International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

One Point Perspective Vanishing Point Estimation for Mobile Robot Vision Based Navigation System

Gideon Kanji Damaryam

Abstract: ***Where it can be found, the vanishing point in a one point perspective image can facilitate analysis of the image. For the purpose of a vision system for a small mobile robot self-navigating within a rectilinear indoor environment such as along the corridor of a faculty building with one point perspective, knowing where the vanishing point in an image captured for navigation purpose is, can be very helpful in recognition of high level features such as corridors and doors. This paper presentsa parameterised algorithm developed to estimate the vanishing point in such an image of the inside of a corridor. It takes advantage of certain characteristics of vanishing points from one point perspective, and the nature of the images encountered in this work. This algorithm is based on likelihoods. It has worked in most cases, been undecided in other cases.***

Keywords: feature recognition, image processing, path detection, self-navigation, vanishing point estimation

1. Introduction

[1] and [2] have presented a method to completely specify important lines in an image captured within a rectilinear environment for the purpose of vision-based self-navigation of a small robot.

In [1], an implementation of the Hough transform is used to determine the angle, θ, of aline relative to the vertical and its distance, ρ, from the centre of the image. [1] uses the polar form of the equation of a straight line,

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

proposed by [3].θ and ρ can be used to specify the slope of the line, *m*, and its intercept on the vertical axis, *c*, by rearranging equation 1 in the gradient-intercept form of the equation of a straight line,

$$y = mx + c \qquad \text{.. . . Equation 2}$$

This yields

$$y = \rho\cos ec\theta - x\cot\theta \qquad \text{... Equation 3}$$

giving

$$m = (-\cot\theta) \qquad \text{, ... Equation 4}$$

and

$$c = \cos ec\theta \qquad \text{... Equation 5}$$

Starting from where [1] stops, [2] has then presented a method to specify the end-point of the lines found in terms of coordinates. It works by finding valid sub-lines of lines that have been found using the Hough transform (i.e., lines whose θ and ρ values) as described in [1]. A valid sub-line is a collection of points on a found line, sufficiently close to each other and numerous enough to be considered parts of the same sub-line, and far away enough from any other points along the same line so that those other points cannot be considered to be on the same sub-line. Details of thresholds for being numerous enough, and close enough (or not close enough) are presented in [2]. Any line found from the Hough transform using the method of [1], but not found to have any valid sub-line using the method of [2], is dismissed as a false recognition. This elimination of false recognitions is in addition to other means of minimization of false recognitions detailed in [1].
Before the method in [1] can be used, images are pre-processed following a series of steps detailed in [4], which involved sub-sampling (to obtain a smaller but still useful sized image), edge-detection and thinning, and result in a thinned binary image. Figure 1 shows a sample image along with completely specified lines determined using the methods specified in [4], [1] and [2] in that order. Figure 1a shows a reduced version of the original image with lines important for this work, navigationally important lines (NILs), cycled in red. Figure 1b shows the completely specified lines superimposed on the pre-processed version of the image also cycled in red. The pre-processed image has black lines showing outlines of major areas of the image in black, and automatically detected lines in colours other than black.

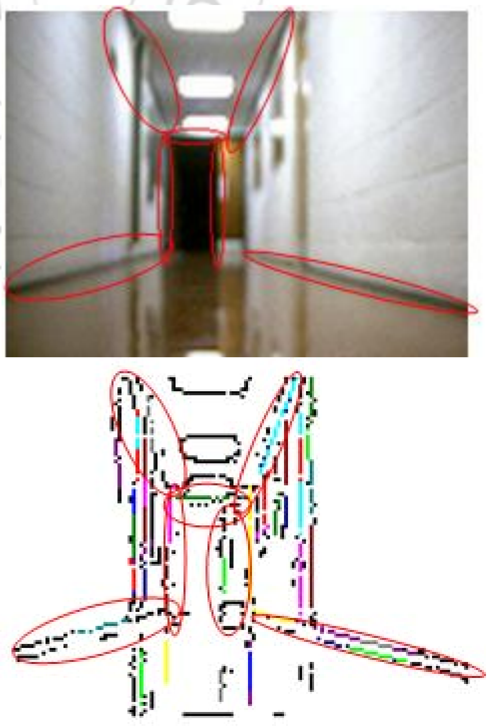


Figure 1: (a) Typical captured image (b) Typical image with important lines automatically detected superimposed on pre-processed version of image

Lines such as those shown to have been found in figure 1b are used to recognise high level features such as corridors and doors. To do that, it was found that knowledge of the location of the primary vanishing point is very helpful. Estimation of the location of the primary vanishing point is presented in this paper. Recognition of high level features has been presented in [5].

The vanishing point discussed in this paper is vanishing point from one-point perspective. "In graphical perspective, a vanishing point is a point in the picture plane that is the intersection of the projections (or drawings) of a set of parallel lines in space on to the picture plane. When the set of parallels is perpendicular to the picture plane, the construction is known as one-point perspective and their vanishing point corresponds to the oculus or eye point from which the image should be viewed for correct perspective geometry" ([6]).

2. Related Work

There has been considerable work related to estimation of vanishing point in machine vision/image processing in general, and path detection in particular.

One that is closely related to this work is the work of [7]. They have developed a model for use in Advanced Driver Assistance Systems (ADAS) for intelligent vehicles. The model detects multiple lanes on highways by detecting vanishing points. Where the lanes are not parallel, such as where a lane veers off the other lanes, or where lanes merge, their model is capable of detecting multiple vanishing points.

[7] pre-process the input image by sub-sampling and converting to a black and white image. Then they detect candidate edges of lanes using the probability Hough transform to detect important straight lines in the image. Although some lanes are not straight, they argue that due to the limited field of view of the cameras they use, the portions of lanes of immediate interest are bounded by lines that are approximately straight. They use the lines from the Hough transform to find vanishing point(s). To do so, they pair detected lines to find their intersection points. Intersection points vote for possible vanishing points on a probability map. Likely vanishing points show up as local maxima on the probability map.They use a Gaussian filter to generate a probability map of vanishing points.

[7] considered clustering points close to each other to decide on (possibly multiple) vanishing points, but decided that would be too noisy for their application. Instead they use what they call a biologically inspired model which involves a *winner take all* approach to select a vanishing point, and then an *inhibition on return*method to prevent that vanishing point from being considered in further iterations. That way, they are able to find other possible vanishing points.

[7] is similar to the work reported in this paper in a number of ways. Both works pre-process input images by resizing and conversion to binary image. The difference between the two pre-processing schemes is that [7] is able to take advantage of the large difference in intensity between the dark colour of roads and the bright colour of lane markings, whereas this work went through edge-detection and thinning to obtain binary images ready for application of the Hough transform. Both works use the Hough transform to detect straight lines. [7] have to approximate curves in lanes to straight lines in some cases since lanes on freeways curve, whereas the current work is limited to rectilinear indoor environments where edges of corridors, doors and ceilings are all straight. [7] targets multiple lanes which may not be parallel to each other and so lead to multiple vanishing points while the current work targets a single corridor surrounded by walls yielding a single vanishing point. Both works find intersection points between pairs of lines to generate candidate vanishing points, and then move towards a decision about a point being a true vanishing point based on the number of intersections points on or near it. Lines intersecting on vanishing points in [7] would primarily be edges of paths (lanes) while in the current work, edges of paths (corridors) as well as edges of ceilings would all intersect at the single vanishing point. [7] use a Gaussian filter, and their biologically inspired model to make a call about candidate vanishing points being actual vanishing points while the current work uses a number of criteria which take advantage of the unique nature of the application at hand. It factors in the fact that ceiling edges will also converge at the same vanishing point as robot path (corridor) edges, that there would be walls around the vanishing point, and that the horizon would be approximately near the horizontal midpoint of the edges.

Another work that is related to the current work, although not as closely as [7] is the work of [8]. They have also estimated vanishing point as a first step in detection of roads for intelligent vehicles. The roads they target may not be well paved or have clearly delineated edges, and there may not be apriori information about the colour or texture of the road. They use a novel adaptive soft voting scheme based on variable sized voting regions using confidence weighted Gabor filters to work out the dominant texture orientation at each pixel to estimate the vanishing point. They then use the vanishing point to constrain the search for boundaries for the road.

This work determines vanishing point for the purpose of automatic path detection for a mobile robot within an indoor rectilinear environment. As a result, images this work considers differ from images in [8] considerably in that (1) paths in the current work are guaranteed to be straight, and (2) edges of paths in the current work are generally well delineated. For this reason, the current work is able to use the straight line Hough transform to detect lines using a scheme detailed in [1], after pre-processing the input images using a scheme detailed in [2]. Candidate vanishing points are then found aspoints of intersection of multiple lines detected. This is informed by the realization that the vanishing point is likely to be the point of intersection of perspective projections of the 2 edges of the path (corridor), along with intersections of perspective projections of wall-ceiling boundaries. Scores for confidence in each candidate point are awarded according to further criteria informed by the specific nature of images in the current work as already

summarised earlier this section, and detailed in section *3. Vanishing Point Detection*.
Apart from detection of paths for auto-navigation, vanishing points have been detected for other purposes. An example of another purpose is camera calibration as done in [9].

3. Vanishing Point Detection

The algorithm for estimation of vanishing points starts with determination of points of intersection for all pairs of significant lines found from the Hough transform. Because this work is done with images taken from a rectilinear corridor environment with one-point perspective (or close to it), in most images encountered, lines due to corridor-floor boundaries and corridor-roof boundaries intersect at avanishing point. This means that the vanishing point can be expected to have at least a few lines intersecting on it.

Suppose that (θ_i, ρ_i), $i = 1$ to n, n being the number of significant lines determined from the application of the Hough transform, is the set of significant lines resulting from application of the Hough transform and subsequent post-processing to the point of dismissing lines without valid sub-lines in them, i.e.

$$\rho_i = x\cos\theta_i + y\sin\theta_i,\ i = 1 \text{ to } n \text{ ... Equation 6.}$$

Vanishing points are determined by finding points to which a majorityof pairs of lines appear to converge. This can be achieved by determining intersection points of every pair of lines. To simplify this, the set (m_i, c_i) of corresponding gradients and intercepts are found using equations 3, 4 and 5.

When intersection points have been found, those in close proximity to each other are clustered to make up for rounding errors in line determination and intersection point calculations. A note is made of the number of points contributing to each cluster, and of the pairs of lines which intersect at those points. This information is important for later processing.

Each cluster of intersection points found is examined against the following criteria:

1) Whether or not it is on the horizon: For images in this work, the horizon will normally be around the horizontal midpoint of images as the camera view is set approximately parallel to the ground. Only points on or very close to the horizon are considered further.
2) Number of lines that intersect on point: Points are ranked according to how many lines intersect on them. If more than 10 clustered intersection points are found, only the 10 with the highest number of intersecting lines are considered. If the number of clustered intersection points is less than or equals to 10, then all points are considered for further processing.

Each point is assigned a weight. 10 is the maximum weight and is assigned to the point with the highest number of intersecting lines, 9 is assigned to the point with the second highest number of intersecting lines, etc.
Where two or more points have the same number of intersecting lines, they are assigned the same weight and subsequent points are assigned weights which reflect the number of points with a higher number of intersecting lines than them. If 3 points have the same number of lines and are assigned a weight of 8, for example, the next point is assigned a weight of 5, not 7, as weights 6 and 7 would have been assigned but for the equality.

For criteria 3 to 5 below, a categorisation of significant lines found is done based on their θ values. This categorisation is done into the classes "vertical to the left of", "vertical to the right of", "horizontal above", "horizontal below", "slash through", "backslash through", and "vertical on". These categories are elaborated in table 1 which follows.

1) Whether or not there is an "X" on the point: There are two parts to this criterion. One part is fulfilled if there is a "slash through" line on the point under consideration. If it is met, a weight of 1.5 is assigned. The second part is met if there is also a "backslash through" line on the point. A further 1.5 is assigned to the point if this is the case.
2) For the purpose of step 3, "slash through" and "backslash through" are as defined in table 1.
3) Whether or not there is a "rectangle" around the point: Is there a set of four (or more) significant lines which include:
   - one that is approximately "horizontal above" it (a weight of 1.25)
   - one that is approximately "horizontal below" it (a weight of 1.25)
   - one that is approximately "vertical to the left" of it (a weight of 1.25)
   - one that is approximately "vertical to the right" of it (a weight of 1.25)

Horizontal and vertical to the left (or right) are as defined in table 1.

Table 1: Descriptions for Categories of Lines

Class	Θ Lower Bound	Θ Upper Bound	Θ Range	Minimum Distance from Point	Maximum Distance from Point
Vertical to the left	-7 (or 173)	7	15	2	n/a
Vertical to the right	-7 (or 173)	7	15	2	n/a
Slash through	15	75	60°	n/a	2
Horizontal above/below	83	98	15	2	n/a
Backslash through	105	165	60°	n/a	2
Vertical on	-5 (or 175)	4	10	n/a	2

4) "Vertical on" it: If there is a vertical line passing right through or very near the point under consideration, 2.0 is subtracted from its weight.

The point that scores the highest against these criteria is returned as the vanishing point, provided the total of points it has scored is at least 13. If 13 is not reached by any point, there is not enough evidence to confidently point out the vanishing point.

All the parameters in this algorithm were decided after trying out the algorithm several times with several typical images.

4. Results

Figure 2 shows the (correct) winning point (-6, -16) from a sample run (by a red dot). The winning score was 13.75. Table 2 shows a breakdown of the scores for the top ten points.

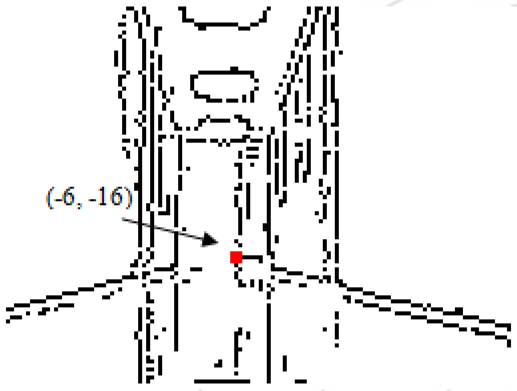


Figure 2: Sample Vanishing points found

Table 2: Breakdown of scores for top 10 intersection points

Point (x,y)	*Criteria*	*Score*	*Cummulative Score*
-18 -18	num of intersecting lines	10	10
	has slash	1.5	11.5
	verToRight	1.25	12.75
	verToLeft	1.25	14
	horAbove	1.25	15.25
	verLineOn	-2	13.25
-6 -16	num of intersecting lines	9	9
	has slash	1.5	10.5
	Backslash	1.5	12
	verToRight	1.25	13.25
	verToLeft	1.25	14.5
	horAbove	1.25	15.75
	verLineOn	-2	13.75
-31 -9	num of intersecting lines	8	8
	has slash	1.5	9.5
	verToRight	1.25	10.75
	horAbove	1.25	12
	verLineOn	-2	10
0 -16	num of intersecting lines	7	7
	has slash	1.5	8.5
	Backslash	1.5	10
	verToRight	1.25	11.25
	verToLeft	1.25	12.5
	horAbove	1.25	13.75
	verLineOn	-2	11.75
-4 -21	num of intersecting lines	7	7
	has slash	1.5	8.5
	Backslash	1.5	10
	verToRight	1.25	11.25
	verToLeft	1.25	12.5
	horAbove	1.25	13.75
	verLineOn	-2	11.75
-13 -17	num of intersecting lines	7	7
	has slash	1.5	8.5
	Backslash	1.5	10
	verToRight	1.25	11.25
	verToLeft	1.25	12.5
	horAbove	1.25	13.75
-32 -21	num of intersecting lines	7	7
	verToRight	1.25	8.25
	horAbove	1.25	9.5
	verLineOn	-2	7.5
-1 -15	num of intersecting lines	3	3
	has slash	1.5	4.5
	verToRight	1.25	5.75
	verToLeft	1.25	7
	horAbove	1.25	8.25
-21 -12	num of intersecting lines	3	3
	has slash	1.5	4.5
	verToRight	1.25	5.75
	verToLeft	1.25	7
	horAbove	1.25	8.25
	verLineOn	-2	6.25
-31 -16	num of intersecting lines	3	3
	verToRight	1.25	4.25
	horAbove	1.25	5.5
	verLineOn	-2	3.5

Table 3 summarises other sample results. In all the results shown, the vanishing point was found with confidence since the winning points scored 13 or above, except in one case. No point was found that was clearly wrong.

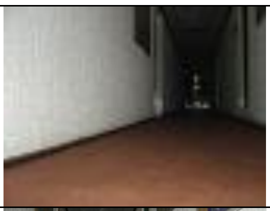
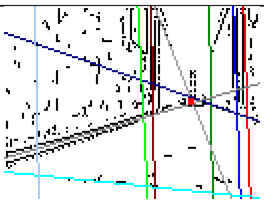
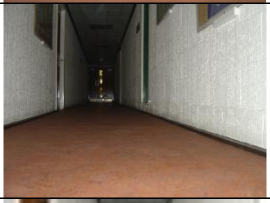
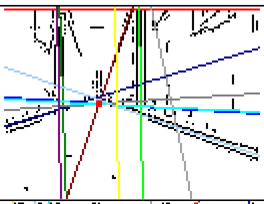
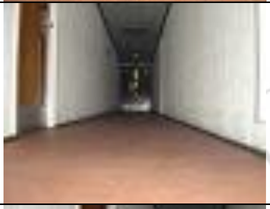
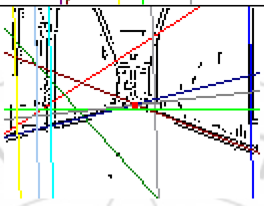
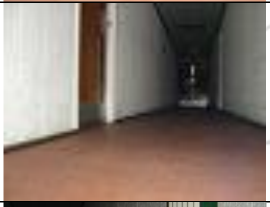
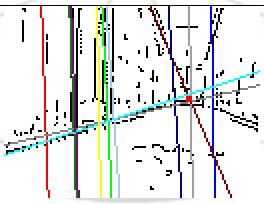
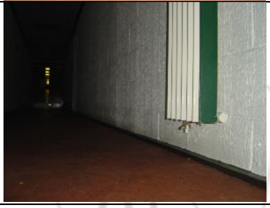
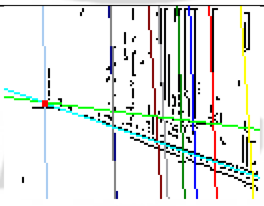
5. Conclusion

An algorithm has been presented to estimated vanishing point for images taken by a robot for self-navigation with one-point perspective (or close to it). The algorithm takes advantage of the nature of such vanishing points, and the particular types of images encountered in this work. It uses certain scoring parameters which have been manually developed for the particular problem at hand. It has been successful in finding the correct vanishing point most of the time, and where it cannot, has been able to indicate that it cannot make a call. Actual false recognitions have been rare.

This algorithm can be adapted for other applications requiring determination of vanishing point from one-point perspective with likely changes to the scoring parameters used. Genetic algorithms or other automatic parameter optimizers can be used to choose the parameters if there are sufficient numbers of sample images.

The algorithm presented will most likely require major modifications to work with higher point perspectives.

Table 3: Some results from VP determination scheme

Image ID	*Original Image*	*Pre-processed Image with Lines Found and* VP	*Estimated* $VP(x,y)$	VP *found* (x,y)	*Weight*
1648			30, 2	29, 1	16.75
1649			-17, 3	-17, 3	16.75
1650			-1,1	-3,0	14.25
1651			28,2	28, 2	13.5
1652			-43,-1	-44,0	12

6. 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, "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] Duda and Hart. 1973. "Pattern Classification and Scene Analysis" (New York: Joh Wiley and Sons, 1973).
[3] 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
[4] 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.
[5] Anonymous, "Vanishing Points", https://en.wikipedia.org/wiki/Vanishing_point, Accessed February 5th, 2016.
[6] C. Li, Y. Nie, B. Dai and T. Wu, "Multi-lane Detection Based on Multiple Vanishing Points Detection", In Proceeding of the 6th International Conference on Graphic and Image Processing (ICGIP), 2014
[7] H. Kong, J.-Y. Audibert, andJ. Ponce,"Vanishing point detection for road detection", In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,(CVPR) pp. 96-103, 2009
[8] B. Li, K. Peng, X. Ying and H. Zha, "Simultaneous Vanishing Point Detection and Camera Calibration from Single Images", International Symposium on Visual Computing(ISVC), pp. 151-160, 2010.
[9] 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