18-551, Spring 2002
Group 7, Final Report

What's Your Sign?

Improving Driver Safety with Road Sign Recognition

Michael Kaye (mhk@andrew.cmu.edu)
Anusha Krishnakumar (anushak@andrew.cmu.edu)
Loris Stehle (lstehle@andrew.cmu.edu)

May 6, 2002

TABLE OF CONTENTS

1.0 INTRODUCTION

Driving, especially on unfamiliar roads and highways places a large number of demands on the operator of a vehicle. It is not uncommon for drivers to drive right past a road sign without realizing (or realizing too late) that they failed to notice it. Overlooking a road sign could have severe, even fatal consequences.

Driver safety is a predominant concern in America and around the world. According to the Pennsylvania Highway Information Association, 1 out of every 34 Pennsylvanians will be involved in an automobile accident every year [1]. The group was inspired by these concerns and decided to implement a road sign recognition algorithm to enhance driver safety.

2.0 PROJECT DESCRIPTION

2.1 THE BIG PICTURE

The road sign recognition project is a smaller part of a much larger system. Recently, there has been much advancement in the on-board systems of smart vehicles. The particular system the group's work will be an essential part of is the Driver Support System (DSS). This system is designed to help the driver of a vehicle by monitoring the current traffic situation and by serving as a beneficial source of information [2]. It also assists the driver in avoiding potentially risky situations while on the road.

The aim of the road sign recognition project is to recognize road signs in a picture of a roadside scene and help the driver of a vehicle determine the type of sign that is approaching. The group assumes this picture is taken by a video camera mounted on a car. The camera first takes a long-range shot of the traffic scene, and upon detection of a road sign, zooms in to take a close-up.

The group's application focuses on three types of road signs: stop signs, do not enter signs, and yield signs. These signs are shown in Figures 1(a), 1(b), and 1(c), respectively. These signs were chosen because overlooking one of these three signs is more likely to have serious consequences than overlooking other road signs, such as no parking, no U-turn, or divided highway signs.

(a): Stop

(b): Do Not Enter

(c): Yield

Figure 1: Standard American Road Signs.

2.2 INPUTS AND OUTPUTS

The road sign recognition process consists of a detection stage and a recognition stage. The group's algorithm extracts potential road sign regions from images of road scenes based on their color and size. The group implemented the algorithm on a Texas Instruments TMS320C6701 (C67) evaluation module (EVM).

The group worked with images of road scenes that were 1600x1200 pixels in size. The group then used the Adobe Photoshop and ImageMagick applications to resize the images and convert them to a different format, respectively. The resized images were 280x280 pixels. They were then converted from their original JPEG format to a ppm format since this would be most convenient for the group to work with. The training set, on which the group performed their analyses, consisted of 39 images, whereas the test set, on which the group tested their algorithm, consisted of 36 images. The group proceeded to extract the red areas of the image (the details of which are explained in later sections), perform various calculations on them, and output the results.

Opening a Command Prompt session and calling the PC program with the name of the image to be analyzed was the first step in obtaining the final results. The group then ran the program on the EVM side and waited for an output. The current step that the algorithm was performing was displayed in the output window of the Code Composer Studio application while the program was running. The final output in this window was the type(s) of sign(s) detected, if any, and the exact location(s) of the red area(s) in the image.

3.0 POTENTIAL PROBLEMS

Most of the known recognition systems have been developed for highway traffic since this scenario is fairly simple: traffic signs are placed in visible locations, lanes are clearly marked, and change in curvature only very slightly, etc [3]. The situation becomes much more difficult when driving on urban roads. Unlike highways, urban roads often do not have well-marked lanes, road signs are often very small and frequently obscured by trees or bushes, the environment is highly colored, etc.

For these reasons, the group decided that all of the images in their database could not be considered ideal. Considering non-ideal images also poses further difficulties. The group designed their algorithm as best as they could to overcome problems such as rotation, clutter, lighting, background noise, blur, damage etc.

Figure 2(a) shows a yield sign from the group's training set that is rotated. This is the maximum angle of rotation that appeared in an image used by the group. The algorithm detects the sign correctly even though one side is rotated at a significant angle. Figure 2(b) shows clutter, where the main road sign is occluded by another sign not relevant to the project. Figure 2(c) shows a stop sign in a darker area. Lighting conditions outdoors makes it difficult for the group to identify the correct sign because of the complexity of various lighting and shaded environments. However, the algorithm did a good job of detecting this sign correctly. Figure 2(d) depicts a very common situation the group incurred in many of the signs in their database: background noise. The algorithm is tuned to detect red areas in the image of the road scene. As seen in this figure, the red building directly beneath the road sign causes problems in detection. The

yield sign is detected correctly, but the red building in the background is detected as a series of do not enter signs, which is incorrect. Finally, Figure 2(e) shows blur in a sign, where the car was moving so fast, that the image acquired by the camera was not clear. It must be noted that in all of the cases shown in Figure 2, the group's algorithm detects the main sign correctly.

(a): Rotation **(b): Clutter** **(c): Lighting**

(d): Background Noise **(e): Blur**

Figure 2: Problems associated with sign detection.

4.0 PRIOR WORK

4.1 PRIOR 18-551 PROJECTS

The group did not find any of the past 18-551 projects to be relevant to the road sign recognition project. Therefore they had to look to other research groups' work.

4.2 METHODS USED IN THE PAST

Much of the work in this area has used various methods that the group decided was beyond the scope of this class. Some attempts have been made using the color segmentation method to classify candidate regions. These regions are then grouped by a neural network classifier [4]. Other methods have specialized functions to detect signs in particular settings.

One particular group used geometric hashing to specialize in the ability to recognize signs in cluttered scenes [5]. Edge detection is performed, contours are connected and segmented into straight lines and circular arcs, and identification of these closed contours is completed. Hue Saturation Value (HSV) color space has also been employed, where HSV coordinates are calculated from RGB values. A pixel is defined as 'red' if its 'hue' lies within two thresholds and its 'saturation' is higher than another threshold. This method is used specifically to detect signs in adverse lighting conditions [6]. Other groups have proposed using a bank of nonlinear filters to obtain a scale invariant recognition system [7]. The nonlinearity in the filter bank is able to effectively detect the sign for a wide range of distances. Improvements using correlation techniques

have been made to this method to specifically account for in-plane and out-of-plane rotations of the road sign.

4.3 PROJECTS IMPLEMENTED IN A SIMILAR MANNER

The 18-551 group launched their efforts by basing their project on a very similar project carried out by two members of the Department of Electrical Engineering at Texas A&M University. The members of this group, Leonardo Estevez and Nasser Kehtarnavaz, had implemented all their modules on a Texas Instruments TMSC320C40 Digital Signal Processor [8].

Much of the methodology used by Estevez and Kehtarnavaz was also used by the 18-551 group. A foundation for the group's algorithm was obtained using the research conducted by these engineers. These methods are explained in more detail in later sections. This group however had room in their algorithm for better detection and identification of the road sign. It was up to the 18-551 group to decide the factors needing improvement.

4.4 AVAILABLE CODE

Unfortunately, the group was not able to obtain copies of previous code that could be used in the detection and identification stages. Thus, the group developed all of the C code and MATLAB code related to the algorithm from scratch.

The C code used for extracting the RGB vectors from the ppm-formatted imaged was obtained from the 18-798 class in the Department of Electrical and Computer Engineering at Carnegie Mellon University. The C code used for communication between the PC and the EVM was obtained from Lab 3 in the 18-551 class.

5.0 COMPARISON TO PRIOR WORK

The algorithm implemented by the 18-551 group contains procedures to further improve the detection and identification stages. Most importantly, morphological filters were used to merge red areas that should be considered as one large distinct zone.

5.1 MORPHOLOGICAL FILTERS

Figure 3 shows an example of why morphological filters should be used. Figure 3(a) is an image of a stop sign to be detected by the algorithm. This stop sign is obscured by the branches of a nearby tree. In Figure 3(b), without the morphological filter, the algorithm detects a small number of red areas and does not output a result. However, with the addition of a morphological filter to the group's algorithm, the stop sign in Figure 3(c) is detected as one big red area and outputs the correct result.

(a): Original image

(b): No Morphological Filter

(c): With Morphological Filter

Figure 3: Differences in output obtained when using a morphological filter.

5.2 ASSUMPTIONS ON THE NUMBER OF SIGNS IN AN IMAGE

Another way in which the group's algorithm differs from many other prior algorithms is the assumption the group makes about the number of road signs in a given image. The group assumes that, as well as being able to handle only one road sign in the image, the algorithm is able to handle either no signs in the image, or multiple signs in the image.

Figure 4 shows an image with a stop sign in the foreground, a do not enter sign in the background, and a 'no right turn' sign directly underneath the stop sign. As well as being able to determine the stop and do not enter signs correctly, the algorithm designed by the group detects the 'no right turn' sign as a 'no sign.'

Figure 4: Three signs in one image.

6.0 THE ALGORITHM

The group worked with roadside scene images in three different environments: a car, a personal computer, and on the EVM. The images that were used as inputs to the algorithm were taken from a car with a digital camera, converted to a processing-friendly size and format on a personal computer (PC), and then sent to the EVM for analysis. A data flow chart is shown in Figure 5 and the details of each step are described below.

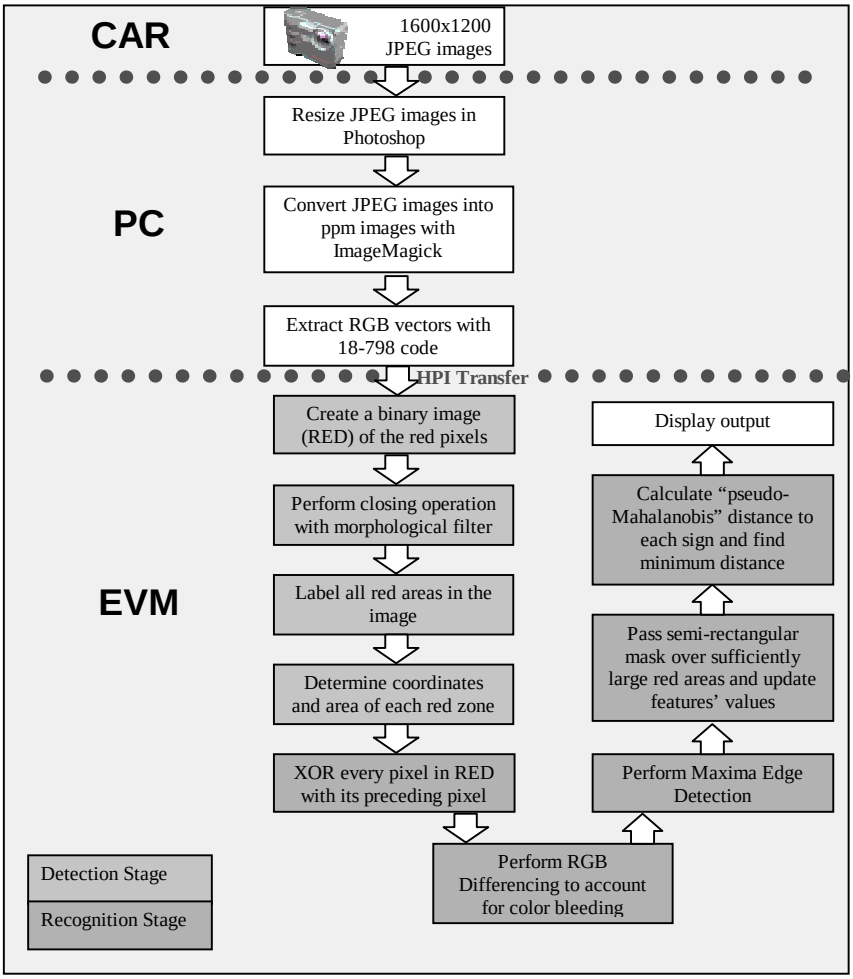


Figure 5: This data flow chart shows the details of the three stages of the road sign recognition project.

6.1 THE CAR

An Olympus D510 digital camera was used to take 1600x1200 JPEG images of roadside scenes. They were either taken by a member of the group who was sitting in the passenger seat of a car as it was being driven on a road or by a group member who was standing on the road where a car would normally be if it were being driven on that road. Thus, the images in the training and test sets are representative of the types of pictures that would be taken by a video camera mounted on a car, which is how the pictures would be acquired if this system were to be implemented in practice. The pictures were taken so that the road signs in the image occupied areas ranging from 50x50 pixels to 170x170 pixels. According to [8], a 50x50 pixel image corresponds to a distance of about 50 feet if a 6mm video camera is used to take the pictures.

6.2 THE PC

The 1600x1200 JPEG images taken by the digital camera were not conducive to processing on the EVM. 1600x1200 pixels were more than what was needed. Images of this size would consume an unnecessarily large amount of memory and they would require a lot of processing time. Also, the JPEG file format compresses the red (R), green (G), and blue (B) values of an 8x8 block of pixels. This was problematic because the group needed access to the individual RGB values of each pixel at various points in the algorithm.

To make the digital camera's pictures more suitable for processing on the EVM, the group first resized the 1600x1200 JPEG images to 280x280 JPEG images with Adobe Photoshop. Figure 6 shows a few images used in this project after they had been resized

to 280x280 pixels. These smaller JPEG images were then converted to ppm files with ImageMagick. The ppm file format was chosen because C code already existed that allowed easy access to the RGB values for each pixel in the image. This is C code that is used in Carnegie Mellon University's 18-798 Image and Video Processing class. The RGB values were extracted from the ppm file with the 18-798 C code running in Microsoft Visual C++, interleaved (i.e. …R_{n-1}, G_{n-1}, B_{n-1}, R_n, G_n, B_n, R_{n+1}, G_{n+1}, B_{n+1}…), and stored flat in a vector. Flat means the RGB values for the last pixel in a given row are immediately followed by the RGB values for the first pixel in the next row. After this vector was created for the image, it was transferred to the EVM via the host port interface (HPI).

Figure 6: These images of a (a) stop sign, (b) do not enter sign, and (c) yield sign were transferred to the EVM for processing.

6.3 THE EVM

After the image was transferred to the EVM, it was subjected to two stages of processing. The first stage of processing, the detection stage, located the red areas in the image because the three signs of interest in this project have red as their dominant color. These red areas were then passed to a second stage of processing, the recognition stage, in which the various angles of a red area's edges were used to determine which, if any, of the three signs were present in that particular red area.

6.3.1 EXTRACTING THE RED AREAS

The first step in the detection stage of the algorithm is to extract all of the red areas in the image. This is done by calculating a redness value for each pixel with the following equation:

$$\text{Redness} = R - \max(G,B) - \alpha * |G-B|, \quad \text{(Eqn. 1)}$$

where RGB are the red, green, and blue values of the pixel of interest and α is a sensitivity factor that can be varied to account for pictures taken in different lighting conditions [8]. The authors of [8] claim that the top line of a roadside image is likely to correspond to the sky and can be used to determine α. The top lines of the images used in this project did not always correspond to the sky so the group chose to keep α at a constant value of 1.

Eqn. 1 generates a redness value between –510 and 255 for every pixel in the image. This value is then thresholded at 25 to create a binary image of the red pixels in the original image. The threshold of 25 was determined by testing different values in MATLAB. The group found that a threshold of 25 produced very good results, as can be

seen in Figure 7. Unfortunately, if there are red objects in the image other than the road sign, they will be present the redness image. Notice that parts of the red brick building in the background of Figure 8(a) are present in the corresponding redness image shown in Figure 8(b).

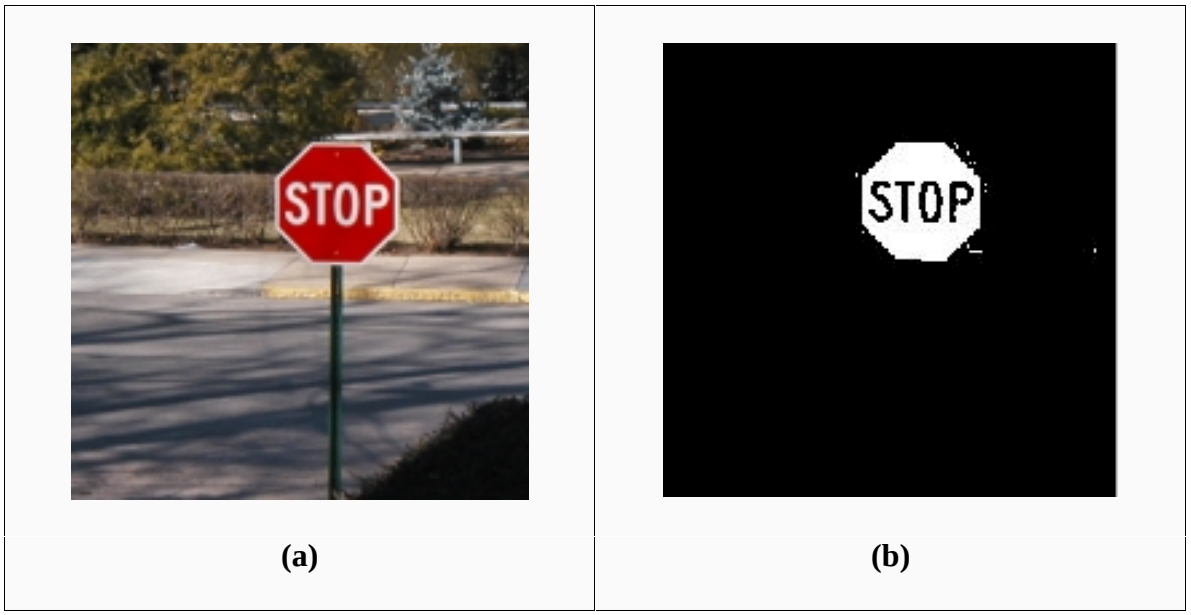
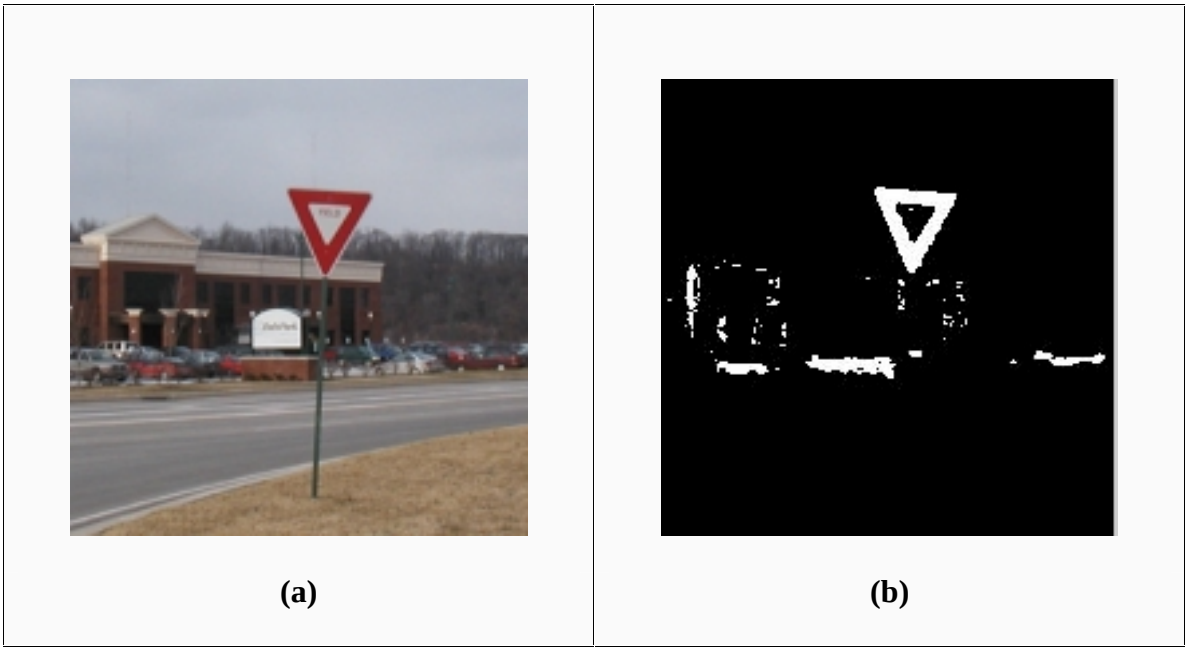


Figure 7: The image shown in (b) is the redness image created from (a).



(a) **(b)**

Figure 8: The image shown in (b) is the redness image created from (a) – notice that parts of red brick building are present in (b).

6.3.2 MORPHOLOGICAL FILTERING: CLOSING OPERATION

The second step in the detection stage of the algorithm is to apply a morphological filter to the binary image that represents the redness of the road scene picture. The type of morphological filter that the group applies, which consists of performing a dilation followed by an erosion, is called a closing operation. In general, there are two reasons why one would apply this kind of morphological filter: fill in holes inside red areas and connect two adjacent regions that are separated by a thin gap.

The group only takes advantage of the second property of closing operations. Since the remaining part of the algorithm is performed on the raw redness obtained from the algorithm described in section 6.3.1, filling holes does not have any effect on the result that the group obtains, but merging the separated red areas does.

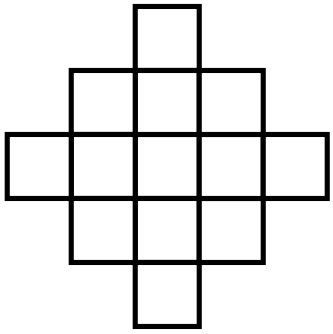


Figure 9: Shape of the structuring element used to perform the closing operation.

The structuring element that the group uses is the 5x5 circular window depicted in Figure 9. This morphological filtering was added after the group realized that a red area corresponding to a single road sign was sometimes split into two or more parts that were not 4-connected after the thresholding of the redness. This problem mainly occurred for small do not enter and stop signs, where the top and bottom halves of the sign were not connected anymore, because of color bleeding effects.

This filter was also designed to make the overall algorithm less sensitive to occlusion such as when tree branches cut through the sign, as shown in Figure 3.

Applying another morphological filter to perform an opening operation to remove the background noise (little red areas that do not belong to the road sign) is not as useful as it may seem to be. Since the group assumes that road signs must be at least a minimum size in order to be detected, the algorithm will only do the recognition stage on areas containing at least a minimum number of 4-connected red pixels. Since morphological filters are quite computationally time consuming and the improvement gained by applying an opening operation is very small, the group decided not to apply this second morphological filter.

6.3.3 LABEL RED AREAS AND DEFINE RECTANGULAR WINDOWS AROUND THEM

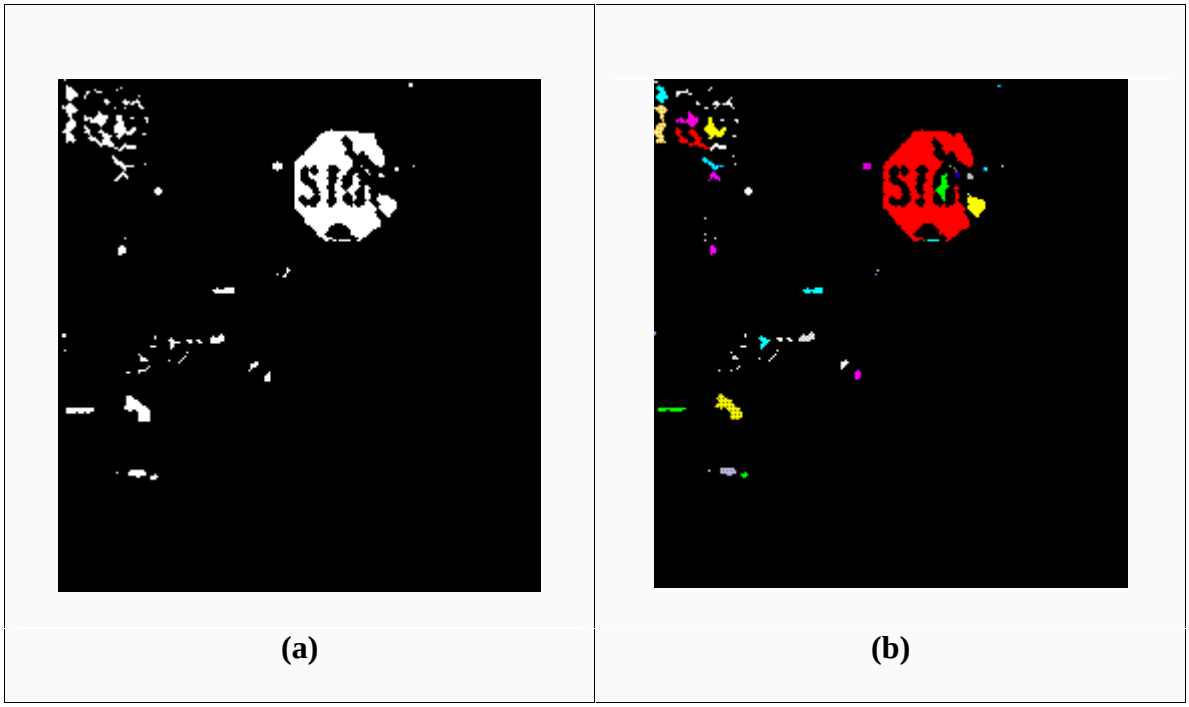
Once the group had applied the morphological filter to perform the closing operation, all red areas had to be labeled in order to be analyzed independently. As well as giving each red zone a different label, this operation also counts the number of pixels

contained in each red area in order to measure their areas. Therefore, the group only analyzed red areas bigger than a given size, experimentally set to be 1% of the area of the image (i.e. area containing at least 784 red pixels).

The group practically labeled each area as follows:

1. Scan the entire image, searching for red pixels
2. If a red pixel is found, label it as follows:
    a. If the pixel above it and the pixel on its left are not red, give the considered pixel a new label
    b. If the pixel above it is red, and the pixel on its left is not red, give the considered pixel the same label as the point above it
    c. If the pixel on its left is red, and the pixel above it is not red, give the considered pixel the same label as the point on its left
    d. If both of the points above it and the point on its left are red, and are assigned to the same label, give the considered point the same label as the point above it.
    e. If both the points above it and the point on its left are red, but are assigned to the different labels (i.e. the point that is considered connects two labeled areas), give the considered point the same label as the point above it. Then scan the image from the upper left corner to the considered point and change the label of points that have the same label as the point on the left of the considered point.

The result of this labeling operation is shown in Figure 10.



(a) **(b)**

Figure 10: Filtered redness (a) and labeled filtered redness (b).

Once all of the red areas were labeled, the group placed a rectangular window around each red area. The entire image was scanned to detect the top, bottom, left-most and right-most points contained in each area to determine the coordinates and the size of each rectangular window. Each rectangular window was then expanded by a few pixels, to make sure that the red edges were contained in it. The result of this windowing is displayed in Figure 11 below.

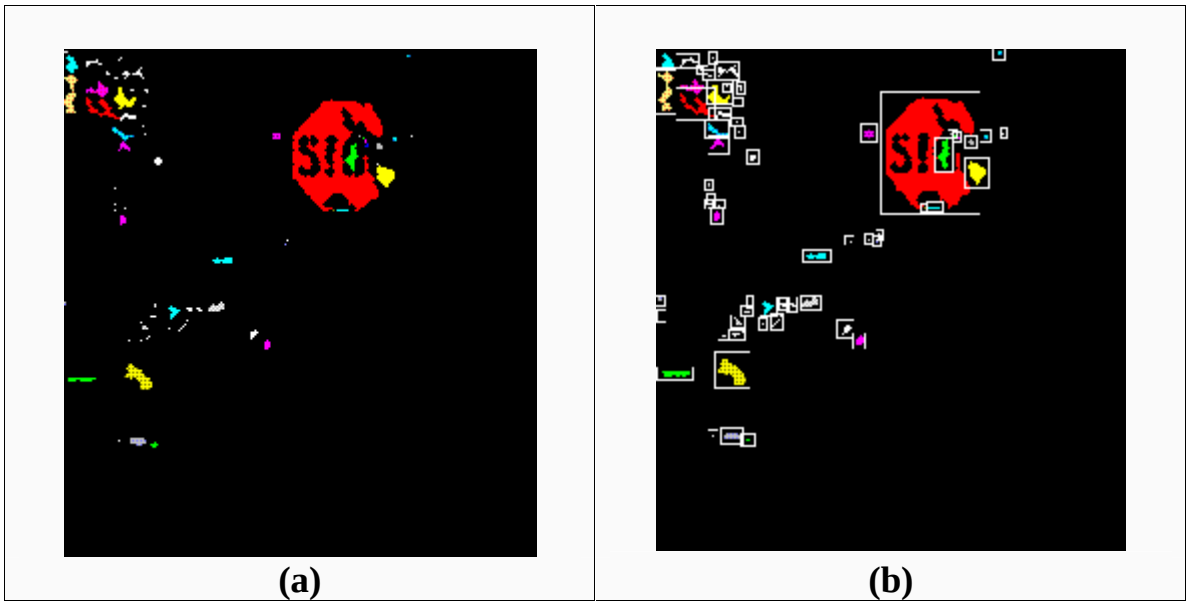


Figure 11: The image on the right hand side (b) is the windowed version of the image on the left hand side (a).

The recognition part of the algorithm was then run inside each window containing a red area bigger than the threshold.

6.3.4 EDGE DETECTION

Detecting the edges in the redness image is a three-step process. First, every pixel is XORed with the pixel to its left. The pixels in first column of the resulting image are all set to 0 since they do not have pixels to their lefts. Because a pixel is only XORed with the pixel to its left, horizontal edges are not detected. Next, the XORed image is scanned and for every pixel that is equal to 1, RGB differencing is performed. Two RGB differencing values for every pixel in the XOR image that is found to be a 1 are computed with the following equation:

$$\text{Difference}[p] = |R[x]-R[p]| + |G[x]-G[p]| + |B[x]-B[p]|, \quad \text{(Eqn. 2)}$$

where $p \in \{x-1, x+1\}$ and represents the pixel to the left or to the right of pixel x, which is the pixel in the XOR image that is found to be a 1. The pixels in first column and last column of the resulting image are all set to 0 because they do not have pixels to their lefts and rights, respectively. RGB differencing accounts for color bleeding and desaturation effects along the edge boundaries, which occur when the camera experiences sudden motion and when a light source, such as the sun, are present in the image [8]. Once the RGB differencing values are computed, the precise locations of edge points are determined with the following local maxima operator:

IF [(Difference[x] > Difference[x-2]) AND (Difference[x] > Difference[x+2])]
THEN Difference[x] = 1
ELSE Difference[x] = 0

The two pixel-spanning distance of the operator is designed to detect parallel edges such as those present along red-white border on signs and white-background borders [8]. The pixels in first two columns and last two columns of the resulting image are all set to 0 because they do not have pixels two columns to their lefts and rights, respectively. Figure 12 shows the XOR image, RGB differencing image, and image following the local maxima operator for the stop sign shown in Figure 7.

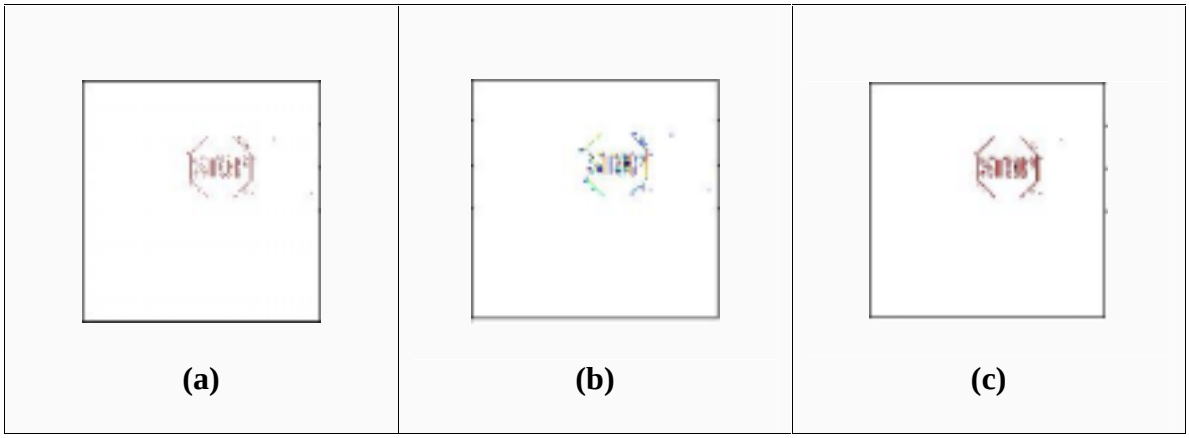


Figure 12: Detecting the edges in an image is a three-step process: (a) XOR, (b) RGB differencing, and (c) a local maxima operator.

6.3.5 PASS SEMI-RECTANGULAR MASK

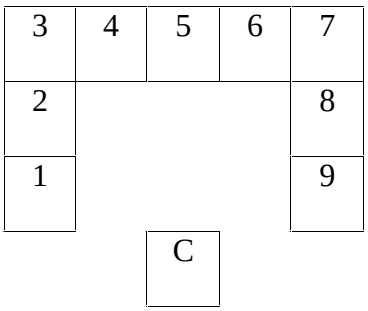
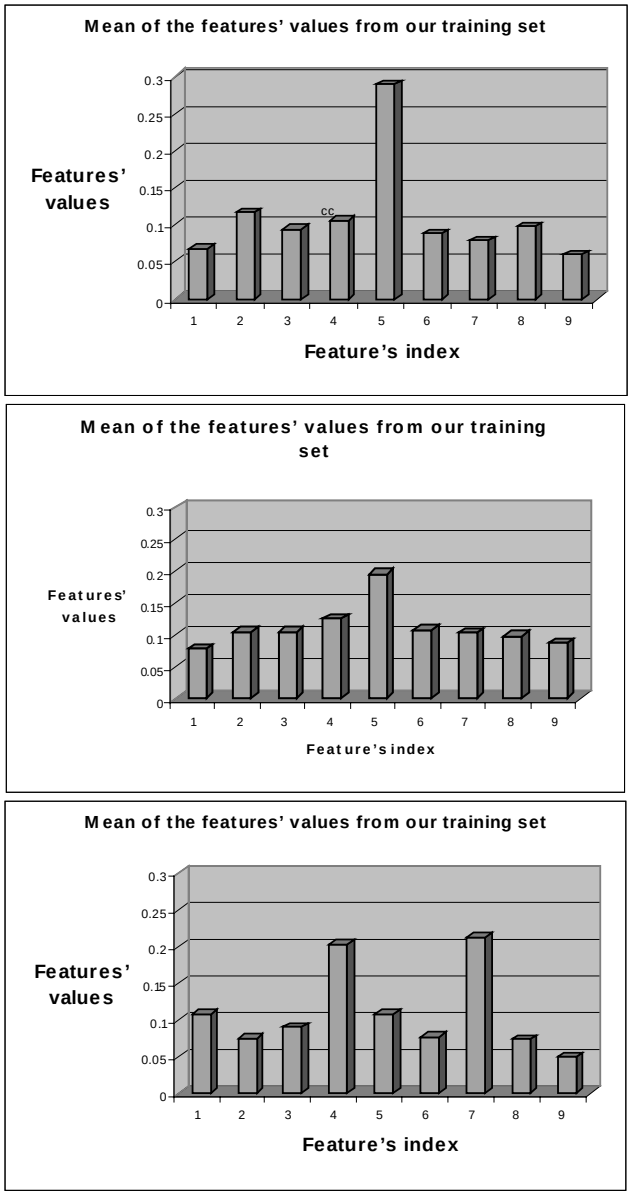


Figure 13: Shape of the semi-rectangular mask

The next step of the recognition stage is to pass a semi-rectangular mask over each labeled window, in order to update what will become the features for shape analysis. The shape of this mask is shown in Figure 13. When "C" is over a pixel that belongs to an edge, the program increments all the registers (1 to 9) that are also over a pixel corresponding to an edge. After this mask has been passed over all pixels inside a given window, the values contained in each of the nine registers is normalized, so that they sum to one. The final values contained in the registers are the features that the algorithm uses to decide whether a stop, a do not enter or a yield sign is contained in this window, or if this red area only contains background noise. These features can be interpreted as probabilities of certain angles:

- Register 5 detects vertical edges (90-degree angle).
- Registers 2 and 8 detect 45-degree angles.
- Registers 3 and 7 detect 60-degree angles.
- No register corresponds to horizontal edges. This is not a problem, since the XOR operation also ignores these edges.

Typical features' values are depicted in Figure 14.



(b) Do Not Enter sign

Figure 14: Typical features' values for a stop sign (a), do not enter sign (b) and yield sign (c)

6.3.6 DECISION BASED ON PSEUDO-MAHALANOBIS DISTANCE

Once the nine features had been calculated for a given red zone, a method based on the Mahalanobis distance was used to make the program decide whether the red zone contained a road sign and if it did, which one it was. Call 'x' the 9-component vector containing the features' values extracted from a red zone. Call 'm_{stop},' 'm_{dne},' and 'm_{yield}' the mean values vector of the features extracted from the training sets of stop, do not enter, and yield signs respectively. Call 'C_{stop},' 'C_{dne},' and 'C_{yield}' the covariance matrix of the features extracted from the training sets of stop, do not enter, and yield signs respectively.

The Mahalanobis distance of the red zone to the stop, do not enter, and yield sign categories are computed as follows:

$d_{stop} = (x- m_{stop})^T C^{-1} (x- m_{stop})$

$d_{dne} = (x- m_{dne})^T C^{-1} (x- m_{dne})$

$d_{yield} = (x- m_{yield})^T C^{-1} (x- m_{yield})$

Since there is little correlation between the values of the different features, and in order to avoid heavy matrix computation, the group neglected the cross correlation and only considered the diagonal of each of the covariance matrices. Eqn. 3 below describes how the group's "pseudo-Mahalanobis" distances were computed:

$$d_{stop} = \sum_{i=1}^{9} \left(\frac{x_i - m_{i,stop}}{\sigma_{i,stop}^2} \right)^2$$

$$d_{dne} = \sum_{i=1}^{9} \left(\frac{x_i - m_{i,dne}}{\sigma_{i,dne}^2} \right)^2$$

$$d_{yield} = \sum_{i=1}^{9} \left(\frac{x_i - m_{i,yield}}{\sigma_{i,yield}^2} \right)^2$$

(Eqn. 3)

where,

- $m_{i,stop}$ represents the mean of the i^{th} feature over our stop sign's training set.
- $m_{i,dne}$ represents the mean of the i^{th} feature over our do not enter sign's training set.
- $m_{i,yield}$ represents the mean of the i^{th} feature over our yield sign's training set.
- $\sigma^2_{i,stop}$ represents the variance of the i^{th} feature over our stop sign's training set.
- $\sigma^2_{i,dne}$ represents the variance of the i^{th} feature over our do not enter sign's training set.
- $\sigma^2_{i,yield}$ represents the variance of the i^{th} feature over our yield sign's training set.

The group decided to divide the distance to the mean by the variance in order to give more importance to features that were "reliable", or in other words, that had a small variance.

The group discovered that a simple minimum detector for these three distances was not very good, thus a multiplication step was added. Before selecting the minimum of these three distances, each of them was multiplied by an experimental weight. It was especially difficult to detect do not enter signs. The pseudo-Mahalanobis distance for a red zone containing a do not enter sign to the do not enter category was often bigger than its distance to the stop category. That is why it was decided to multiply the distance to the do not enter category by 0.27. It is the only weight that is introduced – the stop and yield weights were set to 1. A summary of this decision stage is depicted in Figure 15.

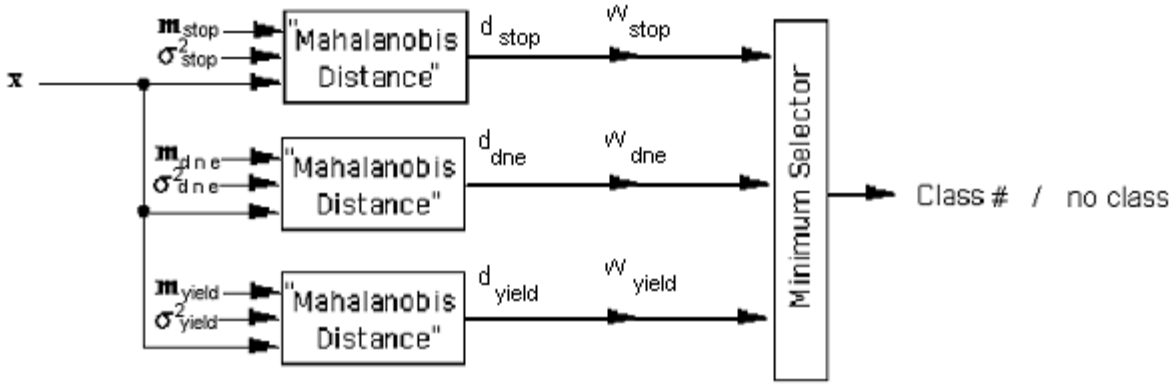


Figure 15: Diagram representing the decision stage

If the smallest distance was still bigger than a given threshold experimentally set at 26000, the algorithm interpreted the red area as “no road sign”. Otherwise, it interpreted it as the sign category having the smallest weighted pseudo-Mahalanobis distance.

7.0 RESULTS

Results for the number of signs correctly identified in the training set are shown in Table I. Table II shows the results for the test set images and Table III shows the results when the training and test sets are combined. The number of false alarms, which occur when a non-road sign red area (e.g. a red car, a red building, etc.) is recognized as a road sign, are also shown in the tables.

Table I: This table shows the results for the training set of images.

Sign	# Correct results	Total # signs	%	# False alarms	Total # red areas without a sign	%
Do Not Enter	8	11	72.73	1	2	50.00
Stop	17	23	73.91	1	8	12.50
Yield	7	7	100.00	4	5	80.00
Total	32	41	78.05	6	15	40.00

Table II: This table shows the results for the testing set of images.

Sign	# Correct results	Total # signs	%	# False alarms	Total # red areas without a sign	%
Do Not Enter	7	10	70.00	1	2	50.00
Stop	7	10	70.00	2	4	50.00
Yield	2	8	25.00	0	0	X
Total	16	28	57.14	3	6	50.00

Table III: This table shows the combined results for the training and testing sets of images.

Sign	# Correct results	Total # signs	%	# False alarms	Total # red areas without a sign	%
Do Not Enter	15	21	71.43	2	4	50.00
Stop	24	33	72.73	3	12	25.00
Yield	9	15	60.00	4	5	80.00
Total	48	69	69.57	9	21	42.86

Before presenting an interpretation of these results and how they could possibly be improved, it seems worthwhile to compare the results of this project with those obtained by [8]. [8] claims to correctly identify 50% of all stop signs, 37% of all yield signs, and 94% of all do not enter signs. As Table III shows, this project achieved better results for stop and do not enter signs. The do not enter results for this project were not as good as [8], but correctly identifying 71.43% is not too bad. The fact that this project's results, for the most part, are better than [8]'s is even more impressive when you consider the images used in the two projects. [8] never describes the images used in their project, but all of the images shown in their report are straight on shots of one sign with no other red objects in the image. The images in the project presented here contain red objects other than the sign, signs at various angles, and partially occluded signs. If this project had used ideal images, the algorithm may have produced even better results. It should be noted that this group's results are better than [8]'s results, so the additions that the 18-551 group made to their algorithm were indeed successful.

7.1 POSSIBLE ACCURACY IMPROVEMENTS

As described in section 7.0, the results for the testing set are not outstanding, especially for yield signs. There is a very simple explanation: half of the yield sign pictures that were used in our training set were rotated by an angle slightly bigger than the maximum in-plane rotation angle that the algorithm could handle, which is around ±10 degrees. An angle bigger than that causes the wrong register to be incremented when the semi-rectangular mask is passed over the red zone.

Since the edges of a yield sign are mostly at 60 degrees, features 3 and 7 should have the highest values among all the features for a yield sign. But as shown in Figure 14 (c), the peak actually occurs for features 4 and 7. This is what produces most of the misinterpretations of yield signs in the testing set; the constants (mean and variances) used to calculate the pseudo-Mahalanobis distances were calibrated using corrupted yield sign images. A solution to solve this problem would be to use a much bigger database, without any signs rotated more than ±10 degrees.

Another smart approach would be to analyze the results obtained for small signs (pictures taken far away from the sign) and with big signs (pictures taken close to the sign), to see if they differ significantly. It is possible that results are much better for a given size of a road sign than another. If this were the case, it would be worth using different constants (means and variances) for each size range. To pursue this idea further, a different decision technique could be used for each size range. Due to time constraints, the group was not able to test this approach, but the program is modular enough to allow future groups of students implementing this idea to do so. One could also replace the pseudo-Mahalanobis distance by more conventional distance calculations, such as nearest neighbor, k^{th} nearest neighbors, or the rigorous Mahalanobis distance.

Finally, several parameters in the algorithm can be further tuned to obtain better results. Such parameters include the following:

- Weights used to compute the pseudo-Mahalanobis distances
- Size and shape of the structural element used in the closing operation
- Minimum red area considered
- Threshold value to create the binary redness image (to take care of color bleeding effect)
- Determine a method for setting α in Eqn. 1

8.0 MEMORY/SPEED/OPTIMIZATION

Due to the amount of memory required for the program and the images, the group chose to store these in external memory. Some variables were stored in on-chip memory for quick access. Table IV shows how the group allocated the EVM's memory in this project.

Table IV: This table shows how the EVM's memory was allocated in this project.

What	Where	How Much
Code	SBSRAM_PROG	66.720 KB
Heap (malloc and calloc)	SDRAM0	1050.360 KB
Variables defined with far		
Global variables	ONCHIP_DATA	4.656 KB
Stack (for local variables)		
Variables defined with const		

To analyze the speed the program, the group ran it on eight images with the optimizer feature of Code Composer Studio disabled and it was found that the program took an average of 104 seconds to process one image. The optimizer was then enabled and the program was run on six images. When the optimizer was enabled, the program took an average of 42 seconds to process one image. The program runs significantly faster when the optimizer is enabled ($p = 0.04$) because it performs up to seven operations in parallel at some points in the program. Although this project is not close to real time, it is not too bad when one considers that some prior road sign recognition projects take up

to 150 seconds of processing time per image [9]. Several possibilities for improving the speed of this program are suggested in a later section.

8.1 POSSIBLE SPEED IMPROVEMENTS

Even though the speed of this algorithm is comparable with what other research groups achieved [9], it still is not running in real-time, which is the goal of this kind of program. The driver does not care that he passed a yield sign 200 feet before the program warns him of the sign! The group realized that the two slowest parts of the algorithm were the morphological filtering and the red zone labeling. It turns out that the algorithm used for the labeling is very simple and naïve, and that its speed can be significantly improved. Another 18-551 project group (group #2) this semester used another algorithm that took only 0.2 seconds to run on their image. The labeling used in this project took about 100 times longer.

Due to time constraints, the group focused its efforts on creating an algorithm that performed well. Therefore, the group did not devote much attention to optimizing the memory allocation. As an example, all of the data (images at different stages of the algorithm) were stored in external memory. It would be possible to use paging techniques similar to those employed in Lab 3, in order to transfer the data to on chip memory, and thus make the algorithm work faster.

9.0 ACKNOWLEDGEMENTS

The group would like to thank Professor David Casasent, Professor Tsuhan Chen, and the 18-551 Teaching Assistants, Matt Juhasz, Parag Patel, and David Wang for their help and advice. The group would also like to thank Cha Zhang for the 18-798 Image and Video Processing C code that was used to extract the RGB vectors from a ppm file.

10.0 REFERENCES

[1] American Driver & Traffic Safety Education Association. "Safety Statistics." http://adtsea.iup.edu/adtsea/resource_library/statistic_articles/safetystat.htm

[2] V. Libel. "The Road Sign Recognition System – RS^2." http://euler.fd.cvut.cz/research/rs2/rs2algorithm.html. 1996.

[3] U. Franke, S. Gorzig, F. Lindner, D. Mehren, F. Paetzold. "Steps Towards an Intelligent Vision System for Driver Assistance in Urban Traffic." *1998 IEEE*. pp 601 – 606. 1998.

[4] P. Paclik, J. Novovicova, P. Pudil, P.Somol. "Road Sign Classification Using the Laplace Kernal Classifier." *Institute of Information Theory and Automation*.

[5] S. Prince, R. Bergevin. "Road Sign Recognition Using Perpetual Grouping." http://euler.fd.cvut.cz/research/rs2/files/www.gel.ulaval.ca/Prince_S.html. 1998.

[6] S. Vitabile, G. Pollaccia, G. Pilato, F. Sorbello. "Road Signs Recognition Using a Dynamic Pixel Aggregation Technique in the HSV Color Space." *2001 IEEE*. pp 572 – 577. 2001.

[7] E. Perez, B. Javidi. "Composite Filter Bank for Road Sign Recognition." *2000 IEEE*. pp 754 – 755. 2000.

[8] L. Estevez, N. Kehtarnavaz. "A Real-Time Histographic Approach to Road Sign Recognition." *Image Analysis and Interpretation, 1996 Proceedings of the IEEE Southwest Symposium*. pp 95-100. 1996.

[9] M. Lalonde, Y. Li. "Road Sign Recognition." *Collection Scientifique et Technique*. Centre de Recherche Informatique de Montreal. pp. 1-29. 1995.