State University of New York at New Paltz
Department of Electrical and Computer Engineering

Senior Design II

EGE 409

Framework for Road Sign Detection in Images

Group Members	Department	Major Contribution
Haidar Khan	CE	All

Advisor: Dr. Faramarz Vaziri

12/18/2013

Abstract

Road signs are crucial to the safety of roads. Frequently, vehicle operators miss or misread important road signs. These errors by the vehicle operators can be harmless, but they can also easily cause accidents. In an effort to reduce the number of accidents caused by distracted driving, a road sign detection system is proposed. Using image processing and machine learning techniques, the road sign detector will be able to identify road signs at a distance and report important information about the road ahead to the vehicle operator. By augmenting the operator's knowledge of the road, the detector can reduce the frequency of accidents on the road due to distracted driving.

Table of Contents

Introduction 3
Subject 3
Objective 3
Motivation 4
Scope 4
Background 4
Theory – Image Processing 5
Theory – Machine Learning 6
Figure 1. SVM on Separable Data 8
Figure 2. SVM on Non-separable Data 8
Approach 9
Figure 3. System Diagram 9
Figure 4. Training and Detection Flowcharts 10
Phase I 10
Phase II 12
Figure 5. Sample Input Image 13
Figure 6. Input Image and Filter Response 15
Figure 7. Positive Blob with HOG Visualization 16
Figure 8. Negative Blob with HOG Visualization 16
Results 17
Hold-One-Out Cross-Validation 17
Table 1. HOOCV Results 18
Speed 18
Table 2. Speed Results 18
Accuracy 18
Figure 9. Detection Results 19
Graphical User Interface 19
Figure 10. Graphical User Interface 20
Future Work 20
Timetable 21
Table 3. Time Table 21
Engineering Constraints 21
Engineering Standards 22
Conclusion 22
References 24

Introduction

Vision is an essential human capability that is unparalleled by any technology in existence. Teaching computers to understand visual information like humans is a newly discovered problem. This type of research is part of the field of computer vision.

Subject

Computers understand visual information at a very low level. For computers, vision is simply the attributes of pixels that make up an image. On the other hand, humans and other animals can understand visual information at a much higher level. They can identify shapes, objects, trajectories, and actions. Giving computers these abilities would be highly advantageous, opening up a multitude of solutions to modern problems through computer automation. Examples of some problems that can be solved with computer vision are factory floor management, city traffic flow analysis, advanced surveillance systems, and automatic video annotation.

This project applies computer vision techniques to augmenting the knowledge of vehicle operators about the traffic regulations of the road and, if developed further, self-driving vehicles. A large portion of information about the road comes from road signs. Training a computer to understand road signs would relieve some of the burden on the vehicle operator.

Objective

Distracted driving is a malignant practice that is rampant on today's road ways. Calling, texting, being absorbed in conversation, and eating while driving are all examples of distracted driving. These types of actions while driving take the vehicle operator's attention away from the road, causing the driver to miss important information and even leading to accidents. Although

the fines and punishment associated with being caught driving inattentively have been increased, the problem remains because it is too widespread.

Introducing laws to prevent distracted driving is a good solution, but it is insufficient on its own. The solution to the problem of distracted driving needs to be multifaceted. This project proposes a solution that utilizes computer vision. The goal of this project is to implement a system which can be used to detect road signs in images and report this information to vehicle operators.

Motivation

A road sign detection system will help prevent accidents on the road due to distracted driving. As mentioned previously, distracted driving is a rampant problem across the globe. The United States Government has extensive campaigns raising awareness about distracted driving. Finding ways to alleviate this problem would prevent deaths and the destruction of property.

Scope

This paper will first cover necessary background information about the topic. Informed readers may skip some background sections without loss of continuity. Then, a detailed description of the approach will be given. Finally, a summary of results will be given along with suggestions for future work.

Background

Before discussing the approach taken for this project, necessary background will be covered. Computer vision relies on two main techniques, image processing and machine learning.

Theory – Image Processing

Image processing is a type of digital signal processing where the input signal is a digital image. Images are the fundamental data-structure in image processing, but they are composed of smaller elements called pixels.

Pixels are the building blocks of images. They are discrete samples of an original image, which may be continuous (real-world). A pixel has one property, color. The color value of a pixel can be expressed in many different formats.

The format the color of a pixel is expressed in is called a color-space. A binary color-space is one in which pixels take either the value 0 or 1. In the grayscale color-space, pixels are given a value based on their brightness. This value is typically eight bits in size. More advanced color-spaces include Red-Green-Blue (RGB) and Hue-Saturation-Value (HSV). Both of these color-spaces assign three values to each pixel. Many other color-spaces exist that express the color of a pixel in different ways.

Images are collections of pixels arranged in a matrix. Each pixel is assigned a location in an image given by its row number and column number. All the pixels in a single image are part of the same color-space.

Videos are collections of images and are a type of temporal data. Videos establish a sense of time by defining a sequence in which to show images, or frames. The speed of the video is expressed as the number of frames shown in a second (FPS).

Color-space conversions are used to express an image in a different color-space. This can be done using simple mathematical operations on the existing color-space values. It is possible to convert between most color-spaces, but if a color-space requires more information than exists in another color-space then conversion becomes impossible. An example of this is attempting to

convert from grayscale to RGB. This conversion will never result in a color image because the grayscale color-space does not contain that information. The result of this conversion will be a black and white (with shades of gray) image expressed in the RGB color-space.

Image filtering is a technique widely used in image processing. This process simply involves taking as input an image, applying a filter to it, and recording the response to the filter. The filter can be of any type, from simple linear ones to more advanced filters. The response to the filter also varies based on the filter type. This project will use a simple linear filter, with the output being a binary image.

Image segmentation involves separating an input image into regions. Segmentation can take many forms. It can be done based on distance, color, intensity, or combinations of these. Details about the type of segmentation used in this project will be discussed in the approach section.

Feature extraction is another important image processing technique. This involves extracting important information about an image and is used to reduce the dimensionality of image data. Many approaches exist in feature extraction; the method employed in this project is Histogram of Oriented Gradients [2]. This will also be discussed in the approach section of this report.

Theory – Machine Learning

Machine learning is a branch of artificial intelligence that studies systems that use data to make predictions. These systems are trained on a set of data, called training data, and then are called upon to classify new data of the same type. Many types of machine learning algorithms exist. Two main types of machine learning algorithms are unsupervised learning algorithms and supervised learning algorithms. Unsupervised learning algorithms are algorithms in which the

training data is not labelled. This type of algorithm looks for structure (i.e. clusters) in the data. Supervised learning algorithms use labelled data to teach the computer what the desired output is for a given type of input. The algorithm is then asked to predict the outputs of unseen input data with the assumption that it will be able to generalize to the unseen data.

This project uses a supervised learning algorithm developed by Vladmir Vapnik at Bell Laboratories called Support Vector Machines (SVM). [1] SVM is a binary classifier, meaning it classifies data into two classes. The input data to a SVM classifier is data labelled as two classes, positive and negative. In the training stage, the SVM algorithm finds the optimal separating hyper-plane between the two sets of data. The algorithm then uses the hyper-plane to classify new data by finding which side of the hyper-plane the new data exists on. SVM can be used to find the separating hyper-plane between separable datasets and non-separable ones. To build useful intuition on what SVM does, Figure 1 and Figure 2 show the hyper-plane between separable and non-separable data.

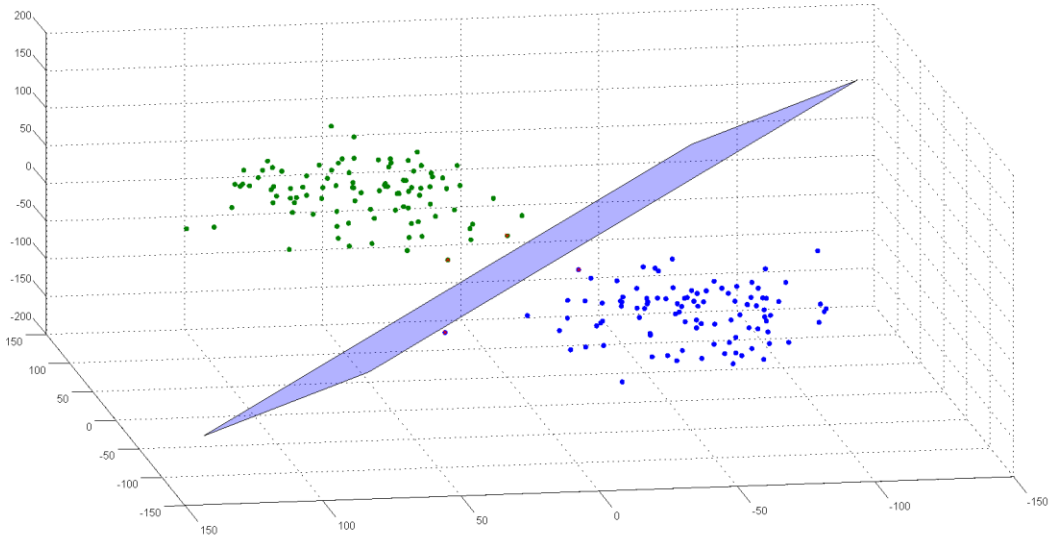

Figure 1. SVM on Separable Data

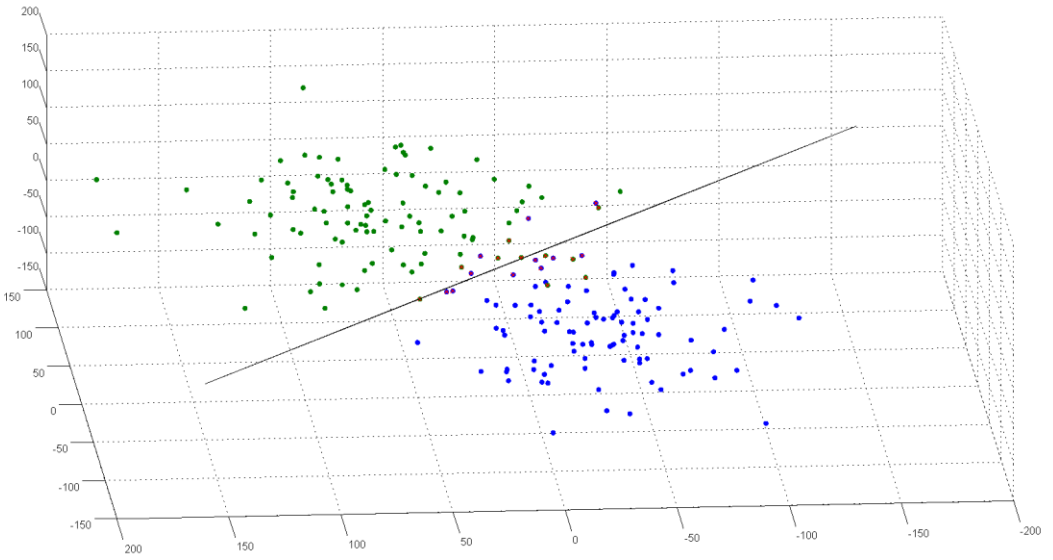

Figure 2. SVM on Non-separable Data

What sets Support Vector Machines apart from other supervised machine learning algorithms is the relatively low amount of computation required to classify new data. Most other supervised machine learning algorithms use the entire training dataset to classify new data. However SVM only requires select training vectors, called support vectors, to classify new data.

Support vectors are those training vectors that are closest to the separating hyper-plane. The advantage associated with using SVM is that only a select number of training vectors will become support vectors. An in-depth discussion of the SVM algorithm will be included in the approach section of this report.

Approach

The detection system developed here will take as input an image and some user defined parameters. The output of the system will be the locations on the input image of any detected road signs. The system diagram for this project is shown in Figure 3.

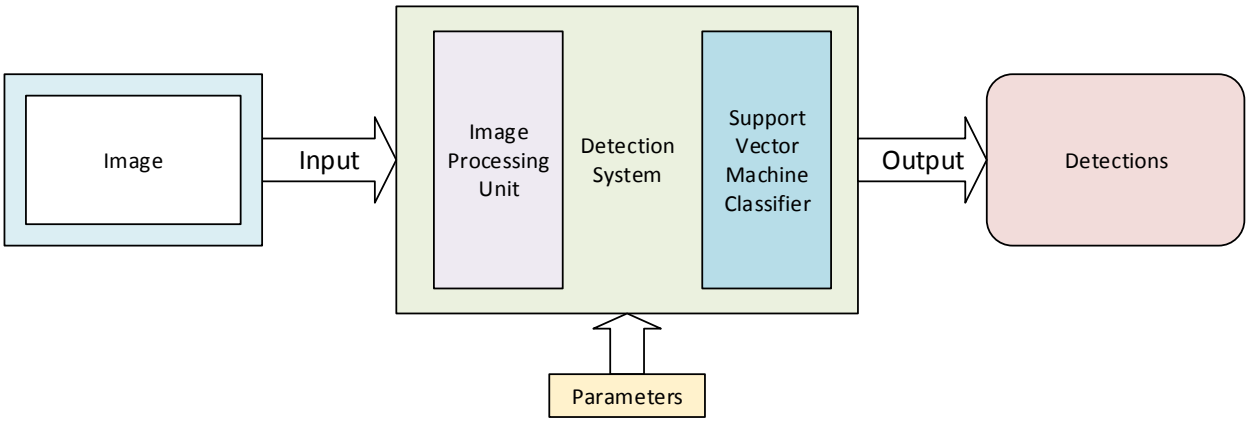

Figure 3. System Diagram

The detection system developed in this project involves two phases, as detailed in Figure 4. The first phase of the system involves training a model for a road sign. This will be done by creating and labelling a training dataset of images. This dataset is fed into the SVM algorithm and a classifier is learned from the data. The second phase of the system will attempt to detect road signs on images from a video stream. This will be done by filtering the images by color,

segmenting the resulting binary image into blobs, and classifying the blobs using the learned classifier.

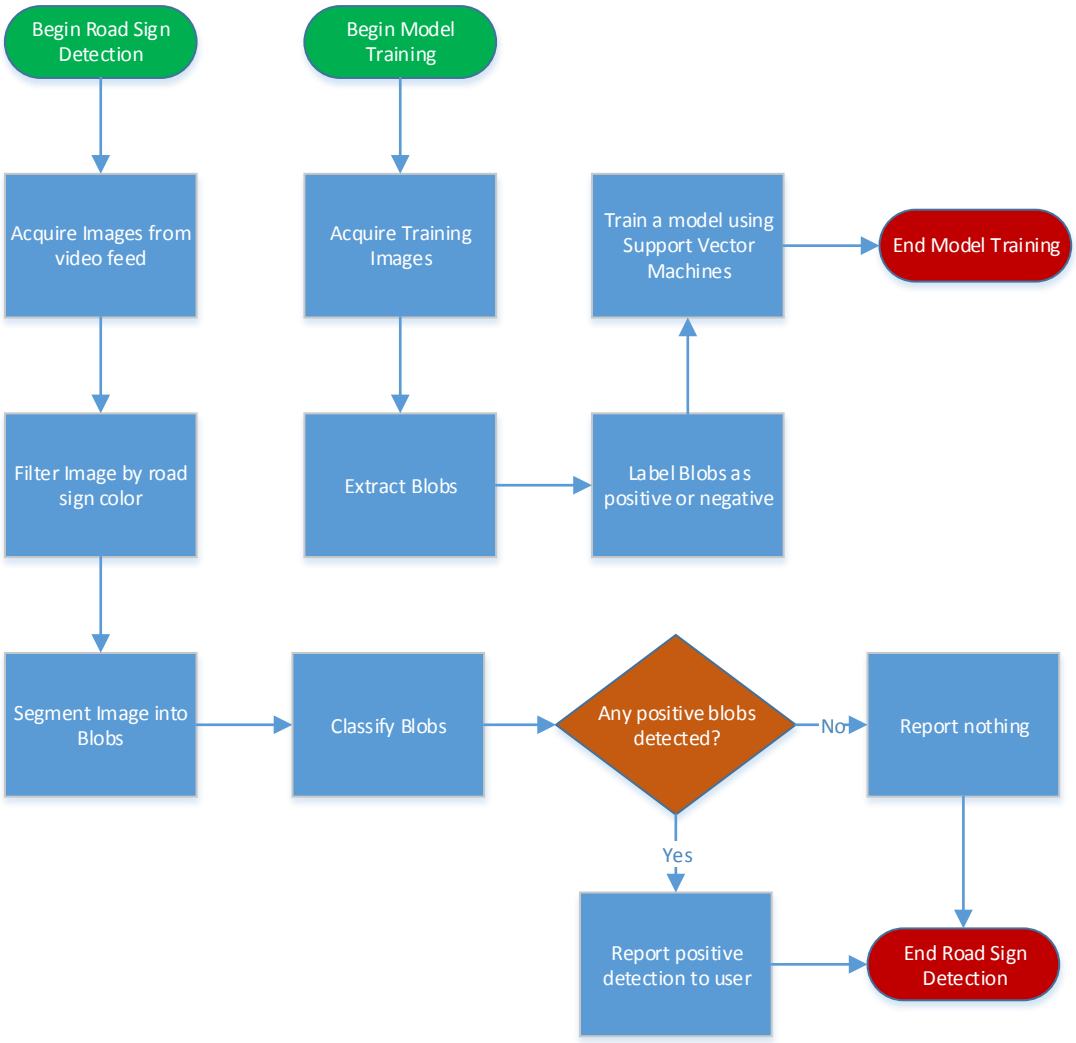

Figure 4. Training and Detection Flowcharts

Phase I

Before road signs can be detected in images, the computer needs to be taught what road signs look like. This is done by giving the computer images of road signs and labelling them as such. The computer is also given images of objects that look similar to the road sign but are not. These images are labelled as negative examples.

Using machine learning to teach the computer how certain objects look is both simple and elegant. It is simple because the programmer does not need to look for features such as shape, size, texture, and color that distinguish the road sign. The elegance comes from allowing the computer to decide on its own what to look for when it tries to classify input images into road signs and not road signs.

The SVM algorithm requires two sets of data as input. The first is the training matrix $\boldsymbol{X}$ composed of n feature vectors. Each feature vector $\boldsymbol{x}$ is a row vector with m features $x_1 \ldots x_m$. The second input data is the column vector of training labels $\boldsymbol{Y}$. $\boldsymbol{Y}$ has n elements, each from the set $\{-1,1\}$. The optimal hyper-plane that separates the two classes in $\boldsymbol{X}$ is found using the primal form of the SVM algorithm:

$$arg\ min_{(w,b)} \frac{1}{2}||\boldsymbol{w}||^2$$

$$\text{subject to: } \boldsymbol{Y}_i\left(\boldsymbol{w}^T\boldsymbol{X}_i - b\right) \geq 1,\ i = 1 \ldots n$$

The dual form of this optimization problem can be found using Lagrangian duality. The Lagrangian introduces α terms and expresses the optimization as:

$$max_\alpha \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j\, \boldsymbol{Y}_i\, \boldsymbol{Y}_j\boldsymbol{X}_i^T\boldsymbol{X}_j$$

$$\text{subject to: } \alpha_i \geq 0, i = 1 \ldots n$$

$$\text{and } \textstyle\sum_{i=1}^{n} \alpha_i\boldsymbol{Y}_i = 0$$

The output of both forms of the optimization problems is $\boldsymbol{w}$, the normal vector to the hyper-plane, and b, the bias value. For the dual form, $\boldsymbol{w}$ and b can be computed using:

$$\boldsymbol{w} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{\alpha_i Y_i X_i}$$

$$b = \boldsymbol{w}^T\boldsymbol{X} - \boldsymbol{Y_i}$$

The power of the SVM algorithm comes from a property of the α terms. α will be greater than zero only for those feature vectors that are closest to the separating hyper-plane, called support vectors. This will be only a fraction of the entire training dataset. This is useful because only support vectors will be needed in order to classify a new data vector $\boldsymbol{x}_{new}$. The formula used to predict the label $y_{new} \epsilon \{-1, 1\}$ of x_{new} is given below:

$$y_{new} = sign\left(\sum_{i=1}^{n} \alpha_i \boldsymbol{Y}_i \boldsymbol{X}_i^T \boldsymbol{x}_{new} + b\right)$$

Since α will equal zero except for the support vectors, the summation will be sparse.

The model trained using the SVM algorithm can be used to classify images as road signs and not road signs.

Phase II

This phase of the system requires extracting blobs, which are candidates for road signs, from an input image. This will be done using a combination of image processing techniques. An example of an input image is shown in Figure 5.

Figure 5. Sample Input Image

The first step in this process is filtering the input image by the color of the road sign. The input image is in the Red-Green-Blue (RGB) color-space, which is a good format for displaying on a screen but does not match with the way humans see color. Humans do not see a particular color as the combination of its red parts, blue parts, and green parts. Instead, colors are seen as distinctive with many shades. For example, humans will not see the color violet as a mixture of red and blue but instead they will label it as dark purple. Here purple is the color they see and dark is a qualifier quantifying the shade of the color.

A color-space that closely matches with the way humans see color is the Hue-Saturation-Value (HSV) color-space. In this color-space, the Value determines how bright or dark a color is. The Hue and Saturation determine the actual color. This is useful for filtering a particular color out of an image because the Value can be fluctuated from 0% to 100% to allow for differing shades while the Hue and Saturation determine the color.

Converting from RGB space to HSV space can be done using simple mathematical operations on the RGB values.

$$[R', G', B'] = \frac{[R, G, B]}{255}$$

$$\Delta = \max(R', G', B') - \min(R', G', B')$$

$$H = \begin{cases} 60° * \left(\frac{G' - B'}{\Delta} mod 6\right), \max(R', G', B') = R' \\ 60° * \left(\frac{B' - R'}{\Delta} + 2\right), \max(R', G', B') = G' \\ 60° * \left(\frac{R' - G'}{\Delta} + 4\right), \max(R', G', B') = B' \end{cases}$$

$$S = \begin{cases} 0, \ \Delta = 0 \\ \frac{\Delta}{\max(R', G', B')}, \ o.w. \end{cases}$$

$$V = \max(R', G', B')$$

The linear filter used to filter a specific color from an image consists of six parameters. These parameters define the acceptable ranges for Hue, Saturation, and Value. The following functions are used to determine the value of a pixel $\boldsymbol{p_{out}}$ in the output binary image, given $\boldsymbol{p_{in}} = [h, s, v]$ and parameters $hLow, hHigh, sLow, sHigh, vLow$, and $vHigh$:

$$p_{out} = \begin{cases} 1 \ if \ hLow \leq h \leq hHigh \cap \ sLow \leq s \leq sHigh \cap \ vLow \leq v \leq vHigh \\ 0 \ otherwise \end{cases}$$

Figure 6 shows an example of an input image and its response to the filter.

Figure 6. Input Image and Filter Response

The next step in extracting blobs from an image is segmenting the binary image that results from the filter. This is done using an agglomerative segmentation algorithm in which the pixels of interest (pixels that have a value of 1) are separated into classes that grow to include pixels that are in close proximity. The segmentation algorithm takes as input an array of n pixels and a distance threshold parameter d. Each pixel of interest is initialized as its own class. Then the distance between every pair of pixels is calculated. Pixels that have a distance between each other less than the threshold distance are labelled under the same class label. This procedure is outlined in the following pseudo-code:

```
//Input: array of n pixels (p = [x, y]), threshold d
//Output: array Class[1…n]- class label for each pixel
//Initialize: Each pixel is its own class
class <= 1:n
for i <= 1 to n
      for j <= i+1 to n
            if( distance(pi, pj) < d)
                  class[j] = class[i]
return class
```

After the segmentation algorithm is run, pixels that belong to the same class form blobs. Each blob is a candidate for the location of a road sign. Before the blobs can be classified using the classifier learned in Phase I, the important features of each blob must be extracted.

Feature extraction is done to prepare data for classification by reducing the dimensionality and removing redundant data. The feature extraction method used here will be Histograms of Oriented Gradients (HOG), developed by Dalal and Triggs. This method works by breaking an input image into cells and computing a histogram of values for each cell. The method computes 31 values for each cell, representing the gradients in different directions inside the cell. More details on how the method works can be found in [2]. Figure 7 and Figure 8 show a positive blob example with features extracted and negative blob example with features extracted respectively.

Figure 7. Positive Blob with HOG Visualization

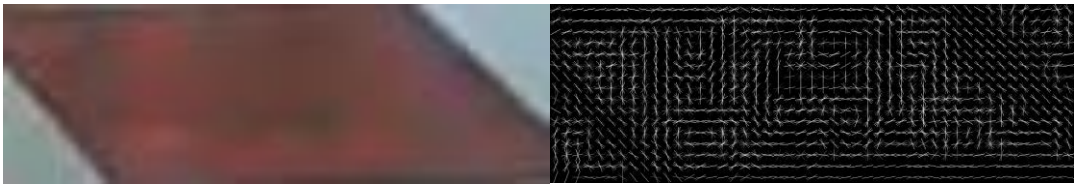

Figure 8. Negative Blob with HOG Visualization

The output of the HOG method is a feature vector for each blob that can be fed into the classifier learned in Phase I. The classifier will label each blob as either a road sign or not a road

sign. Phase II of the system is completed when the system returns the location of any blob classified as a road sign.

Results

Three different models were trained using the approach described above. Detectors were trained for stop signs, speed-limit signs, and intersection-ahead signs.

Hold-One-Out Cross-Validation

The first form of results generated was a measure of how general each model was. This was done using the well-known method called Hold-One-Out Cross-Validation (HOOCV). This method measures how well a trained model will be able to classify unseen data. This is important to measure because the most common problems with machine learned classifiers are that they over fit the training data. This causes them to be unreliable when classifying new data.

HOOCV works by holding out training vectors, training on the remaining data vectors and then attempting to classify the held out vectors. For a training dataset of size n, HOOCV will run n iterations. The method is best described in the following pseudo-code:

```
//Input: Training matrix of n rows and m columns X
//          Label vector Y
//Output: percentage labelled correctly
correct = 0
For k = 1 to n
      T = X       //save original training matrix
      tY = Y      //save original training labels
      hold = X[k,:]
      holdLabel = Y[b]
      T[k,:] = []       //remove held vector from training data
      tY[k] = []        //remove held vector label
      S = trainSVM(T, tY)     //train model on remaining data
      if(classifySVM(s, hold) == holdLabel)     //check if labels match
            correct++
return correct/n*100
```

Table 1 shows the results of HOOCV for each model trained.

Table 1. HOOCV Results

Model	HOOCV Result
Stop Sign	98.83%
Speed-limit Sign	86.45%
Intersection-ahead Sign	90.86%

Speed

Since this design will eventually be run real-time on video streams, speed is another measurement that must be obtained. I used MATLAB's speed measurement tools to measure how fast the algorithm ran. I did this by averaging the time it took to call the detection function multiple times. Table 2 shows the results obtained for images with a resolution of 1280-by-720.

Table 2. Speed Results

Number of function calls	Total time in function	Average time per call	Estimated FPS
858 calls	300.004s	.249s per call	~2.9 FPS

Accuracy

The last aspect of the models that needs to be measured is how well the models are able to handle adverse conditions. This is the most difficult aspect to measure since it is only possible to evaluate a select number of conditions. Some of the conditions tested are shown in Figure 9.

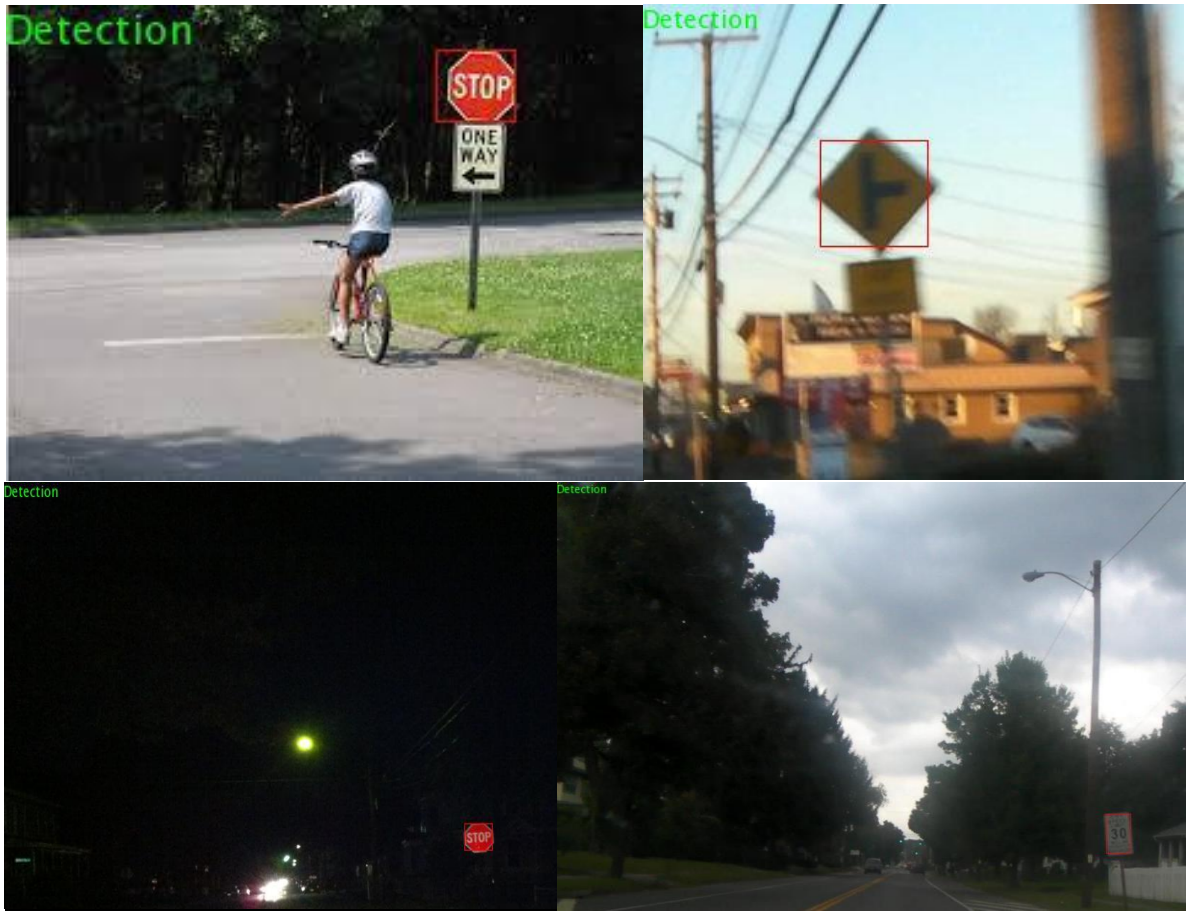

Figure 9. Detection Results

Top-left: stop sign in daylight. Top-right: Intersection ahead sign at dusk. Bottom-left: Stop sign at night. Bottom-right: Speed-limit sign at dawn.

Graphical User Interface

To stream-line the process of training a model for a road sign, a MATLAB Graphical User Interface (GUI) was developed. The GUI allows the user to create a model for detecting a particular road sign. The user supplies training images that show the road sign of interest. Then the user is required to configure a number of parameters such as the filter ranges, distance threshold, blob size, and cell size. After all the parameters have been entered, blobs are extracted from the training images and labelled. Finally, the software trains a model on the training data and returns the HOOCV for the model. Figure 10 shows two screenshots of the GUI.

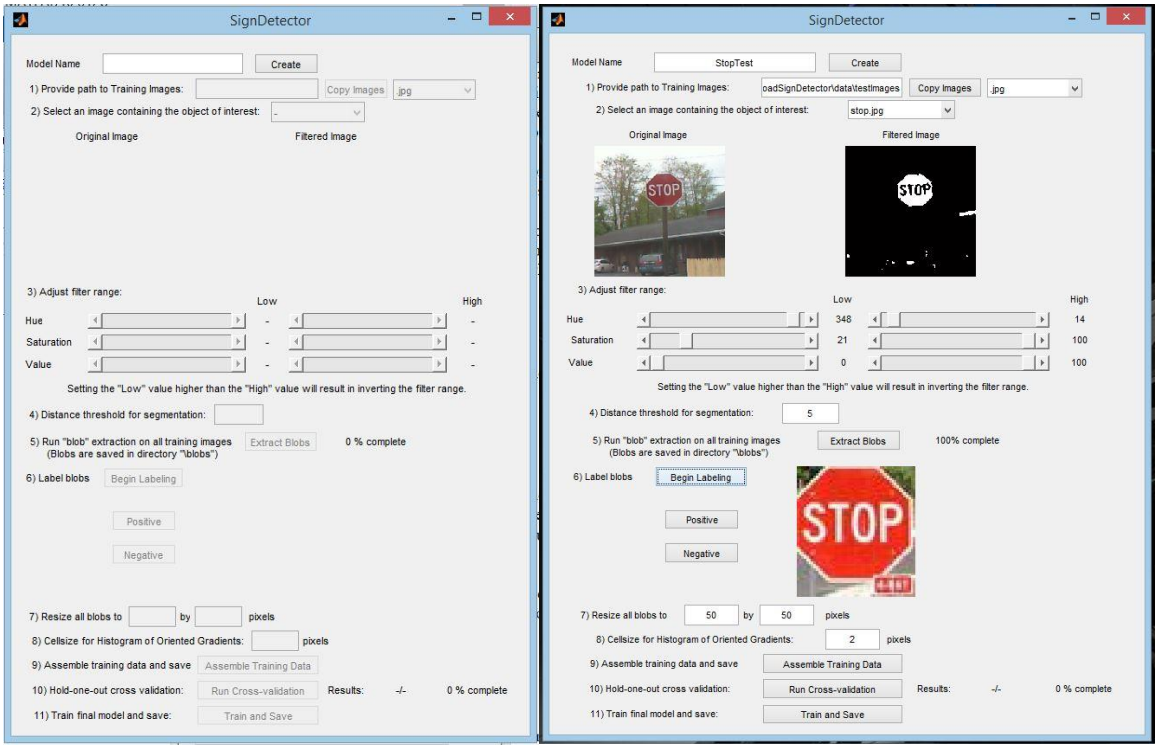

Figure 10. Graphical User Interface
Left: GUI at startup. Right: GUI being used to develop a detector model

Future Work

In order to implement this framework in an embedded setting, the MATLAB code must be converted to another language. The C++ language is an ideal language to convert to. The Open-Source Computer Vision (OpenCV) libraries can be used to handle the image processing functions required to implement the framework. Converting the MATLAB code to C++ will also result in a speed-up because compiled languages (C++) are generally faster than interpreted ones (MATLAB). Conversion from MATLAB to C++ has begun but is not finished as of now.

Another area for future work is improving the image filtering methods to handle white road signs. In the tests conducted, it was found that it is difficult to extract white color from images using the methods above. Another method to filter images by color would improve the

ability to detect white road signs. One way to do this would be to have the computer automatically select filter ranges for each color.

The feature extraction method used in this approach is Histograms of Oriented Gradients (HOG). While this feature extraction method works, it seems that a simpler feature extraction method would work more effectively. A method like Canny Edge Detection, which extracts edges from an image, is simpler than HOG because the output is of lower dimension. Features of lower dimensions would cause a speed-up in runtime without affecting accuracy.

Timetable

Table 3 shows the breakdown of tasks and when they were completed.

Table 3. Time Table

January February March April	**May**	**June**	**July August**	**September**	**October November**	**December**
Researched possible topics, finally chose road sign detector.	Senior design I poster board presentation.	Began research on approach and development	Began design and development. Reported preliminary results to advisor	Continued into second phase of development (Machine Learning algorithms implemented)	Finished MATLAB GUI, began developing OpenCV module.	Senior Design 2 Presentation and Write-up.

Engineering Constraints

A number of engineering constraints had to be considered while developing this project. These include economic, manufacturability, social, sustainability, and safety constraints.

The economic constraints this project has to deal with is that if the product is used it will reduce the number of accidents on the road. This will invariably have an effect on insurance companies and insurance policies.

The manufacturability constraints deal with how easy the product is to manufacture. This product can easily be run on a Raspberry Pi embedded system ($35) and a standard webcam (~$20). This is relatively cheap, totaling to about $55.

The social and political constraint faced by this project is that new laws will need to be developed for dealing with these devices. It might even be possible to pass laws to require them in all vehicles.

This project is sustainable because it will not become obsolete until road signs are not used to convey information about the road. Due to the extensive road sign infrastructure on the roadways today, it does not appear as though this technology will become obsolete in the near future.

Lastly, this project deals with the health and safety constraint because its goal is to increase safety on the road. The motivation of the project is that if vehicle operators are better informed about the road ahead, it will reduce the number of accidents that happen.

Engineering Standards

This project used engineering standards related to evaluating machine learning classifiers. This involved using HOOCV for measuring how well a classifier generalizes.

IEEE standards were also used for citing papers, many of which were published in IEEE Explore.

Conclusion

This project proposed a framework for detecting road signs in images. The goal of this project was to detect road signs in images so that vehicle operators have easier access to

information about the road. The motivation behind this project comes from the problem of distracted driving, where vehicle operators miss important information about the road. The framework developed used a combination of image processing and machine learning techniques. Image processing was used to isolate locations for possible road signs on an input image. Machine learned classifiers were then used to determine which of these possible locations actually represented a road sign.

This project is useful because it allows computers to understand how existing road signs appear. Once this framework is implemented in an embedded setting and placed within a vehicle, it will be able to reduce the number of accidents on the road due to distracted driving. Extending this framework can aid in the development of self-driving vehicles by giving computers a better understanding of the road.

References

[1] Chapelle, O.; Haffner, P.; Vapnik, V.N., "Support vector machines for histogram-based image classification," *Neural Networks, IEEE Transactions on* , vol.10, no.5, pp.1055,1064, Sep 1999
doi: 10.1109/72.788646
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=788646&isnumber=17091

[2] Dalal, N.; Triggs, B., "Histograms of oriented gradients for human detection," *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* , vol.1, no., pp.886,893 vol. 1, 25-25 June 2005
doi: 10.1109/CVPR.2005.177
http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1467360&isnumber=31472