CS512 Assignment 3: LINE DETECTION USING HOUGH TRANSFORM

Adithya Sreenath Chandrashekarapuram

Department of Computer Science

Illinois Institute of Technology

October 21, 2017

Abstract:

Lines present on an image can be detected by using a technique called Hough Transform. This technique uses a binary edge image as an input and detects lines present on the image by using a voting technique. Each edge pixel is checked and converted to a line in a parameter space for evaluation. Intersections of the lines are found and votes are estimated. This method of line detection can take a long time to execute as the execution time increases exponentially depending on the parameters.

Proposed Solution:

The implementation of Hough line detection has two parts:

1. Detection of the Hough lines
2. Drawing of the lines on the final image

The Hough lines algorithm:

```
HoughLines(image,angle,threshold)
    Initializing final_matrix,image_parameters(y_idxs,X_idxs)
    Computing theta value
    Computing Cos(theta),Sin(theta)
    For t in range(len(theta))
        For I in range(y_idxs.size)
            r = round(x_idxs[i]*cos_t[t] + y_idxs[i]*sin_t[t])
            if (r,thetas[t]) in final_matrix:
                final_matrix[(r,thetas[t])] + 1
            else:
                final_matrix[(r,thetas[t])] = 1
    Check for threshold and update the final_matrix
    return final_matrix
```

The Draw Lines algorithm:

```
Drawlines(final_matrix)
    for line in lines:
        rho = line[0]
        theta = line[1]
        Compute Cos(theta),Sin(theta)
        Compute x0,y0
        Compute x1,y1,x2,y2
        Draw lines on image based on x1,y1,x2,y2 values
Display the final image with the Hough lines
Exit
```

This algorithm provides the exact same result as the open cv function cv2.HoughLines. The algorithm uses loops to iterate the statements and takes up considerable amount of time to compute. The HoughLines function is used to get the accumulator matrix which consists of the voting results and the draw lines function is used to take those voting results and draw the Hough lines on the image.

Implementation:

The program is designed in python 2.3 with the usage of numpy and opencv functionalities. The program allows for video capture and detection of lines simultaneously on the images. It consists of a Trackbar which provides the user with options to change the peak detection threshold, edges detected and the bin size.

The program takes considerable amount of time to execute as there are numerous loops involved to compute the Hough transform matrix. This problem can be fixed by converting the code to run on cython which would greatly reduce the time taken to execute the code.

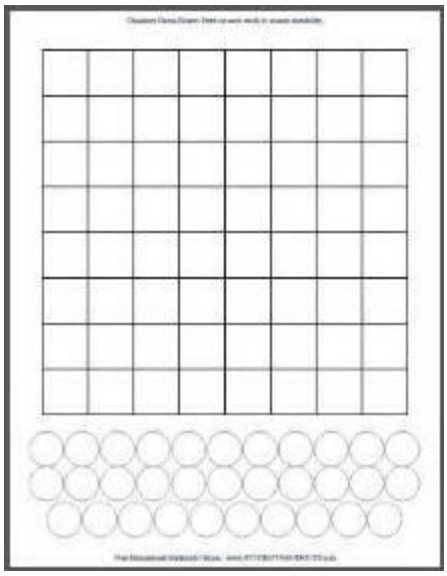
Manual:

There are two programs provided:

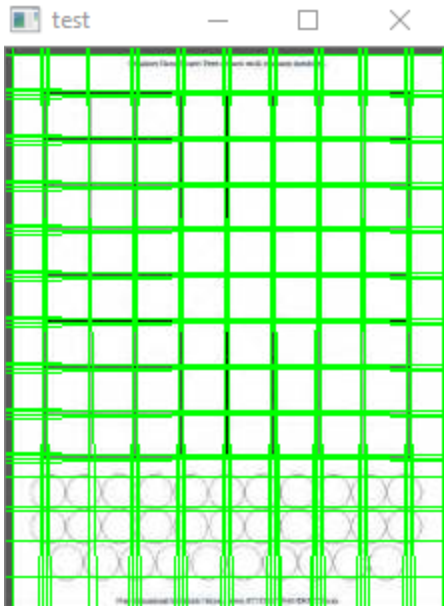
1. Hough_Video:
    - This code consists of the video capture and continuous image processing to detect lines.
    - The frame gets captured as soon as any of the 3 trackbars are moved.
    - The user can change the values to test the frame.
    - The ESC key should be pressed to recapture another frame.
    - Pressing the ‘q’ key will exit the program.
2. Hough_MyImplementation:
    - This code is the demonstration of the algorithm which performs just like the cv2.HoughLines function.
    - This code takes a small image file and computes the lines present on it.
    - The image file can be changed to test for correctness.
    - Note: Larger files have extremely long execution time as cython has not been used. The image provided takes around 15-20 mins to execute.

Results and Discussion:

Input image:



Output image:



This is the output result after running the Hough_MyImplementation.py program.

Evaluation:

Time:

The time taken for execution of the opencv function cv2.HoughLines:

```
(C:\Users\adith\Anaconda2) C:\Users\adith\Desktop\AS3>test.py
0.00791151554945
```

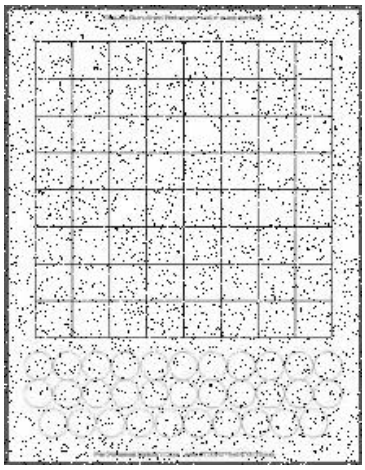
The time taken for execution of the provided algorithm:

```
(C:\Users\adith\Anaconda2) C:\Users\adith\Desktop\AS3>test.py
2121.18004011
```

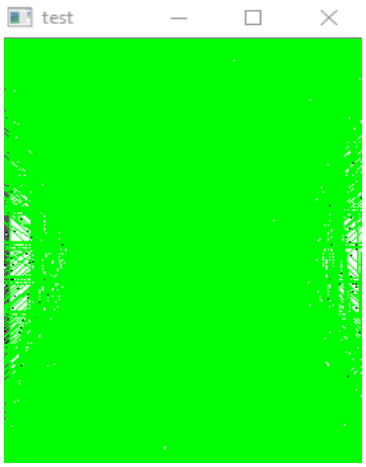
We can conclude by stating that the opencv function is more efficient and easier to use than the provided algorithm. Furthermore the computation time of the algorithm can be reduced by using cython to compile the code bits and make execution faster.

Noise:

Input image:



Output image:



Therefore we can see that when we add salt and pepper noise to the image the result can be completely different. Hence we can conclude that noise causes problems while detecting lines and the images should be smoothened out to reduce or remove the noise before performing line detection.

References:

1. https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html

2. https://en.wikipedia.org/wiki/Hough_transform

3. http://www.geeksforgeeks.org/line-detection-python-opencv-houghline-method/