IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 1, January 2016

An Investigation of the Costs and Benefits of Thinning on the Straight Line Hough Transform

Gideon Damaryam[1], Emeka Ogbuju[2], Haruna Abdu[3]

Department of Computer Science Federal University, Lokoja, Kogi State, Nigeria[1,2,3]

Abstract: This paper presents the outcome of an investigation of the costs and benefits of thinning as part of pre-processing for line detection including specification of end-points, from visual images of indoor rectilinear environments. This is done as part of a bigger process with the goal of detecting lines to enable a small mobile robot self-navigate within the environment based on navigationally important features such as doors and corridors reconstructed from the lines detected. The straight line Hough transform is used to determine parameters which specify gradients and positions of lines, and then the end-points of the lines are determined. To do this images can be pre-processed to the point of edge-detection which typically yields edge lines several pixels thick, or edge-detection followed by thinning yielding edge lines about a single pixel in thickness. Since the Hough transform operates on all pixels in an input image, more work is needed to process the "unthinned" image. However, thinning itself takes time. This paper looks into whether the taking the time to do thinning is justified in terms of overall time taken, and quality of resulting lines found, and concludes that for the purpose described, thinning does appear to improve the quality of line detection, while taking less total time to do it.

Keywords: edge-detection, Hough transform, line detection, processing time, thinning.

I. INTRODUCTION

This investigation presented in this paper was done as part of work to develop a system to enable a mobile robot self-navigate within rectilinear environments on the basis of visual input, to achieve navigation an image is captured by a forward facing camera mounted on the robot, the image is processed, and then some navigation is effected based on the result of the processing. This cycle is repeated until a predefined navigation program is completely executed, or the navigation process is otherwise terminated.

Processing an image essentially means recognition of high-level, navigationally important features such as corridors and doors within the image after necessary pre-processing. Recognition of features, for the purpose of this work, involves a number of stages. The stages include establishing parameters that uniquely specify straight lines, finding valid sub-lines of the lines and their end-points, and then looking into which sub-lines constitute navigationally important features.

Determining parameters that specify straight lights in the image is referred to as line detection and is done in this work using a process called the Hough transform. It yields parameters that specify the equation of the lines of interest, but not information about the actual starting and ending points of the line, hence the need for a separate process to determine end-points. Before application of the Hough transform, the image needs to be pre-processed.

Details of pre-processing, the Hough transform and determination of end-points have been presented in [1], [2] and [3] respectively. Important details relevant to this work are highlighted in the sub-sections "A. Pre-Processing", "B. Line detection using the Hough transform" and "C. End-points Determination" which follow. "D. Investigation Objective" then puts the investigation presented in this paper into perspective.

Briefly though, "A. Pre-Processing" will show that thinning is an optional process in pre-processing since the specification of lines presented in" B. Line Detection using the Hough Transform" can happen with or without it. The investigation presented looks closer into the benefits and costs of thinning and not thinning, and whether the benefits ultimately justify the cost. Benefits could include quality of results of sub-lines detection, and potential saving of processing time in later stages of processing. Cost could be loss of quality of sub-lines detection, and additional processing time. A known fixed cost is the processing time to do the thinning.

A. Pre-Processing

Details of the preprocessing scheme used for this work have been presented in [1]. A summary of the scheme highlighting details relevant to this paper follows.

In this work, after images are captured, in preparation for feature detection, they are resized to a 128 pixel by 96 pixel size and converted to gray scale. The Sobel edge detection filters are then applied to determine edges or prominent regions within the image. This is called edge-detection, and results in binary image with black pixels showing areas that are edges, i.e. boundary areas of prominent regions, and white pixels showing the body of those regions.

Fig. 1(a) shows a typical image after it is resized, and Fig. 1(b) shows the same image after edge-detection.

For the purpose of using the Hough transform to find parameters that uniquely specify lines from the image, a binary image is needed. Edge-detection provides such an image, so it is possible to go straight into application of the Hough transform after edge-detection. Works such as [4] do so.

Edge-detection typically results in images that are several pixels thick. Because the Hough transform operates on individual pixels, images resulting from edge-detection give the Hough transform many more pixels to process than if edges were of unit thickness. The Hough transform is a processor (and memory) intensive activity. This could mean that much more computer time is taken than may be necessary, and for a real time system such as generating information necessary for robot self-navigation, this would be undesirable. Processing the extra pixels also “distracts” feature detection processes down the line from important but salient features of the image. Some works have gone ahead to first “thin” the edges.

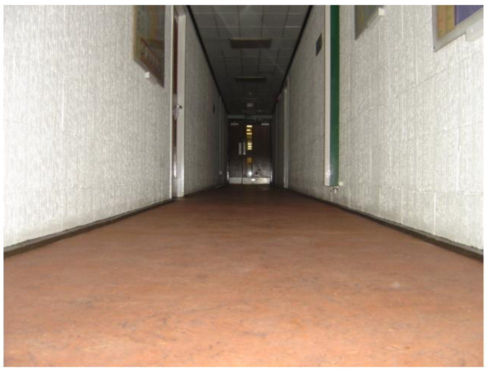


Figure 1a: Sample Image

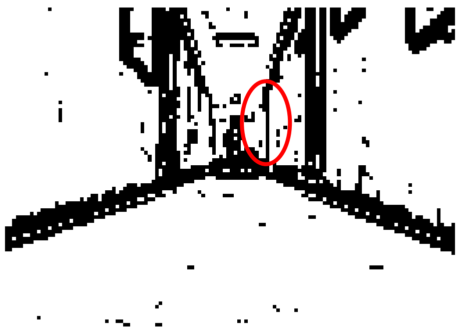


Figure 1b: Sample image after Edge-detection

Thinning is the process of reducing the thickness of edges to as close to unit thickness as possible, while retaining information about all important edges. [1] discusses several thinning method and details of a particular method used for this work. Fig. 2 show the result of thinning for the image from Fig.1

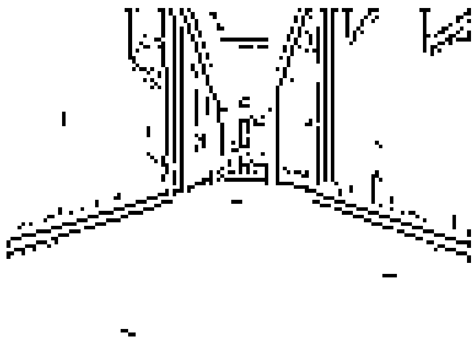


Figure 2 Sample image thinned with modification

The process of thinning does use up processing time itself. This paper looks into whether doing thinning, and the cost associated with it, on the whole, helps or hurts feature detection for the robot self-navigation endeavor.

B. Line Detection using the Hough transform

Details of the Hough transform and how it is used to detect lines in the work discussed in this paper are available in [2]. Summarily, the Hough transform is an image processing technique invented to enable automatic recognition of lines [5]. The normal (or polar) form of the equation of a straight line

$$\rho = x\cos\theta + y\sin\theta \quad . \quad . \quad . \quad \text{Equation 1}$$

has become more popular, and was used in this work. This approach was first proposed by [6]. It is based on the principle that for every point (x, y) in the x-y plane, there is a curve defined by equation 1 in the $\theta - \rho$ plane.

Equation 1 is used to transform every edge pixel in image space to a curve in parameter space. To determine the transform for each edge pixel, various values of θ are taken at a regular interval.

Closely associated with parameter space is the accumulator array. It is a two dimensional array superimposed over parameter space.

The accumulator array is usually set up as a zero array initially. Individual entries are then increased in the course of application of the transform. The value of each entry is called the accumulation of that entry. How this increase is done varies. The most popular way is to increase the value of each entry by 1 whenever a curve resulting from a transform crosses it. This approach is used in this work. Other approaches exist and are discussed in [7] and [2].

With the chosen method of accumulation, when all edge pixels have been processed, accumulator array cells will have varying values. Those that have not been affected by the transform for any edge pixel will still have their initial value of 0. All others will have positive integer values equal to the number of times a transform curve has crossed them. Each transform curve is a result of transform of a point from image space, so each crossing added to a cell means one more pixel was found in image space from the line corresponding to the cell in the accumulator array. Some accumulator array entries will have higher values than their neighbors and are referred to as peaks. Peaks in accumulator array represent lines that have the highest evidence of actually existing in image space. Peak detection is the process of determining which cells are peaks.

Peak detection can be achieved by application of a threshold. All accumulator array entries above a certain threshold automatically qualify as peaks and all others are not peaks. A method to automatically determine the right threshold to use for peak detection based on a target number of peaks was used. Choice of the target number of peaks is presented in [9]. Further discussion on selection of thresholds is available in [2].

In many situations, peaks are not easily distinguishable. This can be due to a number of possible situations. An example is a situation where several accumulator array

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 5, Issue 1, January 2016

entries in a neighborhood have values higher than the edge detection threshold. Although one of them is higher than all others, several points above the threshold that are not actual peaks are returned by the application of the threshold. [2] refers to them as false peaks.

Where they exist, elimination or at least minimization of false peaks is necessary. In [2], the reduced butterfly filter proposed by [8] is used to minimize false peaks. It is a 3 x 3 convolution filter which when applied to an accumulator array will emphasis actual peaks while suppressing false peaks.

To minimise the detection of multiple peaks in parameter space due to a single line from image space, peaks are eliminated if their butterfly filtered value is not higher than that of all other entries within a 5 x 5 pixel neighbourhood surrounding them. Peaks which are local maxima within the 5 x 5 neighbourhood surrounding them are the target significant lines from the original image and they are processed further.

When peaks have been correctly determined, the parameters and associated with them uniquely specify lines in the original image that can be interpreted to be the most important lines from the image. [3] details the mathematics involved. The result is that the gradient of the line m is given by

$$m = (-\cot\theta)\ , \quad . \ . \ . \quad \text{Equation 2}$$

and intercept on the y axis is given by

$$c = \cos ec\,\theta\ , \quad . \ . \ . \quad \text{Equation 3}$$

As part of efforts to minimise the detection of multiple peaks in parameter space due to a single line from image space, peaks are eliminated if their butterfly filtered value is not higher than that of all other entries within a 5 x 5 pixel neighbourhood surrounding them. Peaks which are local maxima within the 5 x 5 neighbourhood surrounding them are the target significant lines from the original image and they are processed further.

C. End-Points Determination

When lines are found using the Hough transform, there is information about their gradient and intercept, but not about their lengths or end-points. [3] presents a method for determining end-points and lengths of lines when detecting the lines with the Hough transform. In summary, it works by keeps a record of all the points that contributed to the accumulation of each point in the accumulator array, so that for any point in the accumulator array returned as a peak, all the points on the line are known. They are then checked against certain criteria to determine if they make-up valid sub-lines for the line. Sub-lines and valid sub-lines are defined in more detail in [3]. Endpoints of a sub-line are then determined as the points farthest away on both ends of a valid sub-line along the direction of the line, and its length is the distance between its endpoints.

D. Investigation Objective

Sub-sections" B. Line Detection using the Hough Transform" and "C. End-Points Determination" have summarized the key points of a line detection scheme presented in [2] and an end-points detection scheme presented in [3] respectively. Both schemes are part of a vision system for a self-navigation robot based on line detection.

"A. Pre-Processing" already summarized a pre-processing scheme for images in preparation for the line detection scheme of [2], providing two possible types of binary images that can be take of points for line detection. One version stops at edge-detection, and the other version goes further to do thinning. The thinned version has much fewer black pixels than the version which stops at edge-detection, even though it has all the important edge-information there is. This means the thinned version gives the line detection scheme of [2] much fewer pixels to process.

This paper presents an investigation into whether or not thinning should be done when trying to determine sub-lines by finding lines and their end-points. The investigation looks into the costs and benefits of doing thinning, i.e. processing time involved, quality of lines and end-points detected, and savings in time for the subsequent line detection and end-points determination scheme.

II. INVESTIGATION

A simple experiment was done to investigate the effect of thinning on the processing time and quality of results of processes sub-sequent to the pre-processing stage of a mobile robot vision system based around line detection, to the point of detection of end-points of sub-lines.

The processes involved were already summarized in 1 Introduction, and detailed in [2] and [3].

To do this, a typical image was pre-processed to two points - to the point of edge detection, and to the point of thinning. The image is shown in Fig. 3 with navigationally critical lines circled in red. Those are the lines that should be detected found by the system ideally.

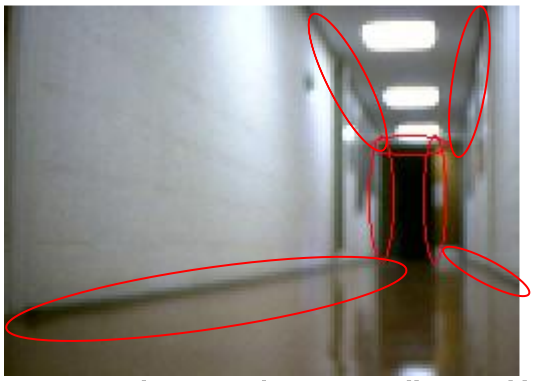


Figure 3. Typical image with navigationally critical lines highlighted

The version with pre-processing done to the point of edge detection was labeled EI and the version pre-processed to the point of thinning was labeled TI for reference purpose. Both EI and TI were then further processed to find lines with the Hough transform end-points of sub lines as presented in subsections B and C of "II Introduction". Both runs were done twice so that random variations in timing can also be monitored.

III. RESULTS AND DISCUSSION

Results for the first and second runs for edge-image EI are labeled EI1 and EI2 respectively. The same labeling scheme was used for thinned image TI. Outcomes of these are discussed in "A. Effects of Thinning on Quality of Results of Subsequent Processes" and "B. Effects of Thinning on Processing Times of Subsequent Processes".

A. Effects of Thinning on Quality of Results of Subsequent Processes

Table 1 summarises results from the experiment to illustrate the effects of thinning. Row 1 of the results shows the takeoff number of pixels after Sobel edge detection, 2491. The next row shows the number of pixels after thinning. This is only applicable to TI as EI was not thinned. Row 3 following that illustrates the two versions of binary images that were used as input to the Hough transform. Lines in EI are noticeably thicker than lines in TI.

Row 4 states the number of peaks found when the peak detection scheme used in this work was applied with a target number of peaks of 200. 236 peaks resulted from TI and 206 from EI. Row 5 shows that 38 and 22 lines respectively resulted when the butterfly filter is applied and local maxima are selected in 5 x 5 neighbourhoods in the way done by this work as discussed in "B. Line Detection using the Hough Transform".

A few interesting observations can be made. Firstly, as stated in row 2, thinning results in a significant reduction of edge points from 2491 to 1008. This represents approximately, a 59.53% reduction. This, as is seen in B. Effects of Thinning on Processing Times of Subsequent Processes which follows, has a significant effect on Hough transform time, and ultimately on the time taken for the entire process up to sub-line detection.

Secondly, sub-lines found in TI appear to match lines in the image that actually represent high level features, to a higher extent than sub-lines in EI. In particular, more navigation critical sub-lines were found in TI than EI as state in row 8 and illustrated in rows 9 and 10 even though more sub-lines were found in EI as stated in row 7. Navigation critical sub-lines are those that would actually enable detection of doors, and corridors possible for the robot, and therefore possibly affect its navigation decisions as discussed in [9]. They are illustrated in Fig. 3.

Tables 2 shows the timings recorded for a number of subsequent processes. An Intel Duo T5600 1.83GHz processor machine with 1GB of RAM was used for this experiment.

Three issues were looked at in terms of processing time - (1) what is the cost of thinning, (2) what benefits can be derived from thinning, and (3) what is the overall effect of thinning. These are discussed in 2) Costs of Thinning, 3) Benefits of Thinning and 4) Sum Effect of Thinning on Time Taken respectively.

Before going there however, a look is taken at errors in the measurement of time in "1) Errors in Time Measurement below".

1) Errors in Time Measurement:

Before discussing the results in relation to the purpose of the experiment, two unexpected observations need to be made regarding the accuracy of the time recording mechanism used to set up table 2. First for some processes 0 milliseconds was recorded as time taken. These include thinning time and time taken to find sub-lines for TI 1, and time taken to find sub-lines for EI 2.

This suggests that the time recording mechanism is limited in its sensitivity. The extent of this insensitivity is not certain but is probably hinted by other times recorded for exactly the same events.

Thinning time for TI 2 was 160 milliseconds, time taken for determining sub-lines for TI 2 was 160 milliseconds, and time taken to determine sub-lines for EI 1 was 150 milliseconds. These suggests that events that have up to 160 milliseconds as time recorded for them to complete may at other times have as low as 0 milliseconds recorded as the time to complete them.

The second related unexpected issue is that exactly the same processes with the same data take different amounts of time to complete at different times. The first two rows of results in table 2 show the time taken to convert the image to grey scale and the time taken to perform Sobel edge detection. The same amount of time was expected for all four runs of the two processes. However, there were variations of up to 10 milliseconds.

Further down the table, other unexpected variations were also observed. The time recorded for peak detection for TI1 was 160 and for TI2 it was 310 milliseconds meaning there was a 150 milliseconds difference. Other unexpected variations were those between butterfly filter application times for TI1 and TI2 (10 milliseconds), between Hough transform times for EI1 and EI2 (150 milliseconds), and between peak detection times for EI1 and EI2 (10 milliseconds). These are in addition to the variations involving 0 times already mentioned. These differences are likely to be due to differences in work load on the processor due to background processes, and related issues at the times that the process times being studied were recorded.

These two observations suggest that errors of up to 160 milliseconds can be expected, and so differences in time of up to that figure should not be taken as significant.

2) Costs of Thinning:

The most obvious cost of thinning would be the time taken for thinning in TI. From row 3 of table 2, the average time taken for thinning is 80 milliseconds although it can be up to 160 milliseconds. Factoring in the error margin adapted in "1) Errors in Time Measurement" above, this time can be ruled as insignificant.

Another way to see it is: Time taken for edge detection on the average is 1090 milliseconds. Time taken for thinning is 160 milliseconds.

Therefore thinning increases binary image derivation by a factor of 160/1090, which is approximately 14.68%.

TABLE I RESULTS OF PROCESSING TO THE POINT OF SUB-LINES DETECTION FROM THINNED AND "UNTHINNED" EDGE IMAGE

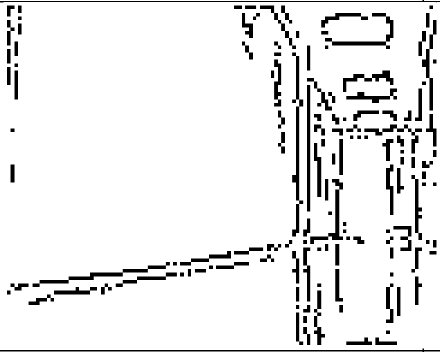
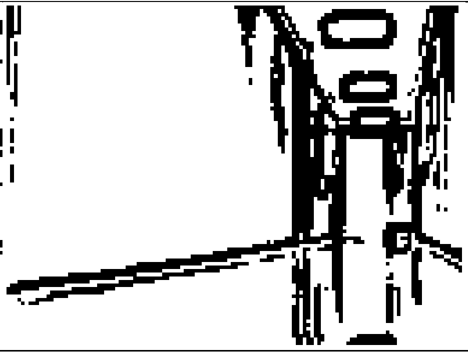
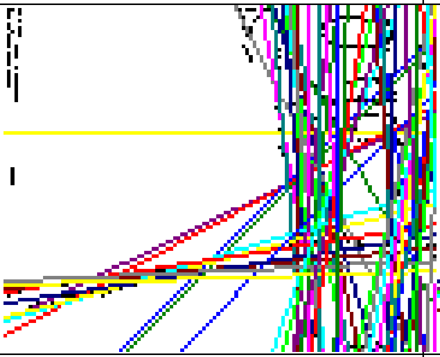
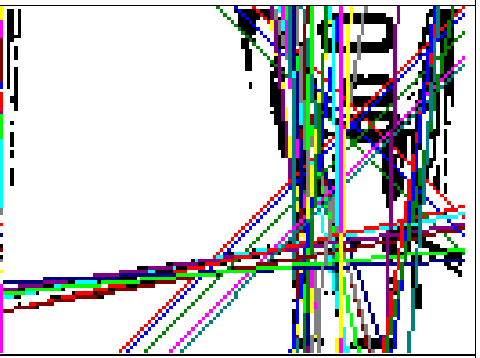
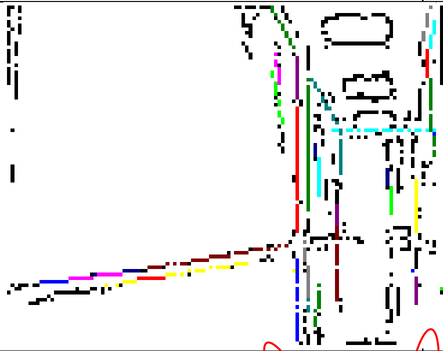
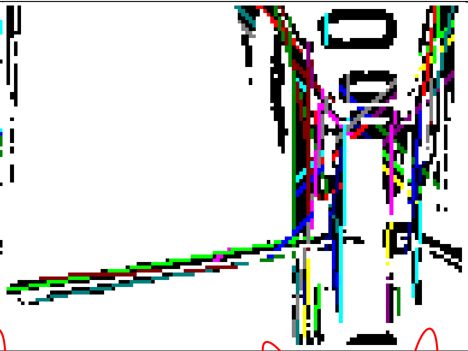
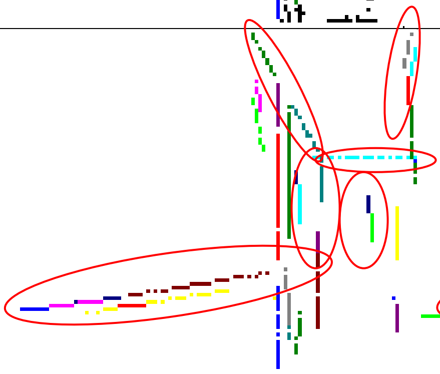
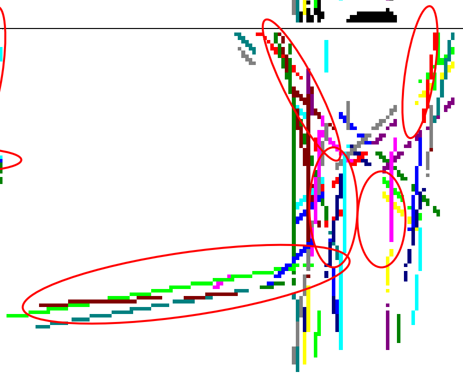
Process		**Pre-processing type**	
		With thinning (TI)	**No thinning (EI)**
1	Number of edge pixels	2491	2491
2	Number of edge pixels after thinning	1008	-
3	Input Image for Hough transform		
4	Number of peaks found	239	206
5	Number of local maxima within 5x5 neighbourhood	38	22
6	Lines corresponding to local maxima in 5x5 neighbourhood superimposed on HT input image		
7	Number of sub-lines found	40	57
8	Number of critical sub-lines found	6	5
9	Sub-lines found superimposed on HT input image		
10			

TABLE 2 COMPARISON OF PROCESS TIMES FOR AN IMAGE WHEN IT IS THINNED AND WHEN IT IS NOT

Process Times (milliseconds)		**Pre-processing type**							
		TI1	**TI2**	**Difference between TI1 and TI2**	**Average of TI1 and TI2**	**EI1**	**EI2**	**Difference between EI1 and EI2**	**Average of EI1 and EI2**
1	Conversion to grey scale time	1100	1090	10	1095	1090	1100	10	1095
2	Edge detection time	1090	1090	0	1090	1080	1090	10	1085
3	Thinning time	0	160	160	80	-	-	-	-
4	HT time	1090	1090	0	1090	2190	2340	150	2265
5	Peak detection time	160	310	150	235	310	320	10	315
6	Butterfly filter application time	150	160	10	155	160	160	0	160
7	Time to find sub-lines	0	160	160	80	150	0	150	75
8	Total time taken	3590	4060	470	3825	4980	5010	30	4995

B. Effects of Thinning on Processing Times of Subsequent Processes

Other than thinning itself, there are only two other processes for which TI records higher time than EI. One is edge detection (row 2) where there is a difference of 5 milliseconds. This is insignificant as well as irrelevant because edge detection happens before thinning. The other is the time taken to find sub-lines (row 7), again higher by an insignificant 5 milliseconds. Although the difference is insignificant, it is not unexpected that TI records higher time here than EI as the number of input lines for that process 38, is higher than 22 for EI as row 5 of table 1 shows.

1) Benefits of Thinning:

In table 2, the most significant saving of time due to thinning happened with the application of the Hough transform where EI required on the average 1175 milliseconds more than TI. Put another way, a saving of 1175 milliseconds represents a saving of about 52% of the 2252 milliseconds taken by the Hough transform for EI on the average. This is expected as the number of times the core of the Hough transform algorithm runs is directly proportional to the number of edge pixels in its input image. TI had 2495 – 1008, i.e. 1487 fewer edge points than EI (row 3 of table 1). This represents a 60% reduction and correlates quite well with the 52% savings on time.

The process whose average EI time is higher with the next highest value is peak detection. It is higher by 80 milliseconds, which by the error margin established in subsection A.1) of section III, is not significant. Other processes for the thinned TI recorded lower times include conversion to grey scale – a 5 millisecond difference that is irrelevant as the process happens before thinning - and butterfly filter application also with an insignificant 5 millisecond difference.

2) Sum Effect of Thinning on Time Taken:

Thinning had no significant effect on time taken for individual processes except for the Hough transform. This effect is so significant it reflects in the average total time row of table 2. There is a difference of 1170 milliseconds, which is very similar to the 1175 millisecond difference of the Hough transform. 1170 milliseconds represents (1170/3825)%, i.e, 30.59% of the total time taken by TI. Thinning makes a significant difference to the amount of time taken.

IV. CONCLUSION

Thinning does reduce the number of edge-points that the Hough transform has to act on. From the current investigation, the reduction is by about 60%. Errors in measurement of up to 160 milliseconds were observed from the time recording scheme used. Time taken for thinning is within the error of time measurement, and on average, is about 15% of the total time to derive binary image from gray scale image. Time taken to perform the Hough transform on the thinned binary image is significant, and 52% lower than time taken to perform the Hough transform on unthinned binary image. This correlates quite well with the 60% reduction in the number of black pixels from the unthinned image to the thinned image. The thinned image takes about 30% less time to process altogether, than the unthinned image. The difference in time is significant. In other words, thinning does reduce the overall time to detect lines and find sub-lines, despite the time taken to do thinning itself.

ACKNOWLEDGEMENT

This paper discusses work that was funded by the School of Engineering of the Robert Gordon University, Aberdeen in the United Kingdom, and was done in their lab using their robot and their building.

REFERENCES

[1]. G. K. Damaryam and H. A. Mani, A Pre-processing Scheme for Line Detection with the Hough Transform for Mobile Robot Self-Navigation , In Press, International Organisation for Scientific Research – Journal of Computer Engineering, 18(1), 2016

[2]. G. K. Damaryam, A Hough Transform Implementation for Line Detection for a Mobile Robot Self-Navigation System, International Organisation for Scientific Research – Journal of Computer Engineering, 17(6), 2015

[3]. G. K. Damaryam, A method to determine end-points of straight lines detected using the Hough transform, International Journal of Engineering Research and Applications, 6(1), 2016

[4]. D. L. Vaughn and R. C. Arkin, Workstation Recognition using a Constrained Edge-based Hough Transform for Mobile Robot Navigation, 1990.

[5]. P. Hough, Method and Means for Recognising Complex Patterns, United State of America Patent 3069654, 1962.

[6]. Duda and Hart. 1973. Pattern Classification and Scene Analysis (New York: Joh Wiley and Sons, 1973).

[7]. A. Low, Introductory Computer Vision and Image Processing, (London, United Kingdom: McGraw-Hill Book Company, 1991).

[8]. J. F. Boyce, G. A. Jones and V. F. Leavers, An implementation of the Hough transform for line and circle location. Proc, SPIE Inverse Problems in Optics, The Hague, Netherlands, 1987.

[9]. G. K. Damaryam, Visions systems for a mobile robot based on line detection using the Hough transform and artificial neural networks, doctoral diss., Robert Gordon University, Aberdeen, United Kingdom, 2008.