

Road Sign Detection

With Multi-Foveation to influence Gaze Direction

Navanit V. Arakeri

Department of Electrical Engineering

Stanford University

Abstract

I present an algorithm that detects sign-like regions from an image taken from an on-board forward pointing camera. The algorithm is designed to detect most common sign shapes. In addition I propose foveation as an effective technique for non-intrusive gaze influencing mechanism. In this algorithm road signs are detected based on rules that restrict color, shape and position of sign candidates in the image. A template matching mechanism is used to create a mask covering the sign-like regions; the mask is then used to foveate the image to influence gaze in the direction of the signs. The method is highly versatile and can be easily modified to include different geometric shapes.

Introduction

The primary motivation behind the project was to enhance driver awareness of road signs while retaining context information. It has been shown [1, 2] that drivers often miss road signs because they are usually concentrated in an area of the road that is inherently distracting (exits, forks, junctions, etc.) and that require increased driver attention on other aspects of driving. Other application includes uses in autonomous vehicles and training along with Optical Character Recognition (OCR) capabilities.

Road signs in the continental United States are typically categorized based on color and shape. Internationally, it is more common to find signs that are distinguished based on color only, with only rectangles and circles used as shape differentiators. On the other hand, the following shapes are used commonly in the United States: Rectangles, Triangles, Octagons, and Diamonds. The following colors are used: Red, Yellow, Green, White, Brown, and Orange.

Prior work involves primarily color based segmentation, such as in [3] where redness measure is used to detect Stop, Yield and Do Not Enter signs. Escalera et al [4] also start with color matching and then add corner matching to detect rectangles and circles. Huang and Hsu [5] use color and shape to detect circular and triangular shapes. They then control the camera to zoom in on candidates for a closer view. Shaposhnikov et al. [6] use color segmentation followed by histogram analysis of edge elements to determine shape. This technique is robust when working with rectangles and circles; however it does not extend easily to other shapes (difficult to distinguish triangles, octagons and circles).

The approach taken in this paper follows the general trend in previous work, however I also include position information and use shape as the first criterion to narrow down candidate signs.

Sign Detection

Motion blur, which is the component that is expected to cause the biggest problem, can be completely eliminated using a large aperture and a high shutter speed. Most images were taken at 1/2000th of a second shutter speed, which takes care of motion blur even at

freeway speeds (65mph).

Most shape detection techniques use a windowing technique with allowed scaling and shearing (Rucklidge's algorithm in machine vision for example) with a template matching algorithm. The windowing technique is useful for generalized shapes and limited scaling. However, in this application, shapes are fixed and geometric with a small subset of line segments and angles. Furthermore, the range of scaling required is not small. This additional information was incorporated in the algorithm by performing a Hough Transform which significantly reduced computation time by eliminating the need to window the template, and by reducing the need to perform any significant degree of scaling. The algorithm is explained below. The algorithm is fine-tuned to work on 800x600 images, the thresholds will have to be modified for different sizes.

A basic edge detector is used to generate an edge image. Sobel edge detection was used throughout, with automatic threshold detection. The edge detection is a crucial step and subsequent steps depend on good edge detection.

SHAPE * MERGEFORMAT SHAPE * MERGEFORMAT **Figure 1.** The original and edge detected image. Edge detection is a critical step.

The edge image is then processed by performing a set of closing and dilation operations. Closing fills in ragged edges, and the dilation offers better purchase for template matching to be performed later. A Hough transform is then performed on the edge image. Peaks in the Hough transform correspond to pixels that lie on a straight line. The transform is performed at 15 degree intervals to capture 0, 30, 45, 60 and 90 degree edges. An example image of a finer grained Hough Transform is shown below.

Figure 2. Hough Transform of an image. Two peaks are marked.

Those pixels corresponding to a value greater than a quarter of the average value of the Hough Transform are counted as peaks. The low threshold is necessary to include smaller edges that may be a part of a distant sign in the image or a rectangle with an aspect ratio that is not close to unity.

The Hough transform contains enough information to recreate the edge image with only edges that correspond to the above 15 degree intervals. Line segment information can also be extracted by detecting the end and beginning of line segments from the peaks in the Hough transform.

SHAPE * MERGEFORMAT

Figure 3. Line segments derived from the Hough Transform.

Line segments that are at 0, 45, and 60 degrees are then identified. Any line segment within a fixed interval from the end points of these line segments are then analyzed to determine if

these segments correspond to rectangles, octagons, diamonds or triangles. If such a relationship is found, a template corresponding to the length of the line segments is created. Note that this implies that any pair of line segments will generate a template. This technique is superior to corner matching, since the scale of the template is immediately apparent. Any given line segment may generate two templates (one corresponding to each end point relationship).

The corners of the two line segments that generated the template are used to snap the template on to the cleaned-up edge image. A Hausdorff distance is calculated for each candidate and inverted (multiplicative) to determine a score. The score is then scaled by the maximum dimension of the template. The inversion is necessary to go from a distance metric to a score, the smaller the distance metric, the higher the score and vice versa. The scaling is performed since only a directed hausdorff distance (i.e. only $h(\text{template}, \text{edge image})$) is calculate and not the true hausdorff distance. This results in possibilities for a point template to have the best score, scaling eliminates this possibility. The true Hausdorff distance is not calculated (i.e. $\max(h(A,B), h(B,A))$ instead of just $h(A,B)$) because of the existence of text within the signs, which contribute edges and foreground pixels, thus completely skewing the distance metric.

The templates are pasted on to their respective positions on a blank image. A binary seed fill is performed on this image with the top left pixel as the seed. Inverting this image results in a mask corresponding to the current candidate templates. The mask is used to find the predominant color component under each connected component of the mask in the original image. RGB ratios are used to arrive at a score:

$$\text{Color score} = \text{scale} * [(r/b, r/b, g/b) - (\alpha, \beta, \gamma)]$$

The (α, β, γ) pairing is determined for each color (red, blue, green, white, etc.) empirically and remain constant in the tests that were performed. A candidate that did not contain any of the sign colors would generate a low or negative score. The scaling constant is empirically determined and is used to bring it on par with the Shape score values.

Once the Shape and Color scores are determined, the candidate mask image is compared with a holding image that contains a Position score for each pixel. Pixels corresponding to the top, left or right one thirds of the image were given a high score. Pixels corresponding to the bottom center of the image were given a low score (all score are non-zero). The values of these scores are subjective and of the same order of magnitude as the Color and Shape scores. Some degree of tweaking was required to arrive at a Position score map that worked well.

The foreground pixels in the candidate mask image are replaced with the weighted sum of the three scores (Shape, Color and Position). This algorithm used equal weights for all three scores. The image is then thresholded (threshold determined by trial and error) and

any connected component that contained a larger than 50% pixel count that were above the threshold is retained (the 50% is actually arbitrary, this could be changed to any value and accounted for by modifying the previous thresholds and scores maps).

SHAPE * MERGEFORMAT

Figure 4. The final mask.

Foveation

The use of foveation to influence gaze is a novel application and needs further study. The result of the steps so far is a mask that deigns to covers the sign-like regions in the original image. Foveation is performed by creating a pyramid of progressively blurred (gaussian low pass filter) images and using the euclidean distance of pixels from the foreground pixels in the mask to determine which image to replace the corresponding pixel with. Linear interpolation is used to create a smooth boundary when the distance changes enough to warrant a change in the decision to pick a particular blurred image. This is a simplistic implementation of foveation, but it serves the current purpose very well. Foveation was originally designed to improve compression, and the techniques used in those applications often include viewing distance and the blurring factor as user defined variables before foveating. In this application, as long as the signs remain at full resolution (unblurred) and the farthest pixels are not too blurred out while retaining an exponential drop in clarity, the implementation serves its purpose.

SHAPE * MERGEFORMAT

Figure 5. Foveated image

Experiments

A hundred 800x600 images were acquired using a Canon Powershot A95 and some via Google Images. All images were taken with the camera on-board a vehicle. Most images contained more than one sign. Images were taken such that the four shapes were approximately equally represented in the set. The foveation creates a harsh penalty for False Negatives (FNs) since there is a distinct possibility that those areas would be blurred out.

The table below summarizes the results on these hundred images.

# of images	FPs	FNs	# of signs	FPs	FNs
100	19	6	244	25	9

Table 1. Results summary

Analysis

False Negatives

The main contributions to FNs were failed edge detection. In cases where at least 50% of edges were found, the algorithm managed to squeeze out the correct mask. However, if

fewer than 50% of the edges of a given sign were detected, the Shape score was often so low that the sign would pass under the threshold and be foveated out. This problem could be mitigated by using different weights for the three scores, but these weights may have to be calculated on the fly based on current lighting conditions. The tests reported above were performed in daylight with reasonable sunlight (especially the ones at 65mph for the fast shutter speed). Test images taken at night showed persistent failure of edge detection, even with very low thresholds. Gradients and reflections contributed to false edges and undetected true edges.

Though uncommon, a sign that was positioned such that it was viewed at a markedly skewed angle also contributed to FNs since the Hough transform would not detect angles that are distorted from 0, 45 or 60 degrees.

False Positives

Areas in the image that looked identical to road signs contributed significantly to false positives. In particular, predominantly white or green vehicles, boards, or rods positioned near the edges of the image are necessarily indistinguishable from signs based on these three rules alone. One way to mitigate this is to use consecutive frames to determine the motion of the candidate regions with respect to the camera. This would decrease the risk of other vehicles being counted as signs.

Computation

This project was implemented in MATLAB, with no emphasis on improving performance (other than the usual practice of eliminating loops in MATLAB). A port of this algorithm to C/C++ would give a realistic idea of the speed and processing times required for 800x600 (MxN) images. Intuitively, it may appear that the Hough Transform is the bottleneck in this algorithm, with every pixel being checked for membership against every other pixel, thus giving $O((M*N)^2)$. However the (ρ, θ) value of every foreground pixel can be computed on a single pass over the image, and the linear relationships can be computed geometrically, much faster than $O(M*N)$ (since the edge image is not dense with foreground pixels). If the number of foreground pixels in the edge image is 'e', then the overall performance of the Hough Transform will be $O(M*N + e^2)$. Since e is largely independent of M and N, and $e^2 \ll M*N$, the Hough Transform has a performance of $O(M*N)$.

The overall performance of the algorithm is $O(M*N)$ since the edge detection, as well as the color and position score calculations are also $O(M*N)$. This may not look like much of an improvement over the windowing template technique (which is also $O(M*N)$), however this only means that the improvement is not of the order of M or N. It is very likely that the Hough Transform saves a fairly large number (at least 10, with four distinct shapes and at least 3 scalings for each and multiple aspect ratios for rectangles, very likely more) of passes over the image matrix. It can also improve the accuracy of the templates that are used, since most windowing techniques use a fairly coarse scaling and aspect ratios range

to improve performance.

Conclusion and Future Work

The algorithm and implementation works well over a wide range of images and signs in daylight. However there is some opportunity to improve edge detection under different conditions and lighting variations. As mentioned earlier, an implementation in C/C++ would give a much better idea of the speeds involved for each of the steps in the algorithm. The Vision Group at Stanford University's Psychology Department has expressed an interest in moving forward in conducting studies regarding using foveation for gaze influence. Determining the rate of exponential drop-off in the foveation in order to influence gaze while retaining sufficient context is one of the primary goals.

Acknowledgements

I would like to thank Dr. Ashok Popat, and Dr. Dan Bloomberg for giving me the opportunity to work on this project and providing the basis for most of the techniques used in this project.

References

[1] "Vision, Visibility and Perception in Driving", Hills, B.L. *Perception*. 1980;9(2): 183-216.

[2] "Direct observation of driving, self reports of driver behaviour, and accident involvement" West R, French D, Kemp R, Elander J., *Ergonomics*. 1993 May;36(5): 557-67

[3] "Real-Time Histogrammic Approach to Road Sign Recognition". Estevez, L. and Kehtarnavaz, N. A., *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation*, 95-100. 1996. San Antonio, TX.

[4] "Road Traffic Sign Detection and Classification," de la Escalera, A., Moreno, L. E., Salichs, M. A., and Armingol, J. M., *IEEE Transactions Industrial Electronics*, vol. 44, no. 6, pp. 848-859, 1997.

[5] “Road Sign Interpretation using Matching Pursuit Method” Huang, Chung-Lin and Hsu, Shih-Hung. *IEEE International Conference on Pattern Recognition (ICPR2000)*, 1329-1333. 2000. Barcelona, Spain.

[6] “Road Sign Recognition by Single Positioning of Space-Variant Sensor Window”, Shaposhnikov, Lubov N., Podladchikova, Alexander V., Golovan, Natalia, Shevtsova, A., Hong, Kunbin, and Gao, Xiaohong. *Proc. 15th International Conference on Vision Interface*, 213-217, 2002. Calgary, Canada.

APPENDIX

1.

Original and mask, no FPs and no FNs

2. Original and mask, 1 FP and 1 FN

3. Original and mask, FN due to skew, and FP due to sign mimicking (this test image not taken from on-board vehicle)