CS512 Assignment 3 Report

Ronit Rudra
A20379221

October 22, 2017

Abstract

This report describes the implementation of a Line Detection Program Using Hough Line Transform. The desired outcomes of the framework, the problems faced, the implemented solutions would be discussed in the upcoming sections.

1 Problem Statement

The objective was to create a simple program which would take in inputs from the webcam and perform Line detection by applying the Hough Transform.

The program should have the following:

- Would be run from the command prompt/terminal
- Display the binary image after edge detection
- Draw the detected lines in color
- Draw refined lines after LSE approximation in a different color
- Color the points belonging to a line in a different color.
- Show the Hough Plane.

The next section describes how the program was structured and implemented

2 Proposed Solution

The Program was split into two parts, the UI and the Hough Transform module.

The UI is responsible for the following:

- Capturing images from the camera
- Setting up Parameter TrackBars and Image Windows
- Calling functions for processing input

The Hough Transform Module does the heavy lifting by finding line parameters from the processed image. It does the following:

- Convert points to polar coordinates
- Accumulate votes in Hough Plane
- Cache set of points belonging to a line
- Perform culling by threshold values

The program was implemented in *Python 3.6.1* with *OpenCV 3* and it made sense to implement Object-Oriented Programming to make the program as modular as possible.

The next section details the implementation of the program.

3 Implementation Details

The program is a set of the following files:

main.py The main file which contains the script. This is the file that should be run when a user wants to run the program.

houghTransform.py Python version of the hough transform. [Extremely Slow. Not Used]

fastHough.pyx This is the cython compiled file which performs the Hough transform. Imported by main.

fastHoughv2.pyx A more accurate but slower version of the above. Imported by main.

speedtest.py Comparison of Different Hough implementations. Run this to check how fast the different implementations are.

test_suite.py Performs Hough Transform on image passed as command line parameter and saves result in *data* folder. See Subsection on testing.

setup.py and setupv2.py Setup files for cythonizing functions. Run as *python setup.py build_ext –inplace*

3.1 Program Flow

The process flow of the program is briefly described below:

1. Read Image from webcam
2. Convert to GrayScale
3. Perform Canny Edge Detection and Binarization
4. Find Line Parameters by Hough Transform
5. Plot Lines on the Image
6. Use Least Squares Approximation on line points to refine line parameters
7. Draw refined lines
8. Color points belonging to detected lines
9. Wait for User key press then go to 1 or exit if 'e' is pressed

3.2 The Hough Transform

The Hough Line Transform was implemented in a naive way using nested loops and compiled in cython in order to benefit from C compilation boosts. The Hough function takes in the following inputs:

Image The binary edge image as a result of Canny Edge Detection

Width The width of the image

Height The height of the image

Rho Resolution The minimum step in the distance axis of the parameter plane

Theta Resolution The minimum step in the angle axis of the parameter plane

Threshold Minimum number of votes for a parameter pair to be accepted as a line

The outputs of the function are as follows:

Accumulator An 2D array of the votes gathered by the hough transform

Rhos A 1D array of the possible distance values

Thetas a 1D array of the possible angle values

Acc_Votes A dictionary with the line parameters as keys and a list of points who voted for it as the values. Only contains parameters who are above the threshold

The θ values range from 0 to π and the ρ values range from negative of the image diagonal to it's positive value. The Theta Resolution and Rho Resolution parameters govern how the range is spaced.

This a tradeoff between accuracy and processing time.

The Accumulator is a 2D array with the first dimension of length equal to the ρ values and the second dimension equal to the length of the θ values.

The Function does the following:

1. Using Theta Resolution and Rho Resolution, generate an equal spaced array for θ and ρ
2. Generate an accumulator using above and initialize to zeros
3. For each pixel of the image iterate through each θ and calculate:

$$\rho = x\dot{c}os(\theta) + y\dot{s}in(\theta) \tag{1}$$

4. Find Closest ρ to this value in the Rhos array and increment the corresponding (ρ, θ) index of the accumulator
5. Add the pixel to the list of points belonging to the parameter (ρ, θ)
6. Use threshold to cull the parameters low votes.
7. Return output values

The function benefits from a number of speedups:

1. Cythonizing the function has a tested speedup of 11 times on average
2. Instead of calculating the sine and cosine values every iteration, the values are calculated once and cached.
3. Instead of iterating through each pixel of the image, only the edge pixels are chosen using *np.nonzero()*

3.3 Testing

The python file *test_suite.py* should be run to test the performance of the Hough Transform. The program takes in the following command line arguments.

Image Path Image should be in the *data* folder. Hence, the path should be like: *../data/filename*

Hough Function Either of the following : *-p* : Python Hough or *-v1* : Cython Hough version 1 or *-v2* : Cython Hough version 2

Canny Threshold 1 Integer specifying lower threshold of canny edge detection

Canny Threshold 2 Integer specifying upper threshold of canny edge detection

Rho Resolution Distance spacing of hough accumulator

Theta Resolution Angle Spacing of hough Accumulator

Hough Threshold Voting Threshold of accepted Line

The output would be an image with the detected hough lines drawn in red, refined lines drawn in green and line points drawn in yellow. The result would be saved automatically in the data folder with the name as current timestamp.

4 Results and Discussion

Although the resulting lines from my implementation of the Hough Transform were close to the in-built function, even with cythonizing the function, the performance was an order of magnitude slower. This was due to:

1. Multiple nested loops and conditional statements.

2. Cython cannot handle numpy functions well and reverts to python implementation. This creates a bottleneck. One can use *cython filename.pyx -a* to generate an html document of the cythonized function. Any codeline using python is highlighted in yellow. A highly optimized Cython code would not have any python interaction.

3. Problems in handling negative distance values. As per the implementation, negative ρ means the line intersects the y-axis above the origin. Handling it resulted in obtuse angled lines not being detected and drawn.

4. Storing the points belonging to a line had to be done using a dictionary which slows down the cython code but was the only way to efficiently store the coordinates. The other method was to find euclidean distance of each point from each line and then assign the point to the closest line which is computationally highly expensive.

Figure 1: Detected Hough Lines

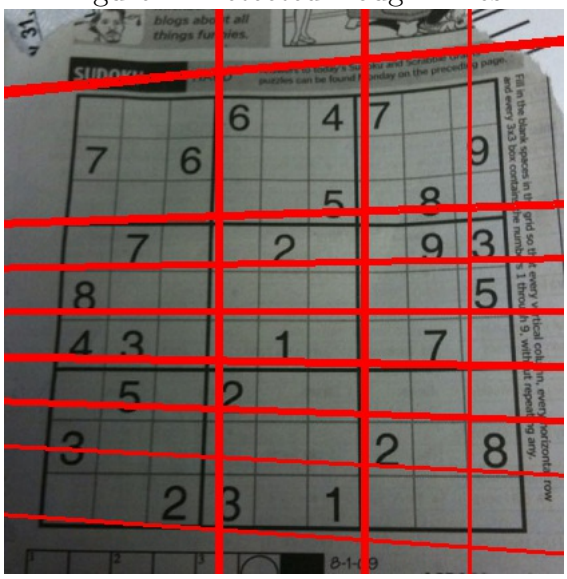


Figure 2: Refined Hough Lines after LSE

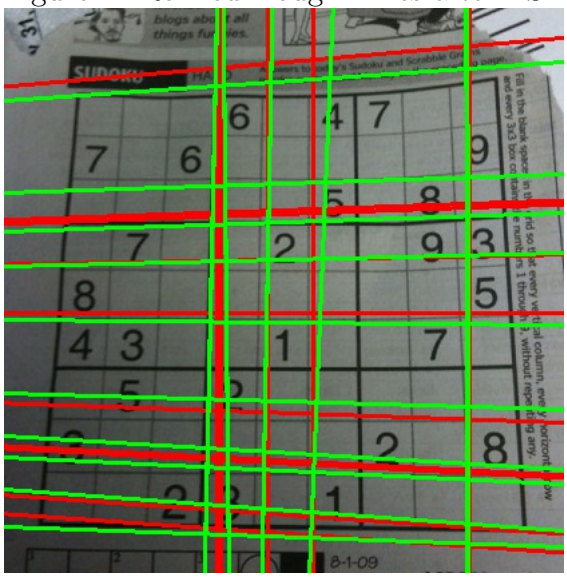
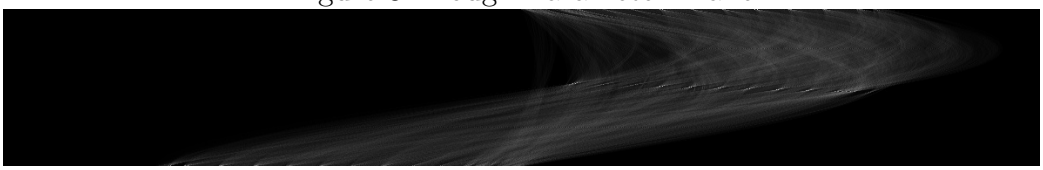


Figure 3: Hough Parameter Plane



5 References

1. http://aishack.in/tutorials/hough-transform-normal/
2. https://docs.opencv.org/3.2.0/d6/d10/tutorial_py_houghlines.html
3. https://alyssaq.github.io/2014/understanding-hough-transform/
4. https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html
5. https://docs.opencv.org/3.1.0/da/d22/tutorial_py_canny.html
6. http://www.geeksforgeeks.org/line-detection-python-opencv-houghline-method/