IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 6, Ver. VI (Nov – Dec. 2015), PP 33-44
www.iosrjournals.org

A Hough Transform Implementation for Line Detection for a Mobile Robot Self-Navigation System

Gideon Kanji Damaryam
Federal University, Lokoja, PMB 1154, Lokoja,Kogi State, Nigeria.

Abstract: *This paper presents an implementation of the Hough transform for detecting straight lines within rectilinear environments for further processing towards automatic recognition of a path by a mobile robot for the purpose of self-navigation. The results of the processes described in this paper are linesalong whichnavigationally important features from the original image would lie, but whose end-points and lengths have not been determined yet. The input to the processes in this paper is a pre-processed binary image with pixels showing the outline of important edges determined by a scheme discussed in [1]. The processes described in this paper that translate the pre-processed image to detected lines include choice or resolution of parameters for the Hough transform, a discussion of the range of values of the parameters leading to discussion of setting up of the accumulator array, the actual transformation and accumulation. After that, a new scheme for peak detection is presented involving automatic determination of threshold to apply to the accumulator array entries to determine peaks, application of the butterfly filter to reduce false peaks, and further reduction of false peaks by application of a neighbourhood criteria. Sample results are shown.*

Keywords: *Automatic threshold determination, butterfly filter, Hough transform, image processing, peak detection*

I. Introduction

This paper presents the application of the Hough transform to detection of straight lines in images captured by a mobile robot within a rectilinear environment for the purpose of autonomous navigation.

Prior to application of the Hough transform as presented in this paper, the images are pre-processed according to a scheme discussed in [1]. In summary, the pre-processing involves re-sizing, conversion to gray scale, edge-detection and thinning, and results in a binary image with black pixels outlining edges of the most important regions from the original image.

This paper deals with the specific implementation of the Hough transform taking the pre-processed binary image as input and resulting equations of the most significant straight lines that the black pixels of the binary image form.

Following detection of lines using the process discussed in this paper, further processes are carried out to complete the full capture-process-navigate cycle, which are not detailed in this paper. The next stage of the full cycle isdetermination of end-points of sub-lines found within the imagewhich is presented in [2]. Stages following that include, high-level features (corridors, doors, etc.) detection from sub-lines found, and the navigation scheme based on the high-level features found. Those are subjects of other papers to be published soon.

Section 2 Background which follows, gives a general overview of the Hough transform and key issues related to any application of it. Details of the specificapplication of the Hough transform in this work are presented in section 3 Implementation Specific Details for the Hough Transform.

II. Background

1.1 The Hough Transform

The Hough transform is an image processing technique invented to enable automatic recognition of lines [3]. It has since been applied to detection of other kinds of regular shapes and even non-regular shapes in images [4]. It has the effect of reducing the search for features such as a line in the original image to a search for a point in a new transformed image, where curves in the new image corresponding to points from the original image that the line would lie on, intersect [5]. The Hough transform is only used for detecting straight lines in this work, therefore versions for detecting other features are not discussed any further.

The Hough transform is "inherently stable and robust"[6]. It copes very well with noisy images[7]. It can find features even if the patterns representing them are broken in the original image [8].

1.1.1 Input Images for the Hough Transform

In this work, input images for the Hough transform are pre-processed robot-mounted-camera images of the surroundings of a mobile robot. Other works such as [5], and [9] to name a few also use (pre-processed) visual images as input for the Hough transform. However, several other types of input data have been used, for example, sonar data used in [10] and [11], and range-measuring laser used in [12]. [3]in which the transform was presented originally, had pictures of bubbles in a bubble chamber as input.
Visual images are not necessarily the best kind of input for the Hough transform. The Hough transform of range data is 'cleaner' according to [12].

This project uses visual images (that have been pre-processed)as input as the purpose of the project is to implement vision systems. Details of preprocessing scheme used have been discussed in [1]. Briefly, it involves conversion to gray-scale, re-sizing, edge-detection with the Sobel edge-detection filters, and thinning with a thinning method presented in [1] that is a slight modification to the thinning method of[13]. The result of the pre-processing scheme is a binary image with black pixels indicating edges of features from the original image. It is 128 pixels by 96 pixels in size with pixels indexed as shown in fig. 1 below

0	1	2	125	126	127
128	129	130	253	254	255
256	257	258	381	382	383
11904	11905	11906	12029	12030	12031
12032	12033	12034	12157	12158	12159
12160	12161	12162	12285	12286	12287

Figure 1 Image points indexing

1.1.2 Transformation from Image Space to Hough Space

Although [3]originally proposed transformation using the gradient-intercept form of the equation of a straight line, the normal (or polar) form of the equation of a straight line has become more popular. 2.1.2.1 Straight Line Hough Transform Using the Gradient-Intercept Form of the Equation of a Straight Line and 2.1.2.2 Straight Line Hough Transform Using the Normal Form of the Equation of a Straight Line which follow introduce the two approaches. 2.1.2.3 Resolution introduces the idea of resolution for the Hough transform.

1.1.2.1 Straight Line Hough Transform Using the Gradient-Intercept Form of the Equation of a Straight Line

Hough's original straight line detecting transform aims at finding out the likelihood that a point (x, y) lies on a line

$$y = mx + c \quad . \quad . \quad . \quad . \quad (1)$$

in the $x-y$ plane whose length is significant enough for the line to be taken as an important feature in the original image. In (1), m is the gradient of the line, c is the y -intercept of the line and x and y are standard coordinates of edge pixels.

The transform is based on the principle that every point (x, y) on the line given by (1), defines a set of m values and corresponding c values. This definition is achieved by rearranging (1) in the form

$$c = -xm + y \quad . \quad . \quad . \quad . \quad (2)$$

For a given point, x and y are constant, and so for varying values of m, (2) is a straight line in another plane – the $m-c$ plane. [14]refers to this as point-line duality. The line so derived is the transform of the point.

When using (2) as a basis for a transform scheme, a problem arises if the line in question is vertical or near vertical. The values of m and c both become infinity for vertical lines and are very large for lines that are nearly vertical. Computer modelling of parameter space to accommodate the plots of m against c becomes

impractical because of these large values. Various ways have been proposed for getting around this problem. One way, according to [4], is to use two sets of plots for parameter space. The standard approach already discussed, using (2), is used for when $|m|$ is less than or equal to 1.0. When $|m|$ is greater than 1.0

$$x = m'y + c' \quad . \quad . \quad . \quad (3)$$

is used where

$$m' = 1/m \quad . \quad . \quad . \quad (4)$$

and

$$c' = -c \quad . \quad . \quad . \quad (5)$$

This has the drawback of requiring much more memory, and a little additional work is required in putting together the findings of the two accumulator arrays.

1.1.2.2 Straight Line Hough Transform Using the Normal Form of the Equation of a Straight Line

Another way to get around the problem that has become popular is to use the polar form (also called the normal form) of the equation of a straight line

$$\rho = x\cos\theta + y\sin\theta \quad . \quad . \quad . \quad (6)$$

This approach was first proposed by [15]. It is based on the principle that for every point (x, y) in the x-y plane, there is a curve defined by (6) in the $\theta - \rho$ plane. Both ρ and θ are bounded within reasonable limits.

θ is bounded by 0° (inclusively) and 180° (exclusively), and ρ is bounded by $-\sqrt{(h^2 + w^2)}$ (inclusively) and $\sqrt{(h^2 + w^2)}$ (exclusively), where h, the highest magnitude of height on the image with the origin at the centre of the image, is given by $h = \lceil H/2 \rceil$, H being the height of the image, and w, the highest magnitude of width on the image with the origin at the centre of the image, is given by $w = \lceil W/2 \rceil$, W being the width of the image.This approach is employed in this work and is the basis for any further discussion on the Hough transform. Implementation of this approach for this work is discussed in more detail in 3 Implementation Specific Details for the Hough Transform. The gradient-intercept $(m - c)$ based transform is not discussed any further.

1.1.2.3 Image Space and Parameter Space

This input image for the Hough transform has points in the $x - y$ plane and is commonly referred to as image space. The new image derived from the transform has points in the $\theta - \rho$ plane and is sometimes called Hough space after Paul Hough the inventor of the technique. The qualities θ and ρ are commonly referred to as parameters of the transform, and Hough space is for that reason also commonly referred to as parameter space. In this work, the expression parameter space will be used because it appears to have become the most common name for the new plane.

1.1.2.4 Parameter Quantisation Intervals

(6)is used to transform every edge pixel (x, y) in image space. The intervals at which the resulting parameters θ and ρ are recorded in parameter space can significantly affect the quality of the results of the transform, and the processing time and space required.

Where x and y are examined at intervals of 1 pixel, as is usually done, an interval of 1 pixel for ρ, the distance parameter naturally makes sense. ρ is evaluated as a floating point number using (6), so the result is quantised to the nearest whole number.

The choice of interval for θ is not as straightforward. To determine the transform for each edge pixel, various values of θ are taken at a regular interval. Generally speaking, the smaller the interval used, the better the results obtained but also the higher the computing time and memory requirements. However, [16] has demonstrated that in detecting a line of length L, if the θ interval goes below $\tan^{-1}(1/L)$, a number of peaks result along the θ direction instead of a single peak, and this can lead to the transform yielding several

lines where there was only one. Depending on the extent of the spread, it might be possible to consolidate the multiple peaks back to a single peak, eliminating false ones. Consolidation of peaks in this work is discussed further in 3.6 Peak Detection.

The choice of interval for θ for the current project is discussed in 3.2 θ Resolution.

1.1.3 The Accumulator Array

Closely associated with parameter space is the accumulator array. It is a two dimensional array superimposed over parameter space. The actual number of elements in the array depends on the level of accuracy required from the application in question. For the highest integral resolution, there will be an element of the accumulator array for every point (θ, ρ) in parameter space, and the value of the element will simply be the value of the accumulation at the corresponding parameter space point (θ, ρ). How the value of accumulation is derived is discussed shortly in 2.1.5 Accumulation. For this resolution, the accumulator array will have as many columns as the range of θ, and as many rows as the range of ρ. This resolution is used in this work.

For other resolutions, the parameter space is divided into cells that are usually of equal sizes. It is common to refer to these cells as bins. The accumulator array is set up to have elements corresponding to each of these cells. Entries to the array are aggregate sums of the values of accumulation for parameter space points within the cell corresponding to the particular accumulator array entry.

The resolution of the accumulator array is chosen bearing in mind processing time and space requirements on the one hand and the level of accuracy required for feature detection on the other. The amount of processing time and memory required both vary linearly with the resolution chosen [8].

1.1.4 Accumulation

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.

[4]has suggested that instead of increasing just the appropriate accumulator array point, a square block of points with the point (θ, ρ) as centre is incremented. If a block 3 x 3 points in size is used for example then 9 bins are incremented for every $\theta - \rho$ pair. This results in thick parameter space lines and can help detection of intersection points during peak detection.

Other ways have been suggested by [4] for further enhancement of peak detection including:

a. Addition of the gradient magnitude of the edge for accumulation rather than simply incrementing by 1. This plots a measure of the likelihood of the source point being an edge.
b. Incrementing a 3 x 3 block of bins but incrementing the middle entry with a larger figure than the others.

1.1.5 Peak detection

When all edge pixels have been processed, accumulator array entries 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. Some accumulator array entries will have higher values than their neighbours and are referred to as peaks. Peak detection is the process of determining which bins are peaks.

Peak detection is achieved by application of a threshold. All accumulator array entries above a certain threshold automatically qualify as peaks and all others are not peaks.

Depending on the application, thresholds can be fixed or they can vary for every image.

1.1.5.1 Fixed Threshold

In applications where it is known how many pixels a line (or whatever feature is being detected) needs to have to be significant, it is possible to determine and use a constant threshold.

Use of a fixed threshold is advantageous in being very easy to implement, and requiring no significant additional computing time as would be required by constantly evaluating the threshold. A fixed threshold can be effective where target features do not vary widely.

It has the drawback of not being sensitive to the nature of specific images. A busy image would tend to have much higher accumulations and using a fixed threshold would mean that such an image returns many more lines than are sensible. These can make further processing more complicated than it needs to be. A sparse input

image would return fewer lines than are required to sensibly interpret the image even though there is enough information in the image to find those lines with a lower threshold.

1.1.5.2 Variable Threshold

Where it is not practical to use a fixed threshold, different thresholds would need to be determined for every image processed. A few approaches can be taken. One proposed by this work is to determine threshold based on an approximate number of expected outcomes. In summary, this means that a decision is taken on how many significant elements are required, n say, and then aiming at selecting the top n elements of the array. This is discussed in detail in 3.6 Peak Detection.

It is also possible to determine the appropriate threshold by introducing a number of different ones on a trial and error basis. The outcome of the transform with each threshold can be studied and then the threshold with the best outcome can be maintained. This approach requires a lot of manual study of results, in addition to understanding of the quality of the outcomes. It is therefore not suitable for real time applications.

1.1.5.3 False Peaks

In most situations, peaks are not easily distinguishable from non-peaks. This can be due to a number of possible situations. An example is a situation where several accumulator array entries in a neighbourhood have values higher than the edge detection threshold. Although one of them is higher than all others, several peaks which are not actual peaks are returned by the application of the threshold. This work refers to them as false peaks.

Where they exist, elimination or at least minimisation of false peaks is necessary. One method proposed by [6] is the use of the butterfly filter. They worked out a 5 x 9 convolution filter which when applied to an accumulator array will emphasis actual peaks while suppressing false peaks. They determined the filter based on the fact that accumulator array entries due to points on a straight line have a characteristic shape that resembles the shape of a butterfly, hence the name. The filter was determined such that a maximum positive response is obtained when it coincides with the butterfly shape due to a line, and a zero response is obtained in a uniform area of the array.

Because application of the 5 x 9 butterfly filter is computing time demanding, [6] also developed what they call a reduced butterfly filter, which is 3 x 3 in size. It has a similar effect to the full filter although it does not take as many points into accounts, sosignificantly reduces the amount of arithmetic required to evaluate convolutions with the filter. This reduced butterfly filter is employed by this work. The butterfly filter is discussed further in[6] and its implementation in this work is discussed in further detail in 3.6.1 Application of the Butterfly Filter.

1.1.6 Reconstruction

When peaks have been correctly determined, the parameters which represent are representations of the lines found. This sub-section discusses the reconstruction of the gradient-intercept form of the equations of the lines found. This is not absolutely necessary but may be helpful if a graphic illustration of the line is required, or as a step towards determining endpoints and length using some techniques. Determination of endpoints and lengths for the lines found is beyond the scope of this paper but will be presented in a separate paper soon.

In this work, as the transformation was done with (6), reversal of the transform to obtain the lines found can be achieved by re-arranging (6) in the form of (1) to give

$$y = \rho \cos ec\theta - x\cot\theta \quad . \quad . \quad . \quad (7)$$

This implies

$$m = (-\cot\theta), \quad . \quad . \quad . \quad (8)$$

and

$$c = \cos ec\theta \quad . \quad . \quad . \quad (9)$$

III. Implementation Specific Details for the Hough transform

In this section, information specific to the implementation of the Hough transform for this work is discussed.

As discussed in section 2, a key element of the Hough transform is setting up the accumulator array. The accumulator array is a table of θ values, and corresponding ρ values. To set it up, first a decision is made about the angular resolution to be used. This is discussed in 3.2 θ Resolution. Secondly, to be able to model the array in a computer programme, the range of corresponding ρ values is determined. This is discussed in 3.3 Range of the Hough Transform Function.

With the accumulator array in place, the Hough transform can be applied to the input image described in 2.1.1. This assigns different accumulations to elements of the accumulator array, and is discussed in 3.5 Transformation and Accumulation. To achieve the goal of finding important lines from the original image, peaks have to be detected in the accumulator array, and reverse transformed. Peaks are determined by application of a threshold which is itself worked out for every image to get best results. Peak detection is discussed in 3.6 Peak Detection.

1.2 Coordinates in Image Space

In applying the Hough transform, it is usual for coordinates for points in image space to be chosen so that the origin (0, 0) is at the centre of the image. That approach is adopted in this work. Given from the discussion in 2.1.1, that the width of the image is 128 pixels and the height is 96 pixels, x ranges from -64 to 63 increasing from left to right, and y ranges from -48 to 47 increasing from bottom to top. From the indexing scheme presented in 2.1.1, if follows that for a point with index i,

$$x = iMod128 - 64 \quad . \quad . \quad . \quad (10)$$

and

$$y = 47 - i/128 \quad . \quad . \quad . \quad (11)$$

where the operators Mod and $/$ refer to the integer remainder and integer division operations respectively. The first (top left) point in the image with i=0, has coordinates (-64, 47), for example, and the origin (0, 0) is the point with index i = 47*128 + 64 i.e. i=6080.

1.3 θ Resolution

An essential step in the application of the Hough transform, as pointed out in 2.1.2 Transformation from Image Space to Hough Space, is deciding what step value to use for θ. The aim is to choose a resolution that will make it possible to find any reasonably significant line possible in image space. This means that a point A at the centre of the image which is taken to be the origin, should be seen to be collinear with another point B at the far end of the image, and also be seen to be collinear with another point C right next to point B along 2 separate lines. Fig. 2 is an extract from an illustration image with the regular size magnified 8 times to show three such points. The green square A is a pixel at the centre of the image, the red point B is a pixel at the farthest possible distance from A and the blue one C is the pixel next to the red one.

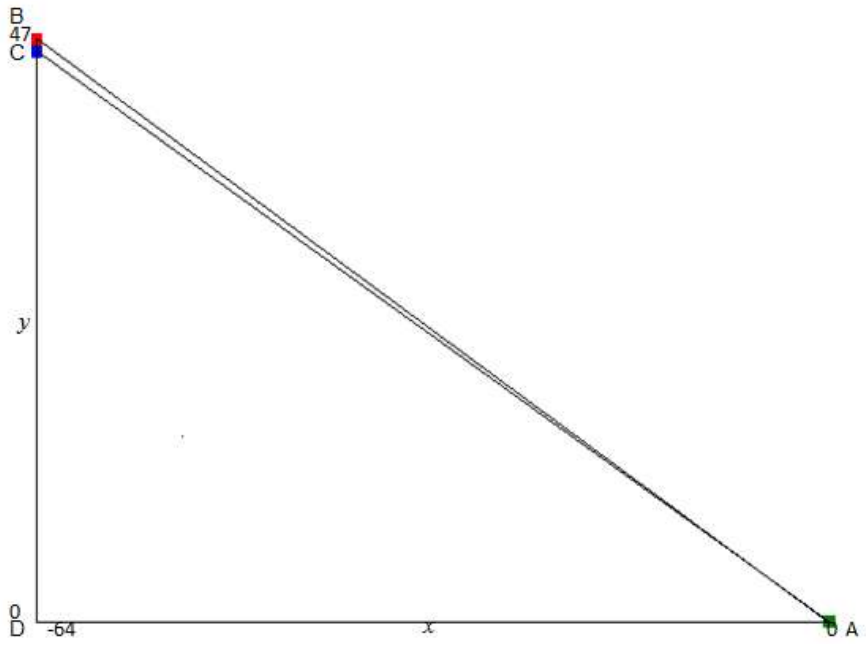

Figure 2Angle due to 1 pixel difference at farthest distance from origin

The minimum change in angle $\Delta\theta$ corresponding to the angle BAC, needs to be so small that the line AB or any sub-line on it can be picked up, as well as the line AC or any of its sub-lines. From the dimensions of the image, AD is 64 and BD is 48. Therefore the angle BAD will be $\tan^{-1}(47/64)$, i.e.,

36.2926°. Similarly, angle CAD will be $\tan^{-1}(46/64)$, i.e., 35.7067°. $\Delta\theta$ which is the difference of the two angles is 0.5859°. This is the value of $\Delta\theta$ that can pick up both lines AB and AC as two separate lines. However, for the purpose of setting up indices for an accumulator array, an integral value is ideal, so 1° is the required interval, being 0.5859°rounded up to the nearest larger whole number.

Fig. 3 shows lines plotted from the origin to the edges of the image, at intervals of 1°. The full image is covered – very densely towards the middle where the multiple-peak effect described by [16] can be expected, and a little sparsely towards the edge. This can be expected because the actual interval of 1° is higher than the ideal interval of 0.5859°. It can be observed though that there is no uncovered part of the image wider than 1 pixel. Points that should have fallen in those spaces would have been quantised to adjacent pixels. For the purpose of comparison with fig. 3, fig.4 was developed in a similar way, except using with a 2° interval. Towards the diagonals, spaces of up to 3 pixels in width exist, and lines of length up to 8 pixels can be missed if that interval is used.

Figure 3 Lines drawn across the range of θ at 1° interval

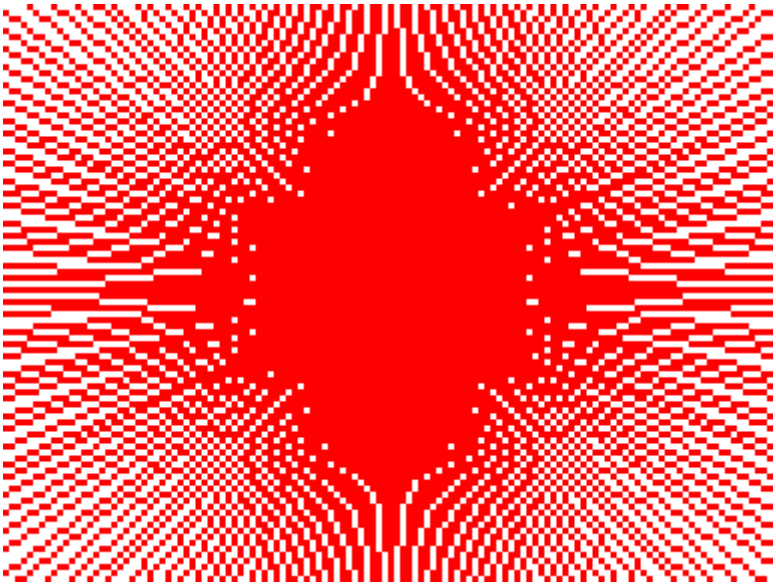

Figure 4 Lines drawn across the range of θ at 2° interval

1.4 Range of the Hough Transform Function

As pointed out in 2.1.2.2 Straight Line Hough Transform Using the Normal Form of the Equation of a Straight Line, values of the parameters ρ and θ are bounded unlike m and c when (1) is used as the basis

for transformation. These bounds dictate the size of the parameter space and the accumulator array. It was further pointed out that θ lies in the interval $[0°,180°)$ and ρ lies in the interval $[-\sqrt{(h^2+w^2)},\sqrt{(h^2+w^2)})$ where h and w are as defined in 2.1.2.2 Straight Line Hough Transform Using the Normal Form of the Equation of a Straight Line. For the 128 x 96 pixel image being used, h is 48 and w is 64. ρ is therefore bounded by $\left[-\sqrt{(64^2+48^2)},\sqrt{(64^2+48^2)}\right)$ or $[-80,80)$.

1.5 Accumulator Array

With the bounds of θ and ρ known, the accumulator array is set up. θ ranges from 0 to 180° as discussed in 3.3 above. However, as discussed in 3.6 Peak detection, it is helpful to take a range for θ up to 185°. This makes peak detection techniques such as application of the butterfly filter easier to apply without loss of important information.

As discussed in 3.2 θ Resolution, θ is increased at an interval of 1°, and ρ values are quantised to a single pixel. Also from the discussion in 3.3 Range of the Hough Transform Function, the range of values for ρ is $[-80,80)$. A resolution of 1 is selected for maximum accuracy. With these considerations in mind, the accumulator array can be set up as shown in fig. 5 indicating the ranges of θ and ρ.

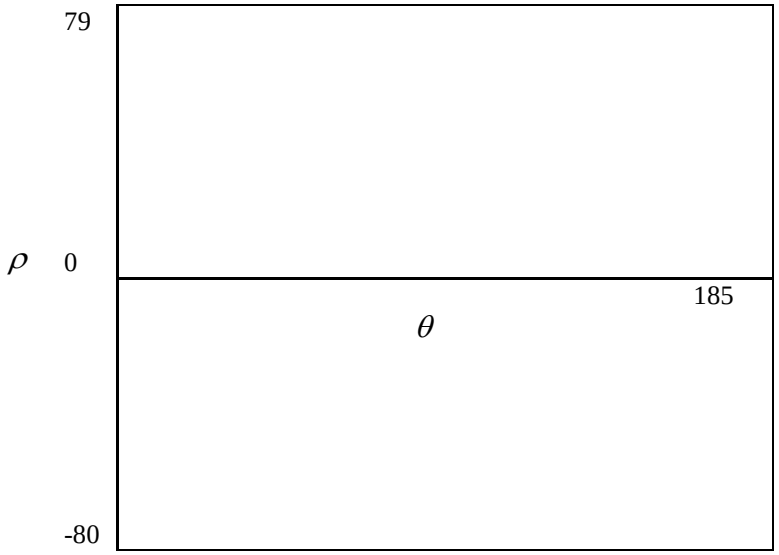

Figure5 Ranges of θ and ρ in Accumulator Array

1.6 Transformation and Accumulation

With the accumulator array in place, the transform proper can commence. The following sequence of steps describes this implementation.

1. Initialise accumulator array entries to 0
2. For all points of image space, do 3
3. If point is black do 4 to 5 else do 6
4. For θ from 0 to r, the upper boundary of θ, do 5 and 6
5. Determine ρ using $\rho = x\cos\theta + y\sin\theta$
6. Increase the accumulator array entry for current θ and ρ

1.7 Peak Detection

As pointed out by [17], the Hough transform can yield false peaks, sometimes in significant proportions.

Peak detection in this work is achieved by a scheme consisting of a number of stages:

1. Determination and application of the most appropriate peak detection threshold for the image under consideration
2. Application of the butterfly filter to entries above the threshold from step 1 above only
3. Determining which elements of the accumulator array selected from 2 above are local maxima within a 5 x 5 neighbourhood

This scheme is different from any other schemes encountered in the literature, and combines the real-time variable threshold determination process described shortly which has also not been encountered in any previous literature, with straightforward threshold application traditionally used with the Hough transform, and also with the butterfly filter approach of[6].

The scheme is used to minimise the emergence and effects of false and multiple peaks which[17] and [16] have pointed out result from the Hough transform.

These stages of the scheme are discussed in detail in 3.6.1 Threshold Determination, 3.6.2 Application of the Butterfly Filter and 3.6.3 Determination of local maxima.

1.7.1 Threshold Determination

The first stage in peak detection is determination of a threshold. As discussed in 2.1.6 Peak Detection, this work uses a simple algorithm to work out the most appropriate threshold to employ given a target number of peaks. The target number of peaks was set at 250 after studying the results for various typical images for different values of T . Details of that study is the subject of another paper to be published soon.

Let m be the lowest entry in accumulator array and let M be the highest entry.

Let C_i be defined as the number of accumulator array entries with value i after application of the Hough transform algorithm, $m \le i \le M$ .The approach involves doing a cumulative sum, S_i say, where

$$S_i = \sum_{j=M}^{i} C_j \qquad . \qquad . \qquad . \qquad (12)$$

i.e., S_i is the sum of all accumulator array entries from the highest entry to the i^{th} entry. S_i is then checked against T , the target number of peaks.

While $S_i < T$, the next value of S_i is determined.

As soon as $S_i \ge T$, i is taken as the threshold. This ensures that at least the required number of peaks is returned.

1.7.2 Application of the Butterfly Filter

The butterfly filter first proposed by (Boyce et al, 1987) is commonly used to enhance actual peaks and suppress false peaks within small neighbourhoods of them. The reduced filter was also proposed by (Boyce et al 1987) to be used for applications requiring low processing times. The reduced butterfly filter is used in this work as it yields fairly good results without requiring as much algebra processing as the full filter. In this work, as part of the scheme presented in 3.6 Peak Detection, the butterfly filter is only applied to elements marked as peaks from the threshold application stage described in 3.6.1 Determination of Target Number of Peaks above. The reduced butterfly filter is shown in fig. 6.

$$\begin{bmatrix} -1 & -4 & -1 \\ 2 & 8 & 2 \\ -1 & -4 & -1 \end{bmatrix}$$

Figure 6The reduced butterfly filter

The reduced butterfly filter is 3 x 3 in size and at the point of application, the element of the array under consideration must have rows above it, below it, to the left of it and to the right of it. So if the accumulator array is designed within the theoretical range of [0°,180°) for θ and $\left[-\sqrt{(h^2+w^2)}, \sqrt{(h^2+w^2)}\right)$ for ρ as discussed in 3.3 Range of the Hough Transform Function, a butterfly filtered array is obtained which only has meaningful values in the range [1°,179°) for θ and $\left[(-\sqrt{(h^2+w^2)}+1), (\sqrt{(h^2+w^2)}-1)\right)$ for ρ . The butterfly filter cannot yield meaningful results for the first and last rows, and the first and last columns of the accumulator array. This is particular undesirable because entries for $\theta = 0^0$ and $\theta = 179^0$, i.e. most representations of vertical lines are lost.

Furthermore, step 4 described in 3.6.3 Determination of Local Maxima, requires finding local maxima within 5 x 5 neighbourhoods. This means that maxima within the second and second-to-the-last rows and the third and third-to-the-last columns of the accumulator array are also lost. This further reduces the range from

which peaks can be detected to [3°, 177°) for θ and $\left[(-\sqrt{(h^2+w^2)}+3),(\sqrt{(h^2+w^2)}-3)\right)$ for ρ. There is no straightforward way of recovering the information lost due to shrinkage of the ρ axis. To get around this for the θ axis, it is necessary to either find a way to wrap from the 180° end of the accumulator array back to the 0° when applying the butterfly filter and choosing maxima, or simply extend the array a little as the same lines begin to repeat at 180° and beyond. The same lines show up at 0° and 180°, and at 2° and 182° for example. What changes is the sign of ρ. This work extends the array to 185° so that 180° can make up for 0° lost to application of the butterfly filter, and 181° and 182° can make up for 1° and 2° lost to the criteria that a genuine peak must be a maxima in a 5 x 5 neighbourhood it is at the centre of. 183° and 184° are needed to complete 5 x 5 neighbourhoods for entries with θ values of 182°, and 185° is needed to provide a 3 x 3 region for application of the butterfly filter for entries with θ values of 184°. This means there are 186 columns in the accumulator array.

The algorithm applied follows. Note that 186° is used for the width of the array as it is the range of values for θ used as explained in the paragraph above, and 160, the height of the array is the range of the Hough transform function as discussed in 3.3 Range of the Hough Transform Function. Note also, that steps 6 and 7 merely set the values for the first and last rows and the first and last columns to 0 because the filter cannot be applied to them as explained earlier in this section.

1. Let A be the accumulator array with $w=186$ as its width, and $h=160$ as its height, and B the butterfly filtered accumulator array with the same dimensions.
2. For all accumulator array rows j, j running from 1 to $h-2$ do 3
3. For each column i of current row j, i running from 1 to $w-2$ do 4
4. If $A_{i,j}$ has been marked as a peak do 5
5. (i) $B_1 = -1\times A_{i-1,j-1} - 4\times A_{i,j-1} - 1\times A_{i+1,j-1}$
   (ii) $B_2 = 2\times A_{i-1,j} + 8\times A_{i,j} + 2\times A_{i+1,j}$
   (iii) $B_3 = -1\times A_{i-1,j+1} - 4\times A_{i,j+1} - 1\times A_{i+1,j+1}$
   (iv) $B_{i,j} = B_1 + B_2 + B_3$
6. Set top and bottom rows to 0, i.e,
   For all i, i running from 0 to $w-1$
   (i) $B_{i,0} = 0$
   (ii) $B_{i,h-1} = 0$.
7. Set side columns to 0
   For all j from 0 to $h-1$
   (i) $B_{0,j} = 0$
   (ii) $B_{w-1,j} = 0$

1.7.3 Determination of Local Maxima

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.

IV. Sample Results

Following in Figs. 7 and 8 are sample results for two typical images, image1 and image2. For each of them, the figures show the original image with navigationally important lines highlighted, the pre-processed version of the image which is the actual input to the process described in this paper, lines founds using the processes described in this paper, and sub-lines found using those lines found with a process describedin [2].

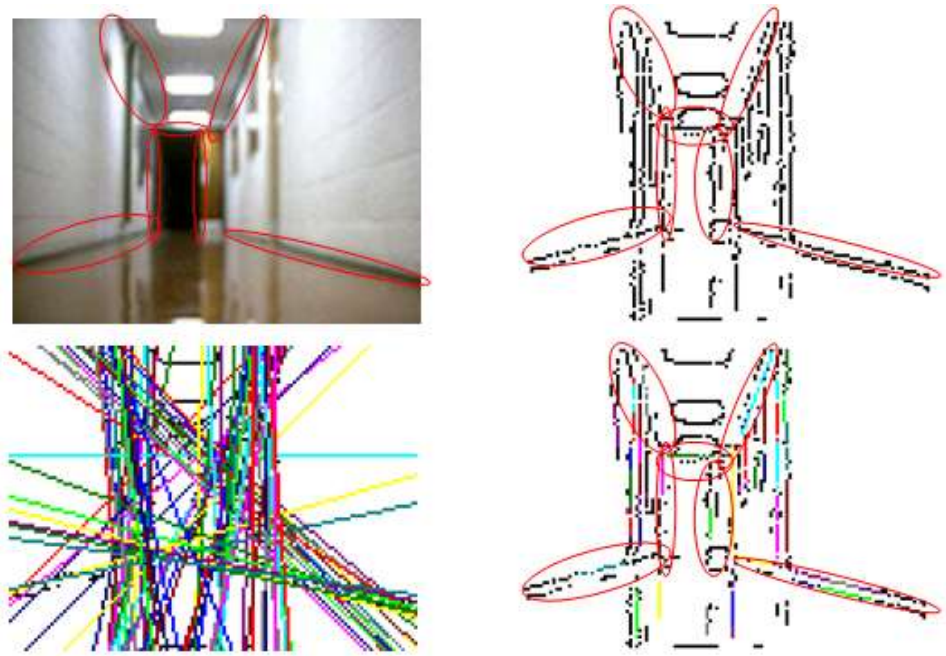

Figure 7Sample image1 and results

(a) A typical image1 (b) Pre-processed version of image1 (c) Lines found using the Hough transform (d) Sub-lines found using lines found (using process described in [2])

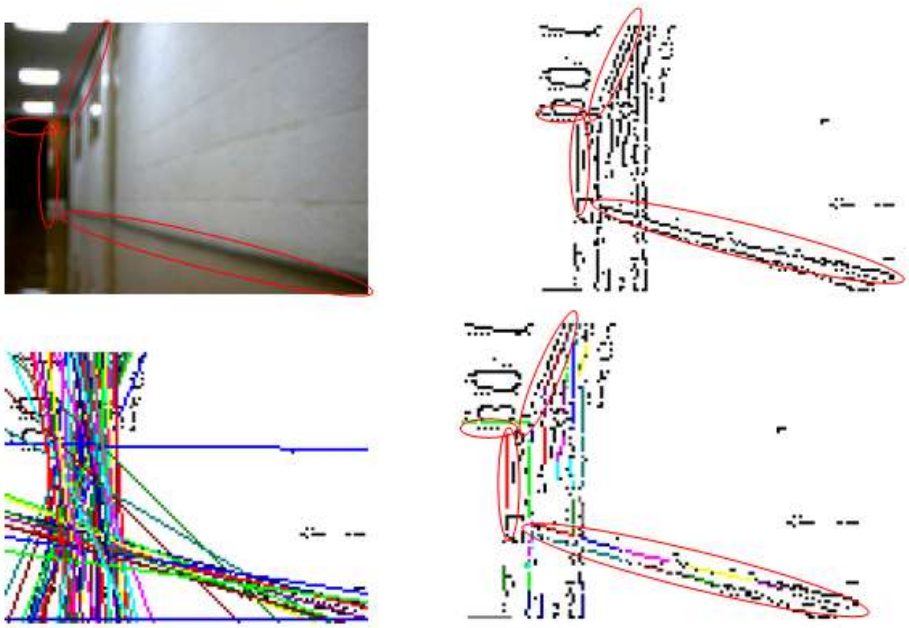

Figure 8Sample image2 and results

(a) A typical image2 (b) Pre-processed version of image1 (c) Lines found using the Hough transform (d) Sub-lines found using lines found (using process described in [2])

Table 1 following summarizes the number of lines found in the images shown in image1 and image2.

Table 1Summary of number of lines found at various stages of line detection for 2 typical images

Target Number of Peaks	**Image1**	**Image2**
Number of Peaks found	289	262
Number of Peaks after Butterfly filtering	49	37
Number of sub-lines found	49	29

V. Conclusion

This paper describes an application of the Hough transform for line detection, as part of a bigger process of mobile robot self-navigation based on visual data. The bigger process starts with capture and pre-processing of an image by a camera mounted on a robot. Line detection as described in this paper is then done, and lines obtained are further processed to obtain navigationally important features that eventually inform the robots autonomous movement in processes not described in this paper, but presented in [1], [2] and in papers to be published in the future. The bigger process has successfully been achieved in part because line detection as described in this paper was successfully achieved.

Line detection with the Hough transform as described in this work was done targeting a specific application. Other applications requiring identification of lines, or other geometric primitives, may be able to adopt it, or parts of it, depending on the nature of those applications,their specific platforms and the processing resources available on them, and the sensitivity of the applications to time. For example, the choice of resolution of parameters for the Hough transform was done with the size of the input images for this particular application in mind, and the nature of the lines targeted. As another example, the reduced butterfly filter was chosen for this work, rather than the full filter because timely completion of peak detection was important for this application, and very high level of accuracy processing one image is not critical in the grand navigation scheme. These may not be true for other applications.

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, 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.
[2]. G. K. Damaryam, A method to determine end-points of straight lines detected using the Hough transform, In Press, *International Journal of Engineering Research and Applications, 6(1)*, 2016
[3]. P. Hough, Method and Means for Recognising Complex Patterns, United State of America Patent 3069654, 1962.
[4]. A. Low,Introductory Computer Vision and Image Processing,(London, United Kingdom: McGraw-Hill Book Company, 1991).
[5]. A. O. Djekouneand K. Achour, Visual Guidance Control based on the Hough Transform,IEEE Intelligent Vehicles Symposium, 2000.
[6]. J. F. Boyce, G. A. Jonesand V. F. Leavers, An implementation of the Hough transform for line and circle location. Proc, SPIE Inverse Problems in Optics,The Hague, Netherlands, 1987.
[7]. G. L. Dempsey. and E. S. McVey, A Hough Transform System based on Neural Networks,IEEE Transaction on Industrial Electronics, 39(6), 1992, 522-528.
[8]. M. Atiquzzaman, Multiresolution Hough Transform - An Efficient Method of Detecting Patterns in Images,IEEE Transactions on Pattern Analysis and Machine Intelligence, 14(11), 1992, 1090-1095.
[9]. C. Espinos and M. A. Perkowski,Hierarchical Hough Transform for Vision of the Psubot, Proc., Northcon 90 Conference Record, Portland, OR, 1991, 291-296.
[10]. X. Yun, K Latt G. J. Scott, Mobile Robot Localization using the Hough Transform and neural networks. Proc.,IEEE International Symposium on Intelligent Control, Gaithersburg, MD, 1998, 393 – 400.
[11]. L. Banjanovic-Mehmedovic,I. Petrovic and E. Ivanjko, Hough transform based correction of mobile robot orientation. Proc., IEEE Conference on Industrial Technology,2004, 1573-1578.
[12]. J. Forsberg and U. Larson and A. Wernerson. Mobile robot navigation using the range-weighted Hough transform. IEEE Robotics and Automation Magazine, 2(1), 1995, 18-26.
[13]. J. Park and H. Chen and S. T. Huang, A new gray level edge thinning method. Proc., ISCA 13th International Conference, Computer Applications in Industry and Engineering, Honolulu, HI, USA, 2000.
[14]. E. R. Davies, Machine Vision: Theory, Algorithms and Practicalities, 2nd Ed(San Diego: Academic Press, 1997).
[15]. Duda and Hart. 1973. Pattern Classification and Scene Analysis(New York: Joh Wiley and Sons, 1973).
[16]. V. F. Leavers,Shape Detection in Computer Vision Using the Hough Transform(London: Springer-Verlag, 1992).
[17]. W. Grimson, W. L. Eric and D. P. Huttenlocher. On the Sensitivity of the Hough Transform for Object Recognition. IEEE Transactions on Pattern Analysis and Machine Vision, 12(3), 1990, 255-274.