IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 1, Ver. I (Jan – Feb. 2016), PP 09-15
www.iosrjournals.org

A Preprocessing Scheme for Line Detection with the Hough Transform for Mobile Robot Self-Navigation

Gideon Kanji Damaryam[1], Haruna Abdu[2]
[1](Department of Computer Science, Federal University, Lokoja, Nigeria)
[2](Department of Computer Science, Federal University, Lokoja, Nigeria)

Abstract: *This paper presents the pre-processing scheme used for a vision system for a self-navigating mobile robot which relies on straight line detection using the Straight Line Hough transform. The straight line Hough transform is an Image Processing technique for detection of straight lines in an image by transforming points in the image to another image in a way that accumulates evidence that the points from the original image are constituents of a straight line type feature from the original image. The pre-processing presented includes image re-sizing, conversion to gray scale, edge detection using the Sobel edge-detection filters, and edge thinning with a newly developed method that is a slight modification of an existing method. The newly developed method has been found to yield thinned images more suitable for later stages of this work than other thinning methods. Output from the pre-processing scheme presented is used as input for the remainder of the vision-based self-navigation system.*

Keywords: *Edge-detection, Hough transform, Image Processing, Machine vision, Pre-processing*

I. Introduction

This paper describes an image pre-processing scheme, which transforms an image captured by a camera mounted on a mobile robot, into a representative binary image optimized for straight-line detection using the Hough transform. Straight lines are detected as part of a vision system for a mobile robot, which works by detecting and interpreting lines and end-point of lines to find navigationally important features.Detection of lines is detailed in [1] and determination of end-points of lines is detailed in [2]. The vision system is part of a self-navigation system intended for use by a small mobile robot within a rectilinear indoor environment such as a University faculty building. The full system is described in [3].

Hough transforms are used for detection of features such as lines, curves and simple shapes within images. They work by transforming potential parts of a target feature in a given image topoints in a new image while accumulating measures of the likelihood that points in the new image aredue to features of the required type from the original image. When the transformation is complete, points in the new image can then be subjected to a predefined threshold so that points that are very likely to be due to the required kind of feature can be selected and the original features identified by reversing the transformation process. The Hough transform used depends on the feature to be detected. As the work that is the basis of this paper is concerned with the detection of straight lines, it prepares images for the transform called the straight line Hough transform, which transforms points, being potential components of lines, in the original image to curves in a new image. The number of curves that intersect at a particular point in the new image is a measure of the likelihood that the points from the original image whose transform curves intersect at the intersecting points in the new image, were points forming a straight line in the original image. The line is defined by values of the transform parameters which can be read off the axes of the new image. This way, lines in the original image can be detected. For brevity, in this work, the straight line Hough transform is simply referred to as the Hough transform, as is commonly done. Further information about the Hough transform is available in several publications including [4] and[3].

In the context of the Hough transform, the processing required to transform an image captured by a camera (a raw image), to an image optimized for application of the Hough transform is referred to as pre-processing. The pre-processing tasks presented in this paper include resizing of the captured image, edge-detection and edge-thinning.Image capture is first discussed also, although it is not part of pre-processing in a very strict sense.

II. Capture, Resizing and Conversion to Gray Scale

2.1 Capture-Process-Navigate Cycle

To achieve vision-based navigation, it is necessary to capture, and process an image, and then effect navigation on the basis of the result of the processing. This cycle is repeated until a predefined navigation programme is completely executed, or the entire navigation process is otherwise terminated.

2.2 Capture

In this work, images were captured using a single forward-facing camera. It was ensured that there was sufficient light to clearly identify separate features in the images such as walls, floors and doors. The base of the camera was set up parallel to the floor.

2.3 Resizing

A standard image size was chosen to give a good compromise between usefulness of output and processing time. The reduced image size chosen was 128 x 96 pixels. When this size of image is fully processed, fairly fine details such as the two edges of a door on the side of a corridor can be extracted, yet the time for processing the image is not prohibitive.

Other image sizes were tried. These included 32x32 pixels and 64x48 pixels. In both cases, the level of detail available when the image is fully processed is limited and means that higher level post-processes to interpret the results do not have adequate input. A feature such as a door that is noticeable to a human observer in an image can be reduced to a single line if the image is reduced to a 32 x 32 size, and so the door cannot be picked up as a door by the post-processing for detecting doors, for example. Fig. 1 shows the various types of results for a typical image. Fig. 1a is the original image magnified by 2.67, figure 1b is the 32 x 32 thinned version magnified by 8, figure 1c is the 64 x 64 version magnified by 4 and figure 1d is the 128 x 96 version again magnified by 2.67. The door circled in figure 1a has no chance of being picked up as a door in the 32 x 32 thinned image because it almost doesn't appear, and in the 64 x 64 thinned image because it appears as a single line.

Also, although a square aspect ratio was considered, a 4:3 aspect ratio was selected as the cameras used all captured in 4:3 ratio and changing the ratio led to unnecessary loss of information from the sides as shownin Fig. 1 below.

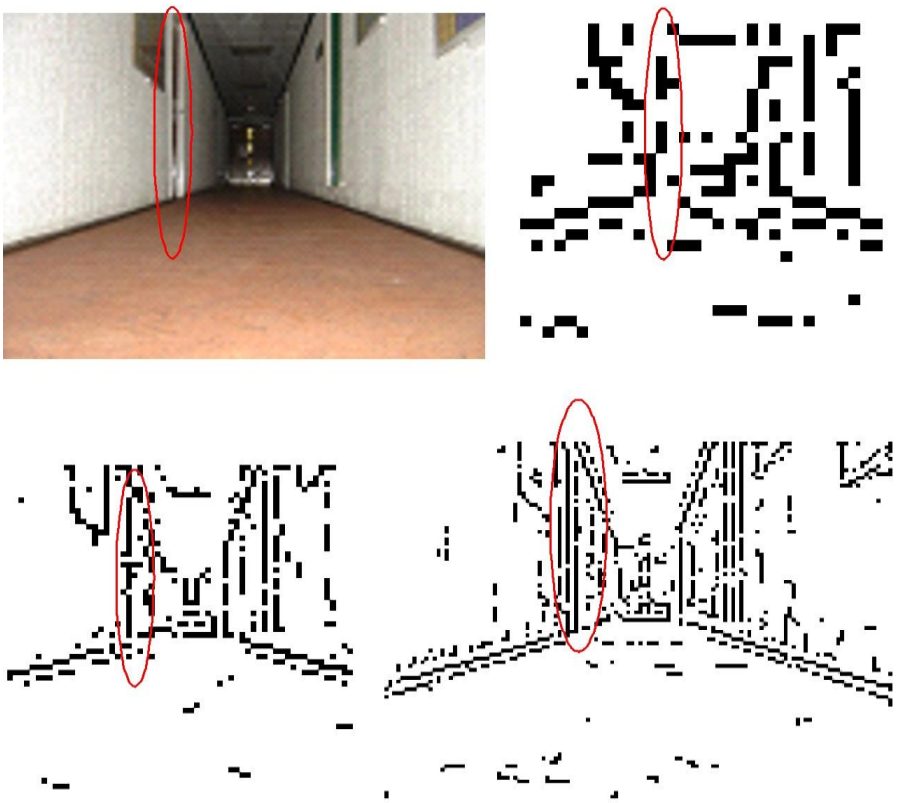

Figure 1 Effects of various image sizes

(Top left) Original image magnified by 2.67(Top right) 32 x 32 thinned image magnified 8 times(Bottom left)64 x 64 thinned image magnified 4 times(Bottom right)128 x 96 thinned image magnified by 2.67

Depending on the target features, the algorithms used for detecting them, and the importance of timeliness for specific applications, other image resolutions can be used. [5] Used a 30 x 32 sized grey-scale

image as input to a neural network for the purpose of navigating a robot to avoid moving obstacles and turning into junctions. [6]Reduced512 x 480 sized images to 64 x 60 sized images in their *Corridor Follower* module and then usedthoseas input for the Hough transform. They then used the resulting Hough space as input to a neural network.They report that the reduction in size has no noticeable effects on the performance of the module.

[7]resizedcaptured images of size512 x 512 to a 256 x 256 size – a higher resolution than the images in the current work - and usedthem to locate a docking station using an algorithm that requires up to 5 runs of the Hough transform. They report very high processing times (up to 10 minutes) however.

2.4 Intensity Determination

The camera used for this work captures coloured images. These are stored as image objects that have information about the levels of primary colors (red, blue and green) at every point of the image. For edge-detection to commence, it is necessary to determine the intensity at each point. This is done by extracting the level of each of the three colours and determining the average at each point.

2.5 Image Point Indexing

Points in images are labeled with identification codes as illustrated in Fig. 2. The point at the top-left position is labeled 0. Subsequent points going right are labeled with consecutive numbers until the end of the row. The labeling is continued on the next row from the left.

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 2: Image points indexing

III. Edge Detection

Edge-detection is the first pre-processing step implemented after an image of the right size has been obtained. It yields an edge-image by plotting lines connecting points where there are significant changes in pixel intensity, and which can therefore be taken as indications of edges of features in the image[8]. An edge image, ideally, contains lines that outline features in the original image.

With the intensities in the grey-scale image determined as discussed in *2.4 Intensity Determination*, a filter is applied across the image, which works out for each point in the image, the possibility that the point is an edge. A threshold, selection of which is a task in itself, is then applied to select points with high possibilities of being edge points.

The Sobel edge-detection filters were chosen for this work. Other edge-detection filters and techniques exist. One example is using the Laplacian edge-detection filters which havealsobeen reported to be accurate for detecting edges which are very gradual [8]. The Sobel filters were chosen for this work because not only do they provide a measure of magnitude for gradients of edges which were found to be good enough for images of the type used in this work, they also provide angles for the gradients that are used in some thinning algorithms, including the one used in this work. Thinning in this work is discussed shortly in *IV. Edge Thinning*. A fuller discussion on the Sobel filters is available in [8], as well as several other resources.

The Sobel filters are two 3 x 3 matrices, M_{ver} and M_{hor} . These are applied across images. $M_{ver}$is designed to find vertical edges and M_{hor} is designed to find horizontal edges.

M_{ver} is defined as:

$$M_{ver} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad . \quad . \quad . \quad . \quad . \quad (1)$$

and M_{hor} defined as:

$$M_{hor} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad . \quad . \quad . \quad . \quad . \quad (2)$$

The filters yield a measure of the possibility that there is a vertical and a horizontal edge, respectively, at a given point. These measures are called gradient magnitudes. The two gradient magnitudes, gm_{ver} and gm_{hor}, are obtained by convolution of the respective filters with the image I:

$$gm_{ver} = \left| M_{ver} * I \right| \quad . \quad . \quad . \quad . \quad . \quad (3)$$

$$gm_{hor} = \left| M_{hor} * I \right| \quad . \quad . \quad . \quad . \quad . \quad (4)$$

The two are then summed to give an overall gradient magnitude, gm for the point:

$$gm = gm_{ver} + gm_{hor} \quad . \quad . \quad . \quad . \quad . \quad (5)$$

The Sobel filters also provide an estimate of the angle, α of the gradient. This is simply the arc tangent of the horizontal gradient magnitude divided by the vertical gradient magnitude:

$$\alpha = \tan^{-1}\left(\frac{gm_{hor}}{gm_{ver}} \right) \quad . \quad . \quad . \quad . \quad . \quad (6)$$

3.1 Edge Threshold determination

Once gradient magnitudes have been determined, the next stage in edge-detection is deciding from the gradient magnitudes, which points are edge points and which ones are not. This involves application of a threshold. This work has developed a scheme where, rather than assign a fixed threshold for determining edges, a target is provided of the number of edge points required. The following algorithmis then used to work out what threshold will result in getting a number of edges equal to, or a little more than that specified:

1. Determine maximum gradient magnitude, M, from the array of gradient magnitudes GM
2. Determine minimum gradient magnitude, m, from the array of gradient magnitudes GM
3. Determine range of gradient magnitudes, R, using $R = M - m + 1$
4. Determine target number of non-edge points, N', as the difference between total number of points, N, and target number of peaks, T', i.e. $N' = N - T'$
5. Determine the number of elements of GM having value a for each a where $m \leq a \leq M$ and store each as G_a
6. Initialize a counting variable i to M, and set S_i, the M^{th} cumulative sum, to G_m
7. Reduce i by 1
8. Add previous cumulative group count to current group count to get current cumulative group count, i.e., $S_i = S_{i+1} + G_i$
9. If current cumulative group sum, S_i, is equal to or greater than target number of peaks, N, do 10, else go back to 7
10. Set threshold to the current count and stop

The gradient magnitudes determined by application of the Sobel edge-detection filters, provides input for this algorithm.

3.2 Sample Edge Detection Result

Sample results are shown in Fig. 3. Fig. 3a is a typical image, and Fig. 3b is the same image after it has been converted to grey-scale and Sobel edge detection has been applied to it.

Figure 3a: Sample Image **Figure 3b**: Sample Image after Sobel Edge Detection

Figure 3: Sample Sobel edge detection results

IV. Edge Thinning

Edge-detection often yields edges several pixels thick. This can make further processing of the image unnecessarily processing time and memory consuming, and "distracts" feature detection processes from important but salient features of the image. The objective of edge thinning is to reduce edges to unit thickness without losing any information about the connectedness of edges or introducing any form of distortion to the image.

Several thinning algorithms exist. The most popular method is the non-maximum suppression method. This method works by removing edge responses that are not maximal across each section of the edge direction in their local neighbourhood. However, the result of this method is still under-thinned in some places and removes real edges in other places [9].

[9]have proposed another method based on comparing gradient magnitudes within 3 x 3 neighbourhoods. It produces more accurate results than the non-maximum suppression method, and also has the added advantage of minimizing the use of the edge direction, which introduces a lot of arc tangent calculations.

This work found that the method of [9]produces very good thin edges except that sometimes it loses information about edges that are significant in the context of the original image, and that would also be helpful for robot navigation. A slight modification has been proposed to step 1 of their method that has solved this problem.

Steps 0 and 1 of their method follows:

Step 0: Select an unprocessed edge point

Step 1: Determine number of edge points, n, in the immediate neighbourhood of the current point.

If $n \leq 2$, set current point to a non-edge point, i.e., consider as noise

else, go to step 2.

The modification to step 1 is:

Step 1: Determine number of edge points, n, in the immediate neighbourhood of the current point.

If $n = 0$, set current point to a non-edge point.

If $n = 1$, then find out the number of neighbouring edge points, nn, of the 1 neighbour.

If $nn > 1$, the current edge point is maintained otherwise it is made a non-edge point.

If $n = 2$, maintain as edge point

If $n > 2$, go to step 2.

Further processing is done exactly according to step 2 and further steps described in [9].

4.1 Sample Edge Thinning Result

Fig. 4 shows theComparison of the results of the thinning method of[9]and the modified version of it used in this work. Fig. 4a is a sample image. Its edge image is shown in Fig. 4b after the application of the Sobel operators. The results of the algorithm of [9]are shown in Fig.4c and the results of the modification by the

current work is shown in Fig. 4d. Although the method of[9]results in a cleaner result, it loses important lines such as the door border highlighted in Fig. 4b.

Figure 4a: Sample Image

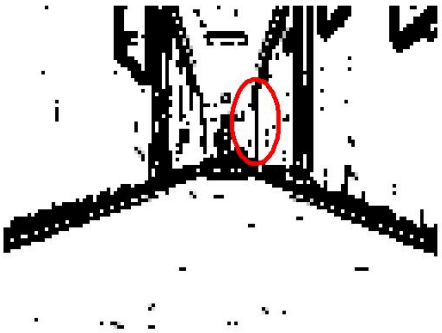

Figure 4b: Sample image after application of the Sobel Operator

Figure 4c: Sample image thinned with the method of [9]

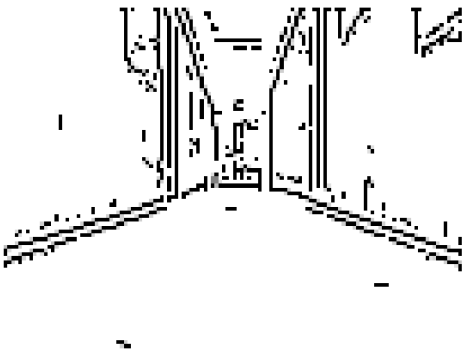

Figure 4d: Sample image thinned with modification to the method of [9] by this work

Figure 4 Comparison of the results of the thinning method of [9]and the modified version of it used in this work

V. Conclusion

In conclusion, this paper presented the pre-processing scheme used for a vision system for a self-navigating mobile robot which relies on straight line detection using the straight line Hough transform, as part of a bigger process of mobile robot self-navigation based on visual data. The scheme starts with image capture by a camera mounted on a mobile robot andends with a representative binary image optimized for straight-line detection using the Hough transform.It includes image re-sizing, conversion to gray scale, edge detection using the Sobel edge-detection filters, and edge thinning with a newly developed method that is a slight modification of the method of [9].

The newly developed thinning method has been found to yield thinned images more suitable for later stages of the capture-process-navigate cycle of this work. It enabled detection of more navigationally important features at later stages of the overall vision system, and is more accurate than other thinning methods such as non-maximal suppression commonly used, while minimizing the use of processor intensive functions such as arctangent calculations. It relies on the gradient magnitudes and angles provided by edge-detection using the Sobel filters. Other edge-detection methods, for example using the Laplacian edge-detection filters, do not provide both of these.

Threshold for determination of edges after application of the Sobel filters, was chosen automatically by targeting a fixed number of edges. This works for this application as images are generally similar. This would not work for applications where images varied a lot.

The size chosen for images in the schema presented is also a direct result of the nature of the specific application in question. Other applications would most likely do better with other image sizes.

Output from the pre-processing scheme presented provides input for the remainder of the vision-based self-navigation system for a mobile robot, which works by detecting and interpreting lines to find navigationally important features.

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.

References

[1]. 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

[2]. 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

[3] 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.

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

[5] R. M.Inigoand R. E. Torres, Mobile Robot Navigation with Vision Based Neural Networks, Proc. SPIE 2352, Mobile Robots IX, 2353, 68-79, 1995.

[6] X. Yun, K Latt and 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.

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

[8] V. F. Leavers,Shape Detection in Computer Vision Using the Hough Transform(London: Springer-Verlag, 1992).

[9] 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.