing technologies and steps:

- First, we did enhance the image quality and de-noise it from possible corruptions. In order to do that, we used a very common de-noising technique. **the Gaussian blur**.
- Second, in order to start the Hough Transform, we needed to have the edges of the image detected, to perform the transform on. For edge detection, we were introduced to two different techniques in class: **Sobel Edge Detector** and Canny Edge Detector.
- To find the best detector, we applied both techniques on the original image, and came up with better results from **the Canny Edge Detection** Technique.

In the following, we introduce the techniques and techniques used in project briefly.

## Gaussian Blur

In image processing, a Gaussian blur is the result of blurring an image by a Gaussian function. It is typically used to reduce image noise and reduce detail. The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen.[4]

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

*Equation 1*

The equation of a Gaussian function in two dimensions is presented in Equation1.

## Sobel Edge Detection

The Sobel edge detector uses the masks shown in Fig. 2 to approximate digitally the first derivatives Gx and Gy and finds edges using the Sobel approximation to the derivatives. [4]

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

Gx

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

Gy

*Figure 1 Sobel Edge Detector Mask*

However, we decided to use Canny Edge Detection to detect all edges in our image as it provided us with a more accurate and efficient output than the Sobel Edge Detector.

## Canny Edge Detection

Canny operator is based on three criteria.[5] The basic idea uses a Gaussian function to smooth image firstly. Then the maximum value of first derivative also corresponds to the minimum of the first derivative. In other words, both points with dramatic change of gray-scale (strong edge) and points with slight change of gray-scale (weak edges) corresponds to the second derivative zero crossing point. Thus these two thresholds are used to detect strong edges and weak edges.

We used inbuilt modules for this purpose. These are the steps that are taken in the canny edge detection algorithm

- Apply Gaussian filter to smooth the image in order to remove the noise
- Find the intensity gradients of the image
- Apply non-maximum suppression to get rid of spurious response to edge detection
- Apply double threshold to determine potential edges
- Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

After these pre-processing, we applied the actual Hough Transform with the following algorithm.

## C. Project Methodology

The Hough Transform is a feature extraction technique used in image analysis and computer vision for detecting circles and semicircles. In order to do so, some pre-processing needs to be applied to the original image. In the following sections, we will first propose the main work flow and the procedure of the work in our project, then we will give an algorithmic illustrative description of Hough Transform method for circle detection. And then we will continue by explaining how we implemented the project and code description will follow.

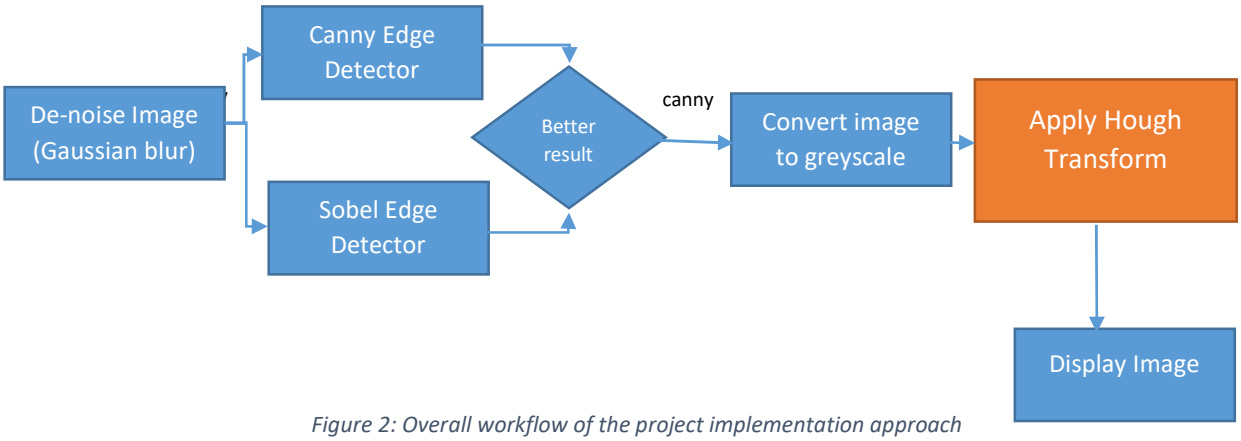


*Figure 2: Overall workflow of the project implementation approach*

## D. Our Approach

As described before, we used Gaussian blur to de-noise image, then we applied canny edge detection technique for getting the edges, since this method had better results, compared with the Sobel Edge Detector, which we also applied on image. For this pre-processing techniques we used the built-in methods in python. By doing so we completed the pre-processing of the image, and the image was ready for the actual Hough Transform method implementation.

We implemented the Hough Transform using an accumulator and voting. Our threshold is set to 40, which is the minimum votes required for considering circles.

To find the best radii, we iterated over a pre-defined range called maximum and minimum. By iteration we chose the radii with best accuracy result.

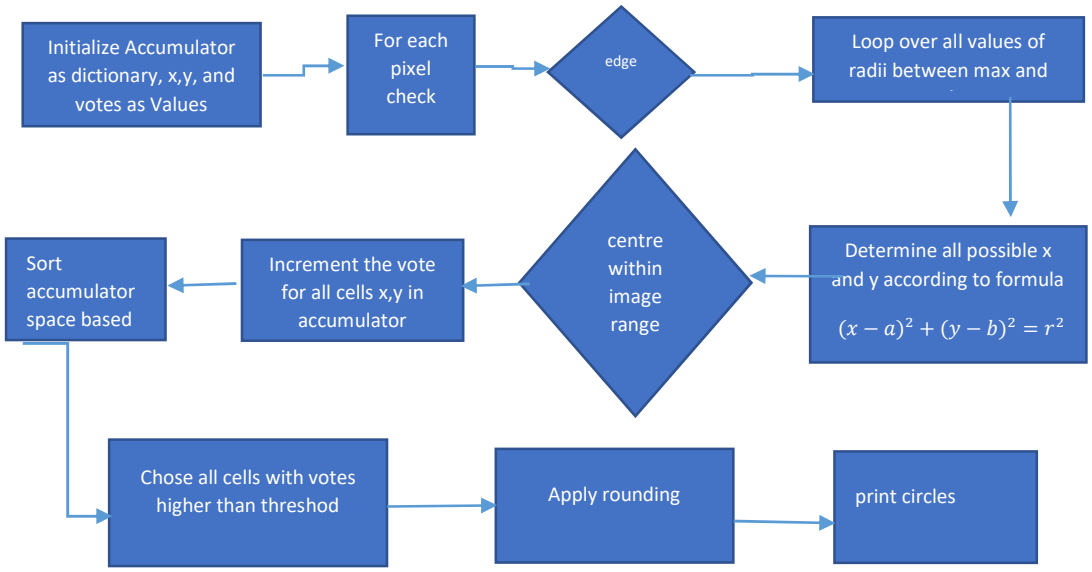


*Figure 3 Hough Transform for Circle Detection flowchart. Used in our implementation*

## E. Outcomes and Deviations

Based on the implementation of our algorithm to detect circles in an image using the principle of Hough Transform on the test image provided to us, we are able to get the output images depicted below which shows the circles being detected pretty accurately.

The major change that we made to try and make the implementation of the project more dynamic and not only for the test image was by first sorting the accumulator based on the number of votes present, in decreasing order, and then picking the entries that have the maximum number of votes first.

If we implement Hough Transform directly, then there is a heavy reliance on defining the optimum threshold for the operation and this value will fluctuate greatly when the input image is changed. By making this modification, we have attempted to make the algorithm more generic and also make it such that it is more accurate over all images rather than simply making it 100% accurate only for the given image.

*Figure 4: Output of Test Image when Threshold = 40*

*Figure 5: Output of Test Image when Threshold = 60*

For the given test image, the amount of time taken by the algorithm to finish and detect the circles in the image was approximately 50 seconds.

***Time taken for execution 50.2799999714***

From this project algorithm, we were able to learn how specific curves can be identified and detected from any image, using the concept of an accumulator array and the process of voting.

There is still some work required to make our approach even more accurate and efficient in terms of performance for large images, but this is a start in the right direction to correctly and efficiently identify a range of circles from any given image.

## F. Explanation of software and program Development

We implemented our project using python, using inbuilt functions from openCV library.
Based on the flowcharts shown in figures 2 and 3, we have implemented our project, by performing pre-processing of the image and then applying Hough Transform.
We will give brief description of the functions and parameters used in our code in the tables given below.

Parameters:

| ***Parameters*** | ***Description*** |
|---|---|
| *start_time* | *Parameter defined for start time of algorithm.* |
| *end_time* | *End time of Algorithm.* |
| *edged_image* | *Stores the result of canny edge detection on image.* |
| *height, width* | *Height and width of image used in Hough Transform.* |
| *Rmin, Rmax* | *Minimum and Maximum range of Radius.* |
| *accumulator* | *Our initial accumulator, a 3 dimensional array storing x, y, v, where v is the vote value.* |
| *thresh* | *Minimum votes required to be considered a circle in output.* |
| *sorted_accumulator* | *An instance storing the values of initial accumulator storing only the values larger than threshold, and in descending order.* |
| *redundant_circles* | *This is used to store the redundant circles based on rounding.* |
| *roundr* | *Round off radius. To ignore circles with similar radii with same centre.* |
| *roundc* | *Round off centre. To ignore circles with nearby centre with similar radius.* |
| *time_taken* | *Total time taken starting from taking image as input to performing Hough Transform with default threshold.* |

Functions:

| **Functions** | **Description** |
|---|---|
| *display()* | *Function defined to display image* |
| *output_circles()* | *Function defined to draw circles* |
| *cv2.GaussianBlur()* | *Gaussian Blurring of original image* |
| *cv2.Canny()* | *Applying canny edge detector on image* |

## G. Summary and Discussion

### *i. Summary of Project*

The basic idea behind this project is to use the techniques and concepts learned in class to successfully execute Hough transform for detecting circles given in an image. We were given a random image containing circles and we were required to detect all the circles present in the image using Hough Transform algorithm. We were expected to successfully build the Hough transform algorithm from scratch using the information we have learned from this class over the course of the semester.

The main objective of this project was to test our knowledge of the key concepts in image processing and to help us understand the intricate working of the various techniques involved in computer vision.

### *ii. Lessons learned in this course*

This course did provide us with a wide range of knowledge regarding Computer vision and image processing techniques.

During the course of learning the Fourier transform, and convolution, we learned how to take image from the space dimension into the frequency space. Knowing median filtering, Gaussian filtering, high-pass and low-pass filters gave us an understanding of how to process images. Image segmentation and edge detection, along with image understanding gave us the ability to detect, segment and label objects in images and relate them to real world components.

## References


1. Shapiro, L. and G.C. Stockman, *Computer vision. 2001.* ed: Prentice Hall, 2001.
2. Gomółka, Z., et al., *The use of the Circular Hough Transform for counting coins.* Measurement Automation Monitoring, 2015. 61.
3. Rhody, H., *Lecture 10: Hough circle transform.* Chester F. Carlson Center for Imaging Science, Rochester Institute of Technology, 2005.
4. Savant, S., *A review on edge detection techniques for image segmentation.* Int. J. Comp. Sci. Inform. Technol, 2014. **5**(4): p. 5898-5900.
5. Canny, J., *A computational approach to edge detection.* IEEE Transactions on pattern analysis and machine intelligence, 1986(6): p. 679-698.