Traffic Sign Detection and Recognition System for Intelligent Vehicles

by

Jingwen Feng

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Road traffic signs provide instructions, warning information, to regulate driver behavior. In addition, these signs provide a reliable guarantee for safe and convenient driving. The Traffic Sign Detection and Recognition (TSDR) system is one of the primary applications for Advanced Driver Assistance Systems (ADAS). TSDR has obtained a great deal of attention over the recent years. But, it is still a challenging field of image processing.

In this thesis, we first created our own dataset for North American Traffic Signs, which is still being updated. We then decided to choose Histogram Orientation Gradients (HOG) and Support Vector Machines (SVMs) to build our system after comparing them with some other techniques. For better results, we tested different HOG parameters to find the best combination. After this, we developed a TSDR system using HOG, SVM and our new color information extraction algorithm. To reduce time-consumption, we used the Maximally Stable Extremal Region (MSER) to replace the HOG and SVM detection stage. In addition, we developed a new approach based on Global Positioning System (GPS) information other than image processing. At last, we tested these three systems; the results show that all of them can recognize traffic signs with a good accuracy rate. The MSER based system is faster than the one using only HOG and SVM; and, the GPS based system is even faster than the MSER based system.

Acknowledgements

First of all, I would like to offer my sincere gratitude to my supervisor, Professor Azzedine Boukerche, who has provided me with financial support during my studies. Professor Boukerche has supported me with his kind heart, his encouragement and his patience; in addition to providing a nice environment for my research. He has always been kind and has also been a friend. I greatly appreciate his kindness.

And secondly, I would like to thank the Dr. Abdelhamid Mammeri who accepted me as a member of the Mobile Vision Group in PARADISE laboratory. I have been extremely lucky to be aided by him as a tutor. He always provided useful information and orientation to me regarding the image processing field. And during my entire research, he always gave me support and the confidence that I needed to figure out all the tasks I have met.

Thirdly, I would like to thank all the PARADISE researchers, especially my colleagues in the Mobile Vision Group. We always discuss amongst ourselves inside the group; and, thus we make the process of researching more fluent. My colleagues have provided so much inspiration, knowledge, help, and friendship during these two years. And, we have all shared this experience in a harmonious environment.

Finally, I thank my parents; their support and understanding have made me a better person.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background and Motivation

As a product of social development and technological progress, the motor vehicle has contributed tremendously to the progress of human civilization and social development. However, motor vehicles also lead to a lot of problems, such as potential traffic safety hazards, traffic congestion and environmental pollution. How to effectively improve the safety and efficiency of transportation becomes a problem for both governments and research institutions.

To solve this problem, many researchers have made a lot of effort to trigger people interest in autopilot technology. Throughout the continuous research process, many new technologies have been applied to the automobile industry. From the beginning of the 1980s, some developed countries like Japan, the United States, Germany, etc., have carried out intelligent transportation systems.

Intelligent vehicle technology helps reduce driver fatigue and to improve comfort during driving. Intelligent vehicle technology can be divided into two aspects: assisted driving and autopilot. Autopilot technology relies on the establishment and on the improvements of the application of the Intelligent Transportation System (ITS). And autopilot technology is difficult to achieve over a short time. Therefore, the United States, Japan and other developed countries sought to instead develop Advanced Driver Assistance Systems (ADAS). As early as 1987, Japan researchers H. Akatsuka et al. began the work on automatic traffic

sign detection and recognition [6]. The majority of research [7], [8].

The road Traffic Sign Detection and Recognition (TSDR) system is a smart system designed to help drivers react properly in dangerous traffic situations. It is one of the unresolved problems in ITS research areas. Moreover, it is one of the real difficult problems for pattern recognition. In automotive vision systems, how to effectively identify traffic signs is a very important research concern. As an important subsystem of ADAS, TSDR system is bound to obtain a wider range of applications. Furthermore, it is one of the essential components of future automotive driving and the semi-automatic driving system. In addition, in the electronic traffic maps, automatically identifying traffic signs will provide necessary information for the position of the vehicle. Therefore, research of TSDR is extremely important for both practical significance and application value and it has become that much more necessary.

In our research, a system that prevent accidents was designed. It is built directly onto cars, and interprets traffic signs for drivers. If all drivers had access to this, the number of accidents would decrease and traffic would be more fluent. The TSDR system picks up live images or video streams from one or more in-vehicle cameras, analyze them and provide user-friendly and recognizable traffic signs to the driver.

1.2 Thesis Outline

In Chapter 1, we explain why we built the TSDR system. In Chapter 2, we discuss the related work in the context of different techniques already used in this field. In Chapter 3, we create a dataset that will be updated daily with all the North American traffic signs, especially with those found in Canada. We do this because there is a variety of different kinds of traffic signs in the world and no one single open-sourced dataset that contains all of them. We then explain how we used the dataset we created to train and test our final system.

We then present the most important chapter in this thesis, Chapter 4. In this Chapter, we discuss the elements described in the following three paragraphs.

1. We discuss the algorithm details for the detection and classification stages we used in this thesis. As the feature descriptor, we introduce the classic descriptor of Histogram

of Oriented Gradients (HOG). We then discuss two classification algorithms that will be presented separately: Support Vector Machine (SVM) and tree classifiers. These algorithms, with different setting characteristics, will help us distinguish the positive from the negative samples.

2. We discuss how to train and build the weight vector. The training of the HOG feature is based on the HOG parameter testing from our own dataset, which will result in better performance.

3. The construction design needed to suit the different purposes of system, including the separation of the background color and time saved will be discussed. We will also test the different videos under different weather conditions and road conditions, as well as at different times of the day, to see how they would work.

In Chapter 5, we provide a new concept for solving the same problem without cameras. Instead, we suggest using navigational information and the algorithm for calculating the distance between two coordinates and for simulating part of the client side component based on this algorithm. Finally, Chapter 6 concludes this thesis and future plans are discussed.

Chapter 2

Related Work

Over the past twenty years, French researcher De Saint Blancard M. developed red sign recognition systems [9], in 1992, these developed systems used a red filter, edge detection method, a closed curve detection method, and a neural network recognition method to complete the extraction process and target classification. The American researchers N. Kehtarnavaz et al. adopted a color clustering method in 1993 to discriminate against stop signs [8]. But, this approach did not achieve real-time identification. In 1994, German scientists S. Estable et al. launched the world's most advanced TSDR system [10] at that time. The system used color segmentation, shape analysis and statistical pattern recognition techniques. In the same year, Italian researchers G. Piccioli et al. proposed an extraction method using image geometry, namely linear detection, triangle detection, circle detection and so forth [11].

However, although the research on TSDR system has achieved a few practical results with its evolution over the past decades, there are still some shortcomings. According to the research in this area and to the characteristics of signs, most traffic detection and recognition algorithms divide the problem facing TSDR system into the following elements:

1. Environmental conditions and road conditions: the environmental conditions always vary according to lighting, illumination and weather. As uncontrollable as the above-mentioned factors are, environmental conditions which affect the results of TSDR system. Additionally, the detection quality will be lower. The road conditions will influence TSDR system in other aspects. Because the cameras are always installed

in the vehicle, holes on the roads, or the muddy conditions of roads, will cause vibrations and oscillations, which will affect the video quality captured by cameras. Road conditions are a tough issue for moving vehicles as well.

2. Region determination: objects that have the same characteristics as traffic signs may present difficulties for systems when determining the location of the signs; the barriers on the road like advertising boards will result in incomplete target region determination as well. Besides, the states of many traffic signs are also degraded.

3. Candidate classification: this problem happens when identifying the content of extracted regions. Mis-recognition may happen when some signs have the same patterns but use different texts (speed limit signs). Few techniques exist here; each technique needs to be combined with the detection stage so that there are many restrictions in terms of choice. In addition, few techniques can classify the entire content or all the information of signs clearly.

Because of the problems we obtained, TSDR system is always studied in three basic stages:

- Pre-processing stage
- Detection stage
- Recognition stage

According to the above-mentioned stages, we determined the following objectives for future research and development in this field: simplify complex problems, improve the traditional method, and use a smart approach.

2.1 Pre-processing Techniques

The processing step is not a mandatory one. But, this step may increase the efficiency of the following steps. The fundamental function of the pre-processing stage is to enhance the input image and to avoid the dense scanning of it.

2.1.1 Camera calibration

Camera calibration is a step which finds the extrinsic and intrinsic parameters for a camera set on vehicles [12]. In the process of image measurement and machine vision applications, we need to establish the imaging geometry model on cameras. This model helps us to determine the corresponding relationship between points in the image and the 3-dimensional geometric position of respective that points in the surface space of the object. Under most situations, these parameters are obtained by experiments and calculations. The process of obtaining these parameters is called camera calibration.

Regardless of the image measurement or the machine vision applications, camera calibration is a very critical aspect. The accuracy and stability of the calibration algorithm will affect the calibration accuracy directly. Therefore, good camera calibration is a valuable premise for follow-up work; and, exactly how to improve the calibration accuracy should be focused on by scientific researchers [13]. So far, there are three main methods for camera calibration: traditional camera calibration, camera auto-calibration and camera self-calibration based on active vision [14].

Traditional camera calibration requires an object of a known size to calibrate. This is done through an established relationship between the known coordinate points of the object to be calibrated and its image points. Additionally, a certain algorithm is needed to obtain the internal and external parameters of the camera model [15]. The objects that will undergo calibration can be divided into two kinds: the 3-dimensional object and the plane type of object [16]. A 3-dimensional object can be calibrated under a single image; this has a high calibration precision. But, high-precision calibration for 3-dimensional object is difficult to process and to maintain [17]. However, a plane type of calibration for a respective object is easy to process and the accuracy can be ensured. But, during calibration, this type of calibration needs to use two or more images which is different from the calibration of a 3-dimensional object. The disadvantages of this method are that the object will affect the accuracy of the calibration results. Meanwhile, some scenarios which are not suitable for the calibration of objects, limit the application of traditional camera calibration.

The auto-calibration algorithm [18] restricts camera movement. The main constraint is that some parallel or orthogonal information in a scene must be used [19]. That is, the

spatial intersection of parallel lines in the camera image plane called the vanishing point is used; this is a very important feature in projective geometry. Many researchers have studied camera auto-calibration based on this vanishing point. This method has a strong level of flexibility [20]. However, because it is a method based on the quadratic curve or surface, it has a low level of robustness. Meanwhile, the constraints of camera movement are strong, thus making it impractical in practice.

Camera self-calibration which based on active vision [21] uses some moving information from the camera to complete calibration. This method does not need a calibration object; but, some special movement of the camera is needed, that uses the special feature of this movement, the internal parameters can be calculated. The advantage of this method is that with a simple algorithm, we can obtain a linear solution that has a higher level of robustness. But, the disadvantages are the following [14]: the high cost of systems, expensive laboratory equipment and demanding experimental conditions. Meanwhile, it is not suitable for the situations in which the position of cameras is uncontrollable.

So far, camera calibration methods have been proposed in many research works. In [22], Y. Zhang proposes a new method of camera calibration, which only requires the camera to observe a planar pattern showing at least two different orientations. The camera or the planar pattern can be moved. The motion does not need to be known. Radial lens distortion is modeled. But, there is still only a small amount of work that has been applied to the calibration of cameras. Current research should focus on its practical applications. To the best of our knowledge, the mandatory steps of TSDR system do not explicitly include a camera calibration step. However, some experiments [23] test every frame to speed up the overall system.

2.1.2 Dynamic range adjustment (DRA)

Dynamic range refers to a span that varies according to changes in objects: i.e., the range between the lowest pole and the highest pole may vary. The general description of the range is the difference between highest and lowest points. DRA is a ratio that indicates the range from the smallest to the largest measurable light intensities. The contrast of an image needs to be increased under some adaptive dynamic ranges. The high dynamic range (HDR) [24] can deliver better quality images than compared with ordinary images; it can

provide more dynamic ranges and better image details. The final image can reflect real environments in terms of human vision. This can be achieved by using each low dynamic range (LDR) image with optimal detail at corresponding exposure time to compose a final HDR image [25]. Recently, the HDR image-based systems have been used in the field of ADAS. The high quality HDR image is important for night detection as well.

According to the research in this area of the pre-processing, although there has been some research on this step, it is not explicitly included in the existing system [26]; and, research work in this area is still needed.

2.2 Detection Techniques

The main task in this section is to generate candidates which are probably signs. Thus, in this stage, we need to extract the Region Of Interests (ROIs) from respective frames and to prepare them for the recognition stage. In order to guarantee the accuracy in the recognition stage, all possible ROIs should be considered; this is because each ROI could be a sign.

There are many kinds of techniques that can be used for searching for ROIs. The object we wish to detect has obvious physical characteristics such as color and shape. These kinds of characteristics are taken into consideration and a great deal of research focuses on these aspects. Another consideration for these stages is the use of statistical methods, such as the feature sample statistics of an object. Nowadays, lots of researchers focus on this to get better results. The detail of detection methods can be divided in to three main parts: detection through colors, detection through shapes and detection through hybrid methods which combine the different kinds of methods together.

2.2.1 Detection through Colors

Information transmitted though bright color is undoubtedly the most important and notable characteristic of traffic signs, because traffic signs need to be easily perceived by humans. Examples of these colors are red, blue, and yellow [27]. Recently, the TSDR system mostly regarded color analysis as a preliminary basis for segmentation. Color-based

detection methods are used in order to segment different colors and to provide ROIs which will facilitate recognition. Color segmentation consists of partitioning an image into subsets of connected pixels that share similar color properties. Detection is often based on a process of segmentation which focuses on similar kinds of properties; and, it reduces the search space for finding ROIs.

Some detection methods are used commonly, such as color space threshold segmentation [28–34], the artificial neural network method [35], [36], regional division [37] and other methods.

Color segmentation thresholds are the most common method. Though a series of thresholds, an image is divided into ROIs distinguished by different colors; this method is simple and requires less computation. However, a series of thresholds is hard to control and when the images are affected by light and other factors, on the impacted segment will result in a low level of robustness.

The main approach to improve this situation is from the aspect of color space selection and thresholds. The object color in an image is actually the color luminance reflected under the light source; the color of an object depends on the location and intensity of the light sources. Many researchers are dedicated to finding color invariance. The color space makes evident coordinate system and subspace before such kinds of segmentation are used. The choice of a suitable color space is very important; it will affect the detection result. Different color spaces are used in much of the literature [38–46]; examples of color spaces are: RGB, normalized-RGB, HSI/HSV, YUV, YCbCr, CIELab, etc.

RGB color space

The most commonly used space in image processing is RGB color space; this space is based on human eye recognition. The RGB model assigns a range of intensity values from 0 to 255 for each pixel of RGB components in images. However, intensity values can be mixed in different ratios, showing up to 16,777,216 ($256 \times 256 \times 256$) colors on the screen (see Figure 2.1). These intensity values can represent a wide range of colors in the world. RGB color representation model forms the basis of colors; and, other spaces can be transformed from RGB representation model.

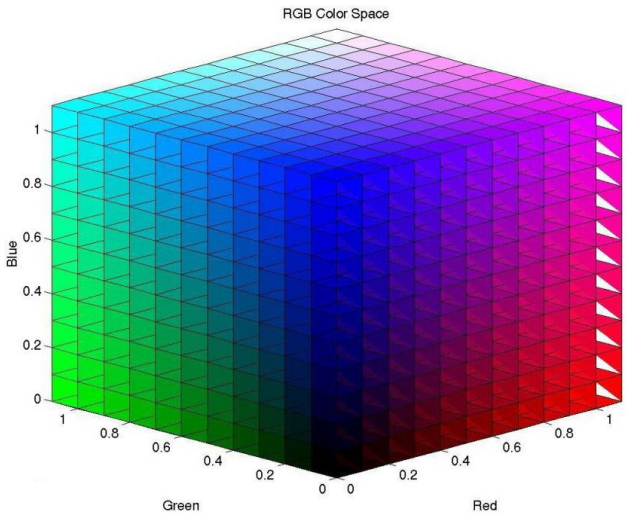
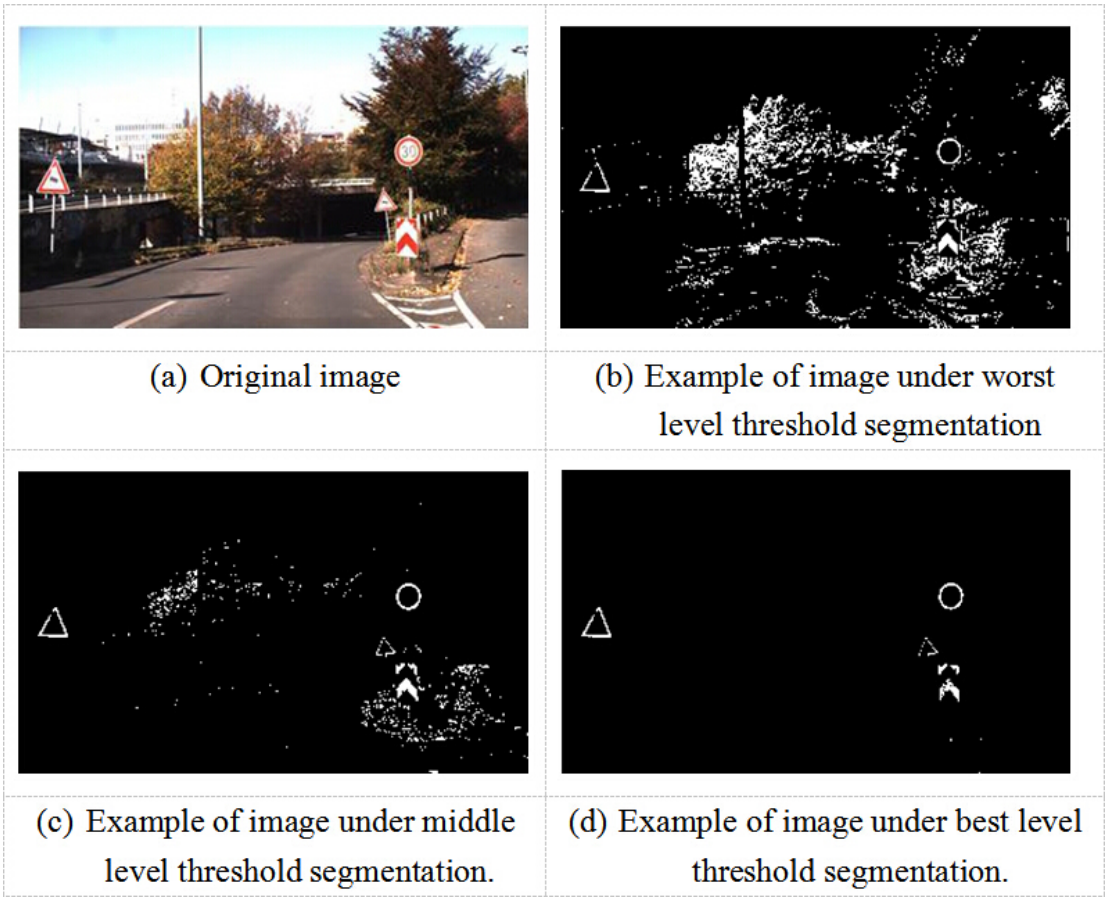


Figure 2.1: RGB color space [1]

The relationships among RGB colors are often used. The most essential consideration is the thresholds in relation to these three colors; how to divide them clearly is a big problem. In [22], color enhancement is used to extract red, blue and yellow blobs. However, in [47], chromatic and achromatic filters are used to extract red rims and white interior spaces from speed limit and warning traffic signs respectively. The segmentation through color thresholds based on RGB space showed in Figure 2.2

Figure 2.2 shows different thresholds that occur when we extract ROIs from an image taken in sunny weather based in RGB space. It is obvious that the levels of settings for threshold are hard to control; under different environmental conditions, the thresholds may be changed directly.

Since the objects have similar chromatic characteristics, the chromatic components are not directly emphasized in the standard representation of RGB color; and, when the environmental factors change, the thresholds of segmentation will also be changed. Additionally, as is the case for the RGB space, the measurement of luminance and chrominance are not independent. The early RGB color space based TSDR system can then only be used in relatively simple scenarios. A simple improvement is the parameter of normalization for RGB [48]; the definition of standardized color changes is as follows:



(a) Original image

(b) Example of image under worst level threshold segmentation

(c) Example of image under middle level threshold segmentation.

(d) Example of image under best level threshold segmentation.

Figure 2.2: RGB threshold levels

$$r = R/(R+G+B) \tag{2.1}$$

$$g = G/(R+G+B) \tag{2.2}$$

$$b = B/(R+G+B) \tag{2.3}$$

Where and are called chromaticity coordinates, $r + g + b = 1$.

The brightness of each pixel in an image equals the sum of the three RGB components. The use of normalized RGB parameters in a system can therefore reduce the impact on changes in brightness. However, the r ,g and b of the normalized RGB model only eliminate components of the relative brightness in RGB. The color saturation does not separate from the normalized model; thus, the model is still sensitive to changes in color saturation.

To separate the luminance and color entirely, the color segmentation process is then carried out on images that were initially converted to HSV (Hue or color, Saturation and Value lightness) space; or with intensity, brightness, lightness, etc., instead of Value lightness, generating HIS, HSB, HSL and other spaces model.

HSV color space

HSV is color space proposed to deal with the color that is digitized. HSV color space is based on intuitive color characteristics created by AR Smith in 1978. HSV color space model corresponds to a subset of cones in the cylindrical coordinate system.

The RGB color model is hardware-oriented; the HSV color model is, however, user-oriented. The 3-dimensional representations of HSV model evolved from the RGB cube. If we assume that we are observing the RGB cube from a white vertex to a black vertex along a diagonal, we can see a hexagonal shape (see Figure 2.3). The hexagonal border shows the color, the horizontal axis represents its purity; and, the brightness can be measured along the vertical axis. The apex corresponds to the minimum brightness value, while the center of the conical bottom surface corresponds to the maximum level of brightness.

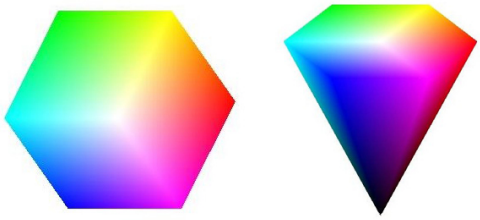


Figure 2.3: HSV color space evolution [2]

The hue (H) has an angle measurement ranging from 0-360; this begins at calculated start point with red at 0, that moves in a counter clockwise direction. Green is at 120; and, blue is at 240. The complementary colors are yellow at 60, cyan at 180 and magenta at 300. The saturation (S) is in a range between 0.0 to 1.0. The level of brightness (V) ranges from 0.0 (black) to 1.0 (white) (see Figure 2.4 for details).

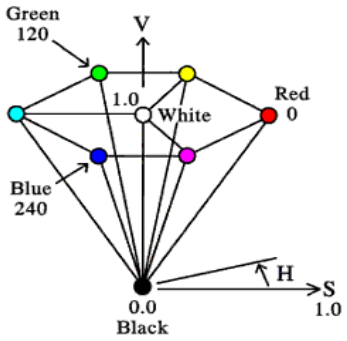


Figure 2.4: Details of the HSV model [3]

The relationship between RGB and HSV is as follows [49]:

$$H = \begin{cases} \frac{1}{6}\frac{G-B}{max-min} & if \quad R = max \\ \frac{1}{6}(2 + \frac{B-R}{max-min}) & if \quad G = max \\ \frac{1}{6}(4 + \frac{R-G}{max-min}) & if \quad B = max \end{cases} \tag{2.4}$$

$$S = \frac{max - min}{max} \tag{2.5}$$

$$V = max(R, G, B) \tag{2.6}$$

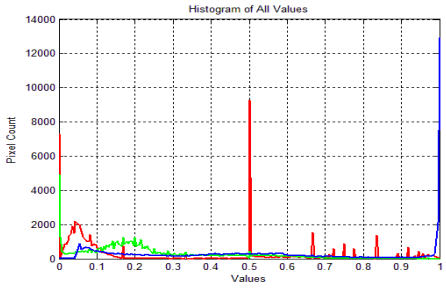
Where $max = max(R, G, B)$ and $min = min(R, G, B)$

Hasan Fleyeh [50] proposed a fuzzy color-based approach for road traffic sign detection. Firstly, the RGB images, taken by a digital camera, need to be changed into HSV space images. And then, H and S components need to be extracted. If the gray values of components have a certain color (red, yellow, or blue) in the gray scale, the gray value will be retained; if this is not the case, it will be set to 0; and, in this way, Hasan Fleyeh achieved the goal of finding the position of ROIs.

In the segmentation process, the component of hue played the central role; this is because it shows more invariance in the context of shadows and highlights, to variations in light conditions, and to changes in the color saturation [45].

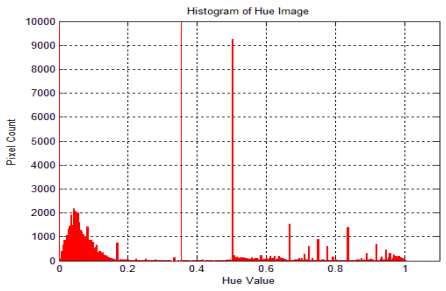
The color components analysis under HSV space:

(a1) Original image

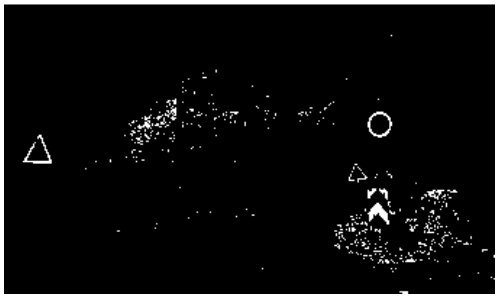


(a2) Histogram of original image

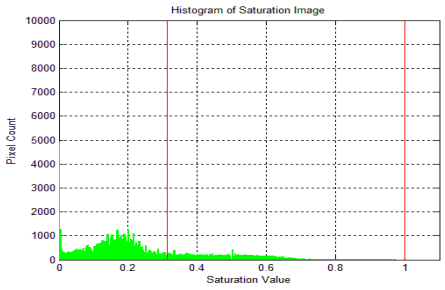
(b1) Hue component



(b2) Histogram of hue image

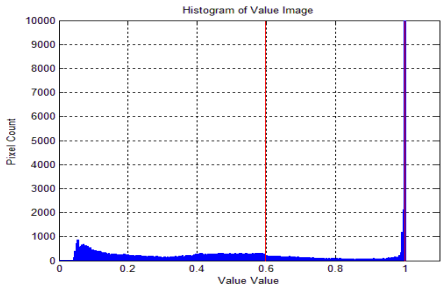


(c1) Saturation component



(c2) Histogram of saturation image

(d1) Value component



(d2) Histogram of value image

Table 2.1: HSV component analysis

Table 2.1 shows an image under HSV component analysis; and, these three components cause different desired effects on the ROIs. For example, we can easily extract ROIs in the saturated image from the table above.

HSI color space

The HSI model (hue, saturation and intensity) is used in [51] because it does not respond to changes of illumination. Empirically, determined fixed thresholds define the range of each HSI channel, where the red and blue traffic sign candidates exist; this is also mentioned in [52].

The Equation distinguishing between RGB space and HSI space is described in the following, we assume that the three components of RGB are already normalized [53]: $R, G, B \in [0, 1]$.

$$I = (R + G + B)/3 \tag{2.7}$$

$$S = 1 - \frac{min(R, G, B)}{I} \tag{2.8}$$

$$H = \begin{cases} \theta, B \geq G \\ 360 - \theta, B \geq G \end{cases} \tag{2.9}$$

$$\theta = \arccos \left\{ \frac{\frac{1}{2}[(R-G) + (R-B)]}{[(R-G)^2 + (R-G)(G-B)]^{\frac{1}{2}}} \right\} \tag{2.10}$$

Some researchers performed the experiments based on this color space. Kiran C. G et al. [54] proposed a Look Up Tables (LUTs) based on color enhancement for traffic sign segmentation algorithms. First, images are converted from RGB space to HSI space; and, LUTs are then used to enhance the luminance and saturation values of red, yellow and blue; these values probably assist in the location of traffic signs.

S. Lafuente-Arroyo et al. [55] present an HS (Hue, Saturation) threshold segmentation algorithm based on HSI space. The algorithm is expressed as follows:

$$g(x, y) = k_1 \qquad if \begin{cases} H_a \leq f_s(x, y) \leq H_b \\ f_H(x, y) \geq S_a \end{cases} \tag{2.11}$$

$$g(x, y) = k_2 \qquad Otherwise \tag{2.12}$$

$$\tag{2.13}$$

Where $g(x, y)$ is the value for the pixel (x, y), H_a and H_b denote the settings for the minimum and maximum component values of H; and, $f_s(x, y)$ denotes the pixel coordinate, S_a denotes the minimum value of component S and k_1, k_2 are the thresholds: in general, $k_1 = 255$ and $k_2 = 0$. The algorithm above combines the dilatation and erosion calculation; these two integrated components can assist in the positioning of the area in the case of red and blue fields, which are probably signs. But this space is computationally expensive due to its nonlinear formula. And during any environmental changes undergone, the pixel will undergo big changes; thus, the original image will become distorted.

Other color space

YUV color space

The YUV coordinate system was first proposed for the National Television System Committee (NTSC) transmission standard; it is mainly used in colorful television signal transmissions and the re-shuffling of R, G, B components into the luminance signal (Y) and into color signals (U,V). The formula for re-shuffling is as follows:

1. RGB to YUV

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{2.14}$$

2. YUV to RGB

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0.114 \\ 1 & -0.39 & -0.58 \\ 1 & -2.03 & 0 \end{pmatrix} \begin{pmatrix} Y \\ U \\ V \end{pmatrix} \tag{2.15}$$

W. G. Shadeed et al. [56] analyzed the advantages and disadvantages of using HSV space and YUV space for traffic sign image segmentation. They found that these two segmentation algorithms can compensate for one another; and, they discussed that when they combine these two spaces together what is carried out AND operator, they can obtain better segmentation results.

YCbCr color space

YCbCr or Y'CbCr (sometimes written as YCBCR or Y'CBCR), another type of color space, is defined as follows: Y is a component of color luminance; and, CB and CR are the concentrations of blue and red values compared to the offset component.

The YCbCr color space is showed in Figure 2.5 and the equation to transform this space into RGB space is outlined below:

$$
\begin{aligned}
Y &= 0.299R + 0.587G + 0.114B \\
Cb &= 0.564(B - Y) \\
Cr &= 0.713(r - y) \\
R &= Y + 1.402Cr \\
G &= Y - 0.334Cb - 0.714Cr \\
B &= Y + 1.772Cb
\end{aligned}
\tag{2.16}
$$

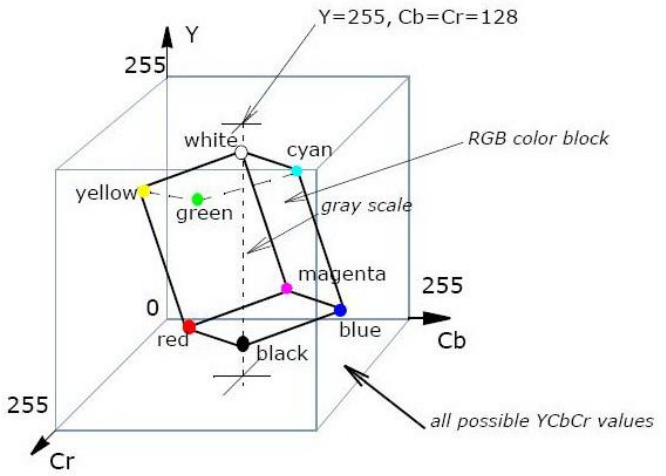


Figure 2.5: YCbCr color space [4]

This color space is seldom used since it can be changed into RGB space; this is why the RGB space is mostly used.

The existing approach is based on color space, as we mentioned above. In [39], A. Ruta et al. suggest using the color enhancement and recursive thresholds to obtain ROIs from robust RGB space; but, the recursive threshold is difficult to choose.

Jae Won Woong et al. [57] proposed a target detection method using color and edge to define significant features and a multi-scale pyramid decomposition method. However, the edge characteristic has poor robustness and a higher computational loss in a complex background. In [45], the authors H. Fleyeh et al. used HSI color space; the hue component has a good robustness to variations in illumination, but it is time consuming when images change from RGB space to HSI space. L. D. Lopez and O. Fuentes [38] use CIELab color space for traffic sign feature searching. This method needs parameter adjustment during

detection; and, it is suitable for traffic sign detection under a simple background. When the environment is complex, the algorithm is less able to ensure high accuracy.

Table 2.2 shows the techniques used by researchers in this field. The most problematic factor in color space lies in how to determine the threshold. There is no general rule for setting it; many researchers choose to use empirical thresholds.

Color space	**Ref.**	**Techniques**	**Advantages**	**Disadvantages**
RGB	[23], [39]	Color enhancement and recursive threshold.	RGB channels robust and efficient.	Recursive threshold.
RGB	[47]	Chromatic achromatic threshold.	Simple and fast.	Thresholds set to smallest.
RGB	[40]	Color histograms.	More robust.	Empirical thresholds.
RGB	[41]	RGB color Haar.	Relative difference of color feature.	Very slow during processing since the sign was a minimum of 20 pixels.
RGB to HSI	[42], [43]	Median filter and followed by noisy rejections.	Good robustness to illumination.	Time consuming.
HSV	[44]	Fixed thresholds for H,S and combined with achromatic decomposition.	Simple, fast.	Not resistant to illumination variations.
HSV	[45]	Shadow highlight invariant algorithm: threshold in H, then in S and V.	Robust to illumination changes.	Time-consuming.

HSI	[46]	HSI threshold.	Hue robust to illumination variation.	Time-consuming RGB to HIS transfer.
YUV	[58]	Multiple thresholds.	YUV color space invariant.	Thresholds set empirically.

Table 2.2: The overview of some existing color segmentation techniques

2.2.2 Detection through Shapes

Shape is effectively used to detect traffic signs; this is because traffic signs regularly have the following shapes: circular, rectangular, octagonal, triangular, etc. In addition, weather conditions and luminance instability always produce strong noise in color analysis; such factors make may result in color-based techniques failing. This is why shape-based techniques are of interest to researchers in this field.

There are shape-based methods which exploit the invariance and symmetry of the traffic signs. Existing methods of shape segmentation are template matching, Hough Transform (HT), etc [59]. Through these kinds of methods, we can distinguish circles or rectangles from the background. Although the results may contain some ROIs that may not have traffic signs, they can still provide some useful candidate ROIs for the next step. As well, ROIs containing traffic signs and other information are processed in subsequent stages. Detailed features can be found using feature descriptors such as HOG feature, Haar feature, etc.

Template matching [60] ,[61]

Template matching is a very common method in image processing and pattern recognition. It is also applied to the TSDR system, such as in the literature [61] and [62], and so on. It is a low-level approach which uses pre-defined templates to search the whole image pixel by pixel or to perform the small window matching; although template matching is a simple

operation, it requires a large amount of calculation to improve the rotation and scaling robustness. Besides, the number of new templates are required be added in, which also increase the amount of computation. In the context of the issue above, some improved algorithms have appeared. C. Grigorescu et al. [63] proposed an image matching method based on distance, which has a high level of robustness. But it is based on templates; a large number of templates is still needed.

Hough Transform

Hough Transform (HT) can be used to detect circles, lines, rectangles, etc. in images. Because of the regular shape due to traffic signs. HT does not vary in terms of rotation scaling, and noise immunity. But the disadvantage is the large amount of computation. Barnes et al. [64] proposed a light symmetry algorithm which is essentially a simple Hough algorithm which had small computation; but, this algorithm can only detect the circular signs.

In [65] the authors G. Loy et al. present a modular system based on shape symmetry; which is applied to both U.S. and European speed limit signs (SLS). The authors developed two different detection modules for circular and rectangular shapes. That is, a circular HT was adapted to the circular European SLS, and a rectangular segmentation was developed to the U.S. SLS. Fast Radial Symmetry, a variant of circular HT, is a simple, efficient method used to detect SLS signs. It is based on the symmetrical nature of circular or rectangular shapes. This method is ideal for a general natural environment; but, it has high false detection rates when there are lots of symmetrical objects in urban environments. As well, ROIs containing SLS and other information will not be processed in this method. In [23], A. Ruta et al. used the color distance transform (DT); under this method, they need to compute DT for each color channel separately. The measure of similarity is smoother and more robust to slight rotations. However, it will be sensitive to affine rotations and occlusions.

Edge detection

Edge detection is a basic method of image processing and computer vision. The purpose of edge detection is to identify the obviously bright variation points in digital image,

because significant changes in an image attribute usually reflect important transformations. These changes include (i) depth discontinuity; (ii) surface direction discontinuity; (iii) object property changes; and (iv) changes in scene illumination. But it always has heavy computation.

Corner detection is based on edge detection. According to pre-defined templates, this method works by finding a corner and then searching for the corresponding corners based on target shapes. D. L. Escalera et al. [66] define the detection of a given triangle as the search for approximately 60 degrees (52 degrees - 68 degrees) corners and the determining of whether there are three other corners constitute an equilateral triangle; rectangle detection falls under similar definitions; but, the search is for approximately 90 degrees corners. This method has a low robustness to rotation. With the increasing complexity of images, the number of corners in images will increase sharply which has a significant impact on speed and detection accuracy.

Graphic coding method

This method uses a vector chain code to describe the graphic and expressed graphic characteristics by through code string, which converts the feature matching problem to a code string comparison problem. H. Chun-Ta et al. in [67], present an ellipse detection algorithm; here the edge points are divided into six kinds of feature points according to the domains of neighbors. The authors define an ellipse based on the distribution of these six kinds of feature points. A. Soetedjo et al. extend the feature points into eight different kinds, that are combined with genetic algorithm (GA) to detect circular traffic signs [68]. L. Priese et al. [69] propose to divide the circular and triangular traffic signs into several polygons which consist of 24 small edges; they then propose to code the color candidate regions which come from color segmentation. And finally, they compare the corresponding edge of templates to determine the category.

Neural network (NN)

Using of the trained neural network to perform shape detection, S. Zhu et al. in paper [32], construct a Back-Propagation (BP) network for triangle signs. C. Fang et al. [70]

define the input layer neurons as edge detectors; and, they combine these with input color information to finish the corresponding shape judgment. However, the structure of NN is complicated.

Extraction through gradient features

Recently, many approaches using gradient orientation information (HOG feature: see Chapter 4.4.2 for details) have been successfully used in the detection phase [37]. HOG detector is different from the template matching; it does not need prior knowledge of a target. It was first used in pedestrian detection. It starts by dividing the image into a set of overlapping blocks. And, each block has several numbers of cells, in which the histogram of the gradient orientations will be computed. Some parameters inside HOG will influence the precision rate of the detection phase, such as the experiments in [71]. The strong point of HOG is that it has the following properties: scale-invariance, local contrast normalization, coarse spatial sampling, etc.

Methods	**Ref.**	**Advantages**	**Disadvantages**
Template matching and distance transformation	[23],[72]	Similarity measurement is smoother and more robust to slight rotations.	Unsuitable for real-time systems; requires many cross-correlation. Computations between the template and the ROIs for sign detection.
Edge Features	[73]	Laplacian filter to extracts outlines and edges.	Produces several ROIs, which burden the subsequent stages.
Hough Transform (HT)	[43], [51], [74], [75]	[51]: combine HT with an iterative process of median filtering and dilation to refine the sign candidate set. [74]: robust to illumination changes.	[51]: remains computationally expensive, especially for large images. [74]: may fail when many edges are present in images. [75]: too many interferon on the road.

Gradient feature	[32], [71], [76], [77], [78], [79]	It has accurate invariance to scale, local contrast normalization, coarse spatial sampling, and fine weighted gradient orientations.	Computationally a bit expensive. Needs a scale of samples to get features.

Table 2.3: The overview of some existing shape segmentation techniques

Other methods

Traffic signs can be divided into external and internal contour graphics. Jiang et al., in [80], use the mathematical morphology method to extract bones in order to further improve the template matching results. Jesmin Khan et al. [81] use the peculiar shape properties of the traffic signs, such as aspect ratio, using the ratio between the perimeter squared and the area to classify different shapes. For example, when the ratio of the perimeter squared and area is 9 to 11.75, it corresponds to octagon traffic signs. However, in the actual work, the images of traffic signs taken by a camera occur on different levels of projection distortion; the general shape of an equilateral triangle becomes generally triangular, and a circle becomes an ellipse. Additionally, a traffic sign may be partially corrupted; thus, the accuracy rate of this method in [81] might decline. Based on this, Pedro Gil Jimenez et al. performed some processing after the traffic sign image segmentation stage in order to correct the distortion of the shape. They thus filled the damaged traffic sign images [82].

2.2.3 Hybrid Methods

Both color-based detection and shape-based detection have different advantages and disadvantages. Color-based detection is simple and requires a small amount of calculation. But, it has low robustness to light, weather changes, image noise, etc. Color segmentation often requires a threshold; the process needed to obtain the threshold is often tedious and has a certain error rate [83]. The approach based on shape reflects a robustness to light, since the shape of signs is always uniform. However, it is easily impacted by similar objects, such

as by a window, a mailbox, or a billboard. And, it is very sensitive to occlusion. Thus, many papers have adopted the color and shape features in a combination of methods; most papers, [30], [32] and [37], use these with colors-based preliminary segmentation. This step is immediately followed by the sign detection stage that is applied on the generated ROIs; and, it uses shapes to do post-judgment. Some papers [28–30], [70] and [84, 85], used color and shape detection simultaneously; this is done while determining the final outcome by way of fuzzy judgment. It is observed that this method has had lower false positive in comparison with non-hybrid methods.

After doing a large number of literature reviews, we found, in the process of building TSDR system, that it is hard to select only one method among several options. Nevertheless, gradient feature-based algorithms, which also can be combined with color analysis, are obviously useful over other approaches (see the details in Chapter 4.4.2). Our following proposed system is also based on this kind of method.

2.3 Classification Techniques

The classification techniques are used to determine the content of the detected list of ROIs which possibly contains one or more signs. They are then classified, with regards to their content, into the appropriate categories (stop signs, speed limit signs, etc.).

The main methods, in this stage, for recognizing signs are machine learning techniques, digit recognition techniques (only for signs with digits or text), template matching, and tree classifiers such as K-d trees and Random forests. The choice of which technique to use depends on the output of the detection phase.

We have already introduced template matching in the former section (Section 2.2.2). It still can be used in this chapter. It requires a lot of templates in a dataset, when similar reappearances of unknown ROIs are being searched for. An interesting method for simultaneous detection and recognition, based on template matching, is found in [23]. With the appearance of Genetic Algorithm (GA), Neural Network (NN), Support Vector Machines (SVMs), etc., more and more researchers have applied these algorithms to the traffic sign recognition phase. For example, Yuji Aoyagi et al. combine GA and NN; this results in the use of GA for the detection of traffic signs and NN for the process of

classifying [73]. However, GA has the following disadvantages: premature convergence and local optimum solutions; the application of this algorithm is subject to restrictions. Over recent years, statistic-based theory like SVM algorithm seems to be more popular, this is because it is based on a minimum structural risk; it is a learning algorithm that uses small samples and this method can achieve better results for traffic sign classification. It guarantees a high recognition rate and is moreover robust. In [47] and [86], SVM algorithm was used to identify traffic signs. In [92], the authors S. Lafuente-Arroyo et al. used different kernel functions and different types of SVM models. They compared the ability of the simulated annealing search method and the grid search method. And finally, with SVM test training, they achieved traffic sign recognition results. At the same time, HOG descriptors were also applied to SVM [78].

S. Lafuente-Arroyo et al. use Distance to Borders (DtBs) to extract feature vectors. They performed this with training and by testing components done by linear SVM, which achieved good results [87]. Kiran C. G et al. proposed Distance of Centers (DoCs) method based on DtBs; and, they combined these two methods to extract the same amount of data as the feature vectors. The result is that the accuracy of the algorithm has been improved [54].

Invariants moment theory has been applied to the detection and recognition of traffic signs. This is because it has invariance of rotation and scale. Invariant moment algorithm can be also used to extract feature vectors to SVM for the classification and identification of traffic signs [88].

Joint Transform Correlation (JTC) is a kind of pattern recognition technology. And, it has played a good role in traffic sign classification. In [81], the authors applied this technique to recognize traffic signs. They divided traffic signs into two groups: one group needs to be recognized; and, the other group encapsulates the standard images stored in the dataset. When the two groups are consistent, the output of corresponding JTC has a large peak value; otherwise, it is small. Two correlation peaks can clearly show multiple relationships. In this way, they can achieve recognition results for traffic signs.

Another comparative study was introduced to evaluate the performance of K-d trees, random forests, and SVMs for traffic-sign recognition [71]. The before-mentioned authors built a random forest which had 100 trees with more than 500 features. A random forest

classifier is a kind of decision tree. It contains many decision trees and yields the class that is the mode of class output of individual trees. The authors show that random forests outperform SVMs and k-d trees due to their unbalanced data sets.

Digital recognition is also a method used in this phase. It can classify traffic signs according to the text in them. Optical Character Recognition (OCR) is one of the popular methods used; the original concept dates back to 1929. It was proposed by German scientist Tausheck. It is used for getting the character information of images. Thus, it is used for text recognition of traffic signs; but, only the signs with text fit to this method. However, we still have a lot of traffic signs without text: an example of this is warning signs that trucks use. This means that OCR cannot satisfy the requirements of those traffic signs.

Despite the large number of papers, the classification steps of signs are still not perfect. Among classifiers found in the literature, random forest and SVMs which occupy powerful positions in the classification step. Ultimately, these different methods fit different conditions. However, the same purpose for them in this phase is to increase the recognition rate.

Ref.	**Techniques**	**Advantages**	**Disadvantages**
[23]	Distance Transform Hierarchical matching.	Fast.	Not robust to occlusions, sensitive to pre-segmentation.
[75], [89], [90]	Hough Transforms.	Robust, simple.	Parameter tuning.
[78], [91], [92]	HOG, SVMs.	Robust, good performance.	Slow, large concatenated HOG vectors.
[71]	Tree classifiers.	Easy to train and to update.	Generally requires a large training set.
[93], [94]	Digit recognition.	Fast.	Strong restrictions placed.

Table 2.4: The overview of some existing classification techniques

2.4 Tracking Techniques

The tracking stage is always used in the situation where the object is moving quickly. Examples of this are found in pedestrian detection or animal detection. This is because the position of these objects will change randomly. However, a traffic sign has different targets since the position of traffic sign do not change automatically. The goal of TSDR system is to give warning information to drivers. This is the reason why there are few researchers working on this part other than on the above two stages.

However, we still found some research work on this step; in [70] and [95], tracking is used to reduce the memory and processing costs. When the ROIs have been found, they are tracked over several frames to reinforce the decision and performance.

There are some implementations based on Kalman Filter (KF). These came from [96] and are used in [70]. KF is often used for linear systems and is used to produce a statistically optimal estimation of the position and scale of signs in consecutive frames of input video. But, the following three problems appear in the KF used by TSDR system: (1) unrealistic assumptions related to the real size of signs; (2) the constant speed of vehicles, which have to be known prior; (3) the assumption that trajectories should be straight. KF is as an optimal stochastic filter used to estimate the motion parameters in the respective following situations: prediction and correction. It cannot solve the nonlinear problem.

The other filter called particle filter (PF) was elaborated on in [97]. Some researchers have chosen the PF. This is because it can be used for nonlinear problems and can be applied to multimodal problems. However, the problem of PF is that a large number of samples will be used to approximate the posterior probability density of the system.

Thus, the authors in [98] used another method called filter which came from [99]. The filter is used to model the relative motion between the camera and traffic sign in the scene. However, the filter is expensively computed. Other authors like L. D. Lopez and O. Fuentes et al. [38] used Gaussian models to detect road signs. This method combines detection and tracking in order to reduce the required computational time. It changes the parameters used when tracking. But, when it is used under a complex environment, it still cannot provide high accuracy.

From the literature we have reviewed, this stage has not received as much attention as

other stages. In our opinion, the system is only intended to give warning information; and, the information will be transmitted to other vehicles on the same road and in the same direction. So, if accurate information is obtained easily, it is not necessary to track the signs; we only need to send the information out. In our research work, we focus more on the detection and recognition part as many researchers have done.

Chapter 3

Traffic Signs Dataset Creation

3.1 Background

Before building the system, another important component part is the dataset. We know that if we want to use object features, we need a lot of samples for that object. The number in the dataset depends on the techniques used in detection and recognition steps.

Existing open source datasets, like ETH80, was first introduced in [100]. This dataset has a total of 3280 images, containing 10 kinds of objects in 8 classes. Each object was taken at 41 different angles. Caltech dataset [101] contains 101 categories and 40 to 800 samples in each category. It collects from the Internet. There are two famous datasets used for human detection: the INRIA person dataset, used in [76], contains the original images, positive images in normalized 64 128 pixel format and original negative images. The other is the Yale face dataset [102], which contains 165 gray scale images in GIF format of 15 individuals with different facial expressions.

For TSDR system, we found there is a famous open source dataset widely used for traffic sign detection which is a German traffic sign dataset. German Traffic Sign Benchmark has two datasets: the German Traffic Signs Detection Benchmark (GTSDB) and German Traffic Signs Recognition Benchmark (GTSRB). These were built and used in a competition at International Joint Conference on Neural Networks (IJCNN) 2011 [114]. The datasets include more than 50,000 images of German road signs in 43 categories: they were divided into 3 categories that suit the properties of various detection approaches. This dataset can

almost represent the European traffic system.

3.2 Traffic Sign Specifications

The traffic sign is a representation that combines text and images used to give drivers information about roads: such as speed limits, warning dangers concerning icy road conditions, construction alerts and warning for pedestrian crossings, etc.

Traffic signs may have different backgrounds and colors. For example, in European traffic systems, the speed limit signs are totally different when compared with North American speed limit signs. In addition, although the speed limit signs in Europe are similar to those in Africa, there still exist many signs specific to Africa; for example, in Figure 3.1, there is a sign which has an elephant on it. For the warning signs in Canada or Europe, we can often see a sign with a moose on it (Figure 3.2). This difference is due to the climate of different regions and to the different animals that may appear. These are the reasons that make signs reference different objects.

Figure 3.1: African warning traffic signs with animals

Figure 3.2: Canadian warning traffic signs with animals

3.2.1 Differences between European and North American Signs

As we discussed in the section above, there are different kinds of signs around the world. Figure 3.3 is a sample of a German dataset. However, there is no existing dataset or at least no open source dataset for both the USA and Canada. Since our system for traffic sign is suitable to North America, especially to Canada, in order to match our situation, a new dataset for traffic signs is needed.

Figure 3.3: Samples of signs in German dataset

At first, we will focus on the speed limit signs in North America, since the other kinds of signs are the same or have only a small difference between the warning signs of America and German dataset. We can use the same part of the German dataset. But the speed limit in North America is extremely different from those in the existing dataset.

The SLS are used to regulate the speed of vehicles and motorcycles in downtown and highways areas. Hence, they protect drivers and pedestrians against dangerous situations. The SLS in North America have a rectangular shape which is contrary to the circular European SLS. The speed limit text is indicated in the middle of the signs on white, yellow, black or orange backgrounds with a black border. European SLS are circular white signs with a red circular border [103], see Figure 3.4 below.

The Canadian SLS usually has a notice saying the MAXIMUM XX on signs to indicate

Figure 3.4: Samples of different speed limit

that XX is the maximum authorized speed; this has been expressed in kilometers per hour (km/h) since 1977. American SLS use SPEED XX to express the same meaning and still display the message in miles per hour (mph) [103].

3.3 Dataset Creation Process

3.3.1 Data Collection

This section will introduce the process of building our own dataset. Our dataset of North American SLS (NASLS) was created from approximately 10 hours of different videos. These were recorded while driving on different types of roads and different weather conditions. These videos were taken in the streets of Ottawa during the daytime, mostly during March 2013 and June 2013. For video collection, a Sony Full HD 1920 1080 pixel camera was used. After video collection, we wrote a simple program to perform the frame extraction which isolated individual frames of video under different parameters depending on vehicle speed and resolution. For example, if the video has a high resolution and the car is traveling at a low speed during recording, the extraction time will then be made a little longer since there is a small difference from frame to frame. Thus, the traffic signs extracted from these kinds of frames will be somewhat similar. This offers little contribution to the diversity of the dataset. The purpose of this dataset is to collect as many different kinds of SLS under different environmental conditions as possible. So when the speed of recorded cars is high, and the videos are taken under bad conditions, we need to pay more attention on the frame selection to guarantee the quality of the dataset.

3.3.2 Data Organization

The images extracted from videos are selected a second time, since not every extracted image can be used for the dataset.

The other criterions of collecting data are as follows:

1. Discard the classes with less than 50 images.
2. Discard the classes which have very bad image quality (i.e., when human cannot see the signs clearly, they will be discarded). Since the car passes different traffic signs with different velocities, the selection step, which depends on the position of a sign and on the traffic situation, is very necessary. The occurrence of a fixed number of images per traffic sign after the selection step increases the diversity of the dataset. This enables the avoidance of an undesired imbalance caused by a large number of nearly identical images.

Generally, there are different kinds of traffic signs, such as circles, rectangles, triangles, etc. And, the lengths of brims and ratios are different. So we unify the differently cut shapes into a square. The dataset contains more than 2000 SLS in Canada. These are normalized to vary from 2525 and to 650650 pixels. See Figure 3.5 for an example of the different sizes in our NASLS.

Figure 3.5: Different size of samples in NASLS

Each image contains a 7%-10% margin around traffic sign to allow for the use of edge detectors [104]. The first version of NASLS dataset contains mainly a training set (positive

and negative) for standard signs (white background with black lettering). The negative training samples contain thousands of images without any SLS. They are collected from the videos or Internet. The second version of NASLS contains both a training set and a testing set with different background colors. One point which needs to be mentioned here is that the test set, in the whole dataset, does not contain any training set. Four main SLS categories are collected for future use. These include yellow, white, black and orange signs of different meanings. The sign with a white background is the enforced mandatory speed limit. It is usually seen on both urban streets and highways. The SLS with a yellow background is just a warning sign and gives suggestion to drivers; it is not mandatory. The orange background signifies road construction. All of the above signs with white, yellow or orange backgrounds have black lettering. Rarely, a sign used for night warning has the opposite: a black background with white lettering.

Each category is supposed to contain a range of specific speed values called classes. White signs contain 10 classes: 10, 20, 30, 40, 50, 60, 70, 80, 90, and 100 km/h. The yellow advisory sign category contains 7 classes: 25, 30, 35, 40, 45, 50 and 65km/h. The orange working area signs category contains 3 classes: 25, 50 and 80km/h. The nighttime sign category contains 4 classes: 35, 45, 60 and 65km/h [103]; see Figure 3.6.

We emphasize that the dataset is under construction and is updated regularly by adding new signs.

Figure 3.6: Speed limit samples for NASLS

Figure 3.7: Samples of negative images in a dataset

Chapter 4

TSDR System Architecture Design

In this chapter, we start by introducing some concepts behind the techniques we used in detection stage such as HOG and MSER; and the methods used in the recognition stage like SVM and Random Forest are also introduced. After that, we describe the architecture of basic TSDR system. Meanwhile, we perform the improvements on the basic method in order to make the system fit more with the situation in Canada and to try to shrink the time consuming of the system.

4.1 Algorithm used in Detection Stage

In the present method of object detection, the first step is to extract features from the object during a step called training step and to build the detection descriptor. The second step is to select the corresponding detected platform, according to the first descriptor.

4.1.1 HOG Descriptors

HOG descriptors are used in the field of computer vision and image processing to obtain object features. It was first proposed in [76] by Navneet Dalal and Bill Triggs in 2005. HOG descriptors and SVM classifier aim to detect pedestrian in both moving videos and static images. These kinds of features have an excellent performance compared with others [76]. Figure 4.1 demonstrates the overview of extraction HOG features extraction.

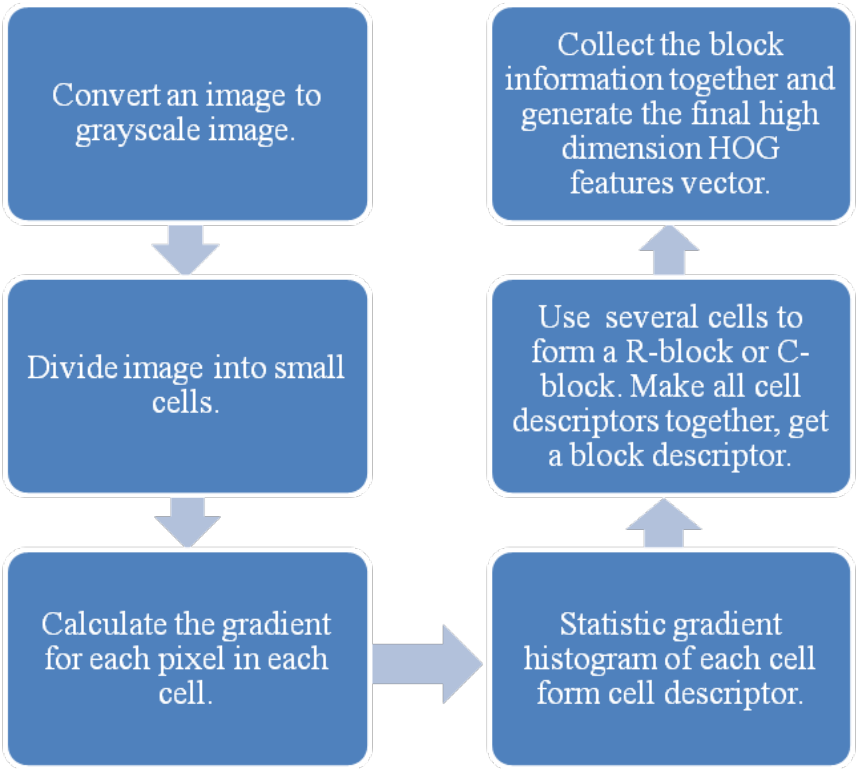


Figure 4.1: Overview processing of HOG features extraction

Color normalization

The input color image is converted into a gray scale image; this is done using Gamma correction method to standardize the input image color space (normalized). The purpose of this step is to adjust image contrast, reduce the impact of local image shadows and illumination changes, while at the same time, suppressing noise interference.

Gradient computation

This step consists in computing the horizontal and vertical gradient [76] for each pixel. It aims to capture the contour information and to weaken the interference of light.

The horizontal gradient $G_x(x, y)$ and the vertical gradient $G_y(x, y)$ are obtained by convoluting the simple, but best 1-D gradient operator [76]

- Horizontal operator: $[-1, 0, 1]$
- Vertical operator: $[-1, 0, 1]^T$

This means that the gradient of pixel (x, y) is as follows:

$$G_x = H(x + 1, y) - H(x - 1, y) \tag{4.1}$$

$$G_y = H(x, y + 1) - H(x, y - 1) \tag{4.2}$$

where $H(x, y)$ is the gray value of pixel (x, y). The gradient magnitude $G(x, y)$ and gradient direction $\alpha(x, y)$ can be also obtained by the following equations:

$$G(x, y) = \sqrt{G_x(x, y)^2 + G_y(x, y)^2} \tag{4.3}$$

$$\alpha(x, y) = \tan^{-1} \frac{G_y(xy)}{G_x(xy)} \tag{4.4}$$

Orientation histogram creation

The goal of this step is to build each cell unit histogram of gradient directions in an image. The "cell" is divided from images and each cell has $n \times n$ pixels which have a predefined size. Each pixel in a given cell performs a weighted vote to the histogram channel which is orientation-based. That is, each vote has a weighted value; this is calculated according to the gradient magnitude of the pixel. Generally, the histogram channels are evenly spread over $0° - 180°$("unsigned") or $0° - 360°$("signed"). Normally, increasing the orientation bins would extend the information of the descriptor. However, Dalal et al. in [76] showed that when testing pedestrian detection, unsigned gradient and 9 orientation bins will give the best performance. In addition, as stated in [76], magnitude itself or its function can be used to represent the weighted values; and, the best performance is given by magnitude itself during pedestrian detection.

Histogram normalization and descriptor blocks

Due to the partial variations in illumination and the background contrast changes, the range in variation of gradient strength is very large. Thus, gradient strength normalization is needed. Dalal et al. [76] adopted a method which combined the individual cells into larger, space connected units called "blocks". In this way, the HOG descriptor becomes a vector which consists of all cell histograms. Because of the overlap between blocks, each cell will contribute more than once to the final descriptor. The shape of the block can be either rectangular (R-HOG) or circular (C-HOG). R-HOG is the normal shape. And, usually each block consists of $m \times m$ cells; and, each cell has $n \times n$ pixels. The gradient direction of each cell is divided into z bins, after the weighted project gradient of z bins; each cell produces a feature vector of z dimension. Dalal et al. [79] selected $z = 9$, an unsigned gradient in human testing. The dimension of pedestrian feature vector will be as follows:

$$N = (\frac{R_{width}}{M_{width}} - 1) \times (\frac{R_{height}}{M_{height}} - 1) \times m \times m \times z \tag{4.5}$$

Where R is the region of an image, $M = n \times n$ pixels, $m \times m$ is the number of cells per block, and z is the vector bins of per cell.

In the block normalization part, Dalal et al. [79] used four different methods for block normalization to make feature vectors more robust to local illumination changes.

$$L1 - sqrt : v \leftarrow \sqrt{\frac{v}{(\| v \|_1 + \varepsilon)}} \tag{4.6}$$

$$L1 - norm : v \leftarrow \frac{v}{(\| v \|_1 + \varepsilon)} \tag{4.7}$$

$$L2 - norm : v \leftarrow \frac{v}{\sqrt{\| v \|_2^2 + \varepsilon^2}} \tag{4.8}$$

$$L2 - Hys : v \leftarrow \frac{v}{\sqrt{\| v \|_2^2 + \varepsilon^2}} \quad (v \leq 0.2) \tag{4.9}$$

$$\tag{4.10}$$

This is the case where v represents an original feature vector, and is a very small value used to ensure the denominator is bigger than 0.

For $L2-Hys$ normalization method, it first performed as $L2-norm$ method; and after that, the results are truncated to 0.2; then, the results are re-normalized use $L2-norm$ again in case to get the final results. Results from Dalal et al. [76] illustrate that only $L1-norm$ performance is reduced slightly compare with the other three methods.

Once this normalization step is finished, all the histograms can be concatenated into a single feature vector. That means, a general HOG feature vector is generated. See Figure 4.2, it gives examples of HOG features extracted based on the processing mentioned above.

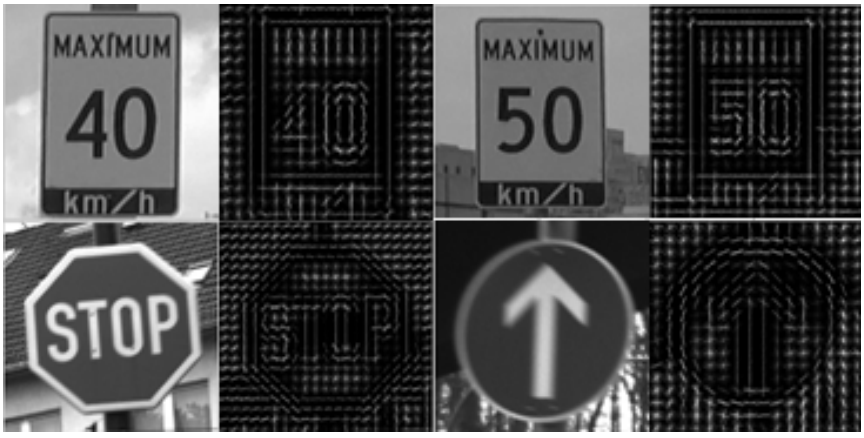


Figure 4.2: Examples of the features extracted by HOG

4.1.2 MSER Segmentation Used for Detection

In computer vision, Maximally Stable Extremal Region (MSER) is used as a method of blob detection in images. MSER is the most stable region obtained by using different gray thresholds to binarize an image.

This technique was proposed by J. Matas et al. in [105] to find correspondences between image elements from two images with two different viewpoints. This method extracts a comprehensive number of corresponding image elements that contribute to wide-baseline matching. Currently, this method has the best performance on affine invariant regions.

Before starting the extraction of the feature region, MSER first converts the target image to grayscale images. It then gets the connection sequence of pixels in a given threshold condition. The stability of these connection components is next tested. Finally, the most stable extreme region is obtained. The grayscale value in the most stable region is greater or lesser than the pixel grayscale value on the boundary.

MSER has the following features:

1. It is invariant to affine changes of grayscale images.
2. The set region of support is relatively stable. When the threshold value changes within a certain range, the area of value does not change with that value.
3. It can detect different levels of regions.
4. Since there is no smooth preprocessing for images when using MSER, it can be used in a variety of scales. Both the small size of MSER and the large size of MSER can be detected (size variability).
5. MSER suits most image structures, regardless of whether the picture structure is coarse or subtle.

Candidate regions are extracted by using the MSER method through the following process. First, a series of thresholds (gray level) from 0 to 255 are used. And, when the threshold increases at the same size of step, the gray value of each pixel is tested. After the pixel with a gray value smaller than the threshold is ignored, the remaining part is

(a) Original image (b) Mark MSER region at a selected threshold

Figure 4.3: Experiment results under MSER analysis to extract and mark ROIs from a scene

counted. The region made by the connected pixels in the remaining part becomes the extreme region. The extreme region areas do not change with the threshold changes, this region is the MSER we wanted.

Thus, from the step above, we know that MSER method allows regions to maintain their areas. MSER segmentation is selected under several thresholds, because of its quality towards variations in lighting conditions; these differ from other methods that select candidate regions by border color. And, the color of the border stays stable during the MSER process.

Our experiment results developed through testing in Table 4.3 show that MSER can be used in our system to segment traffic signs from backgrounds and to get ROIs. From our results, we can discern the best series of thresholds used to control the accuracy of regions as well as to reduce the number of candidate regions.

4.2 Algorithm used in Classification Stage

Using the knowledge obtained from a set of training samples, we can identify the class of input images; and, this process is called classification. Generally, the human eye can recognize an object in a specific category. However, without classification algorithms, computers cannot recognize simple differences. Therefore, we need to train the samples and to tell computers what belongs to which categories.

The classification algorithm has two main effects, the first is to process the object features in the training process in order to generate the classifier based on the image dataset; while the second is to analyze the input image and to find the category of the detected object. In this chapter, two popular classification algorithms will be introduced: such as Support Vector Machines (SVMs) and tree classifiers.

4.2.1 Support Vector Machines (SVMs)

Support Vector Machines (SVMs) transform an input space into a high-dimensional space through nonlinear transformation which is defined by the inner product. SVMs then find the optimal separating hyper plane in this space. The classification SVMs function forms similar to a neural network; the output is a linear combination of intermediate nodes and each intermediate node corresponds to a support vector.

A Support Vector Machine (SVM) [106] is a binary classifier in objects detection field. The best distinction between two classes is represented by a subset of data samples called "Support Vectors". A classical application for SVM is pedestrian detection; this was proposed by Dalal et al., the researchers who adopted locally normalized HOG descriptors and applied SVM as a baseline classifier throughout their study in order to build an excellent pedestrian detector [76].

A test sample can be classified, depending on its distance, to the support vectors. The binary SVMs can be combined into an ensemble composed of one-vs-one or one-vs-all classifiers to solve multi-class problems.

SVM is a useful binary technique for data classification. We will take the example of when 2 kinds of samples are classified in a two-dimensional space, as in Figure 4.4. The square C2 and circle C1 represent two kinds of samples. Obviously, they can be separated by the line l, where l is called hyper plane. In the 2-dimensional space, l is a line, while in 3-dimensional space it will be a plane.

In the case where we want to classify two kinds of objects, suppose the obvious data is $(x_1; y_1), (x_2, y_2), ..., (x_k; y_k)$. For instance, the label pairs are $(x_k; y_k)$, where are the training examples and represents the labels $(-1, +1)$; $+1$ means a positive sample and -1 means a negative sample. Moreover, k is the number of samples and n is the dimension of input

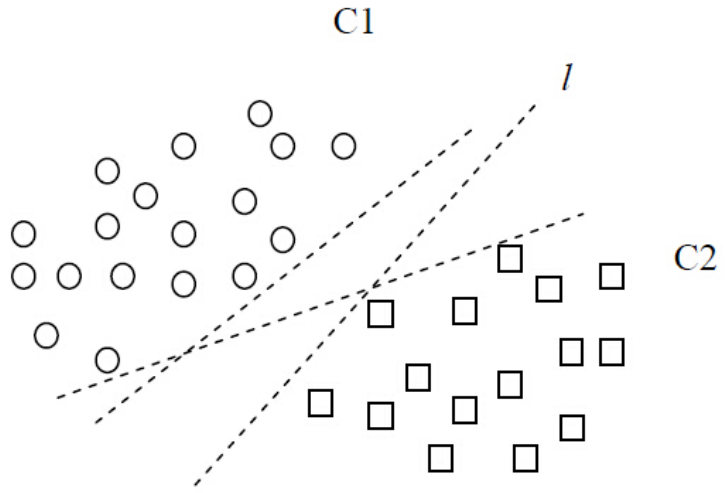


Figure 4.4: Classification of two kinds of samples in 2-dimensional

space. It looks for an optimal hyper plane H, as a decision function in a multidimensional space, in order to separate between classes.

The SVM consists of mapping x_k into a high dimensional space by a function. Next, it finds a linear separating hyper plane with the following equation:

$$w \cdot x_i + b = 0 \tag{4.11}$$

Where w is the normal direction of the Hyper plane and

$$if y_k = +1, w \cdot x_i + b \geq 1; \tag{4.12}$$

$$if y_k = -1, w \cdot x_i + b \leq -1; \tag{4.13}$$

Obviously, there is not only one hyper plane in Figure 4.4. So the problem becomes that we need to find an optimal hyper plane, which has the ability to separate all the kinds of samples. And, if the distance between the nearest vector and this hyper plane is larger compared with other planes, the hyper plane is then called an "optimal hyper plane". This is shown in Figure 4.5, when $H \parallel H_1 \parallel H_2$, H is the optimal hyper plane.

In the case of all samples, samples which make contribution to the optimal hyper plane and decision functions are called support vectors. Under normal circumstances, only a small part of the total training samples are called support vectors.

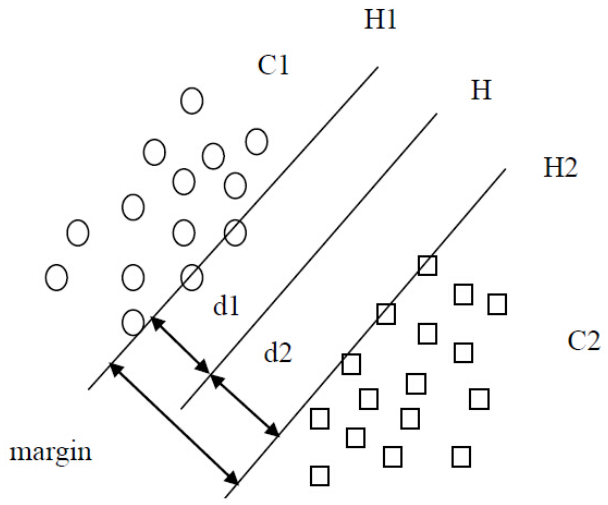


Figure 4.5: Optimal hyper plane under 2-dimensional space

We can easily find that most of the satisfy the following formula:

$$y_i(w \cdot x_i + b) \geq 1; \tag{4.14}$$

Where k is the sample number. And here:

$$p(w,b) = \min_{x_i:y_i=-1} d(w,b;x_i) + \min_{x_i:y_i=+1} d(w,b;x_i) \tag{4.15}$$

$$= \min_{x_i:y_i=-1} \frac{\mid \langle w, x_i \rangle + b \mid}{\parallel w \parallel} + \min_{x_i:y_i=+1} \frac{\mid \langle w, x_i \rangle + b \mid}{\parallel w \parallel} \tag{4.16}$$

$$= \frac{1}{\parallel w \parallel}(\min_{x_i:y_i=-1} \mid \langle w, x_i \rangle + b \mid + \min_{x_i:y_i=+1} \mid \langle w, x_i \rangle + b \mid) \tag{4.17}$$

$$= \frac{2}{\parallel w \parallel} \tag{4.18}$$

Where $\parallel w \parallel$ is the Euclidean norm, $p(w,b)$ is the classification interval margin.

With the maximal margin in this higher dimensional space, the $p(w,b)$ will obtain the maximum value; that is $\frac{2}{\|w\|}$ is the maximum, which equals to $\parallel w \parallel$ which is the minimum. And, the problem can be changed to get the minimum value of $\frac{1}{2} \parallel w \parallel^2$. And, the whole problem of getting an optimal hyper plane is changed to the following quadratic programming problem:

$$\begin{cases} \min(w,\xi)\dfrac{1}{2} \parallel w \parallel^2 \\ subject \quad to \quad \{y_i(w \centerdot x_i + b) \geq 1\} \end{cases} \tag{4.19}$$

But, as Figure 4.6 illustrates it is difficult to use hard margin linear classification to carry out a problem like this. Hard margin classification method is vulnerable to a few sample points, which is not reliable. In this problem, slack variables allow a certain degree of fault tolerance, which enables the following:

$$\begin{cases} y_i(w \centerdot x_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0, \forall i \end{cases} \tag{4.20}$$

Where $i = 1, 2, ..., k$, k is the sample number.

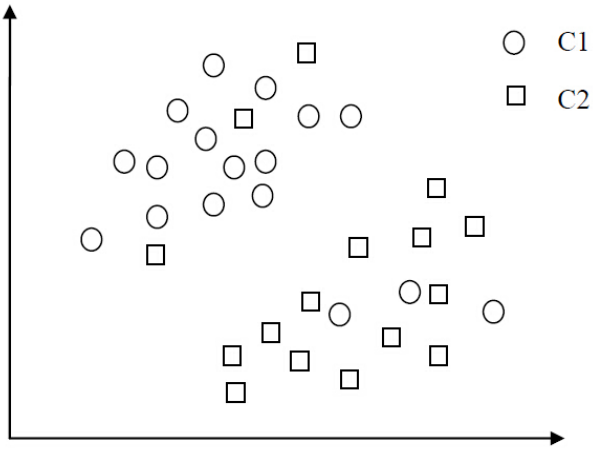


Figure 4.6: Situation of non-linear problems

Due to the following slack variable $\xi_i \geq 0$, outliers cannot be completed and correctly classified. But, these outliers can be considered as some noise which is caused by various behavioral mistakes. Thus, outliers do not necessarily need to be accurately classified. This will inevitably cause the loss of classifiers, but the advantage is that a large geometric margin can be obtained. The penalty factor C (Cost) can be weighed against such losses. The optimization function for Formula 4.19 is transformed as follows [107]

$$\begin{cases} \min(w,\xi)\dfrac{1}{2} \parallel w \parallel^2 \\ subject \quad to \quad \{y_i(w \centerdot x_i + b) \geq 1 - \xi_i, \xi_i \geq 0, \forall i\} \end{cases} \tag{4.21}$$

Where $i = 1, 2, ..., k$, k is the sample number. $C > 0$ is the penalty parameter of the error term. The solution of this problem is obtained using the Lagrangian theory. It is shown in [113] that the vector is like the formula below:

$$w = \sum_{m}^{i=1} \alpha_i^* y_i x_i \tag{4.22}$$

Where is the solution to the following quadratic optimization problem:

$$\max(W(\alpha)) = w = \sum_{m}^{i=1} \alpha_i^* - (\frac{1}{2} \sum_{m}^{i,j=1} \alpha_i \alpha_j y_i y_j K\langle x_i, x_j \rangle) \tag{4.23}$$

under the constraint: . $\forall i, \sum_m^{i=1} y_i \alpha_i^* = 0, 0 \leq \alpha_i \leq C$.

Note that the solution of the SVM problem depends only on the kernel function $K(x_i, x_j)$; this can be for example a linear, polynomial, radial basis (RB) or Sigmoid function, etc. In our case, a linear kernel function has been used.

4.2.2 Tree Classifiers

There are several approaches used to solve the classification problem. Unlike the SVM previously mentioned, tree classifiers like the K-d tree, use optimized techniques such as hierarchical structures and Best Bin First Approximate Search [108]. These techniques are used to efficiently comb through the training samples and to determine the Nearest Neighbor of the test sample.

Binary decision trees organize the data by splitting it hierarchically. The nodes of the tree then recursively divide the data space. To classify a new sample, the tree is traversed down to the leaves. The sample is compared next to the information stored in the nodes and the final leaf.

In this section, we introduce the Random Trees in the Random Forest (RF), which were introduced by Breiman et al. in [109]. This process selects the feature with the highest Information Gain. An extensive description is given in [56] and [110].

Random Forest

Random Forest is a machine learning method for classification. It operates by constructing a multitude of decision trees during the training time and outputting the class that is the mode of the output of the class of individual trees. RFs are composed of many decision trees. Because the formation of these decision trees uses a stochastic method, they are individually called the random decision trees. There is no tree in the forest between the random associations. When the test data is performed on random forests, it is, in fact, used to make a decision tree for each classification. Finally, that class is used for the final results of the largest tree in the classification results.

A random tree is grown in the following way [111]:

1. The tree is grown using a subset of training samples $I \subset I_N$ which is randomly chosen with a replacement.

2. In each node, a subset of features F is randomly chosen. According to the feature $f \in F$ and the threshold $t \in [\min(f), \max(f)]$, the current data subset is separated as I_{learn} and I_{train} with the maximum information gain Δ. The entropy E is calculated over the label frequencies in I.

$$\begin{aligned} I_{learn} &= i \in I \mid f_j < t, \forall 1 \leq j \leq \mid F \mid \\ I_{train} &= I / I_{learn} \\ \Delta &= -\frac{\mid I_{learn} \mid}{\mid I \mid} E(I_{learn}) - \frac{\mid I_{train} \mid}{\mid I \mid}(I_{train}) \\ f_j^{opt} &= max_j \Delta \end{aligned} \tag{4.24}$$

The size of F is an empirical choice. If there are too many features, the training will slow down and the risk will be over-fitting. However, if the smaller F is chosen, randomization will come about faster.

4.3 HOG Parameters Comparison

As previously mentioned in Section 4.1.1, although the study in [76]suggested the use of HOG parameters for pedestrian detection, it is hard to say whether this works in other conditions.

We discuss in this section how we found the reasonable parameters of HOG used for traffic signs detection. We elaborate upon how we exploited the HOG features for traffic signs in both our dataset and GTSRB; and how we set the size of the descriptor. After the systematic study of the effects of various parameters, we set our descriptor for the training step. The default implementation use the following parameters: a gray scale image without gamma correction; simple 1-D masks $[-1, 0, 1]$ filter; linear gradient spread over $0° - 180°$ and voting in 8 orientations; 16×16 pixel block that contains four 8×8 pixel cells with 8×8 pixel stride; $L2 - norm$ block normalization; and, a linear SVM classifier. Not all the parameters explicitly mentioned are set to the default value.

Gradient computation

The HOG descriptors depend on the methods of gradient computation. We tested four different discrete derivative masks. The following masks are tested [112].

1. Simple 1-D masks $[-1, 0, 1]$.
2. 1-D cubic-corrected masks $[-1, -8, 0, 8, 1]$.
3. Sobel filters $H_x = [1, 0, -1; 2, 0, -2; 1, 0, -1;]; H_y = [-1, -2, -1; 0, 0, 0; 1, 2, 1;]$.
4. Prewitt filters $H_x = [-1, 0, 1; -1, 0, 1; -1, 0, 1;]; H_y = -H_x^T$.

The Table 4.1 shows the results (at a SVM threshold equals 0):

Different filters	**Performance** (%)
Simple 1-D	92.38
1-D cubic corrected	91.73
Prewitt filter	91.97
Sobel filter	92.53

Table 4.1: Effect of different gradient computation on detection performance

We use Detection Error Tradeoff (DET) curves to indicate the performance of different filters for gradient computation, see Figure 4.8.

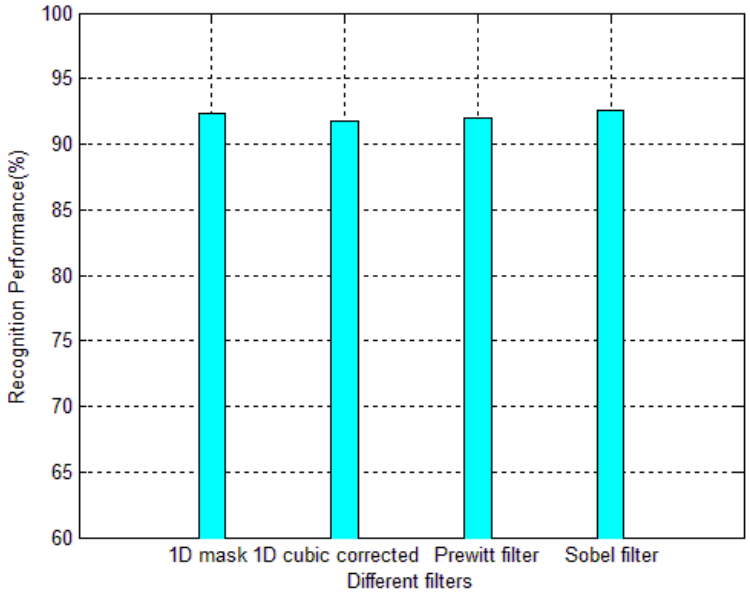


Figure 4.7: Performance of different filters

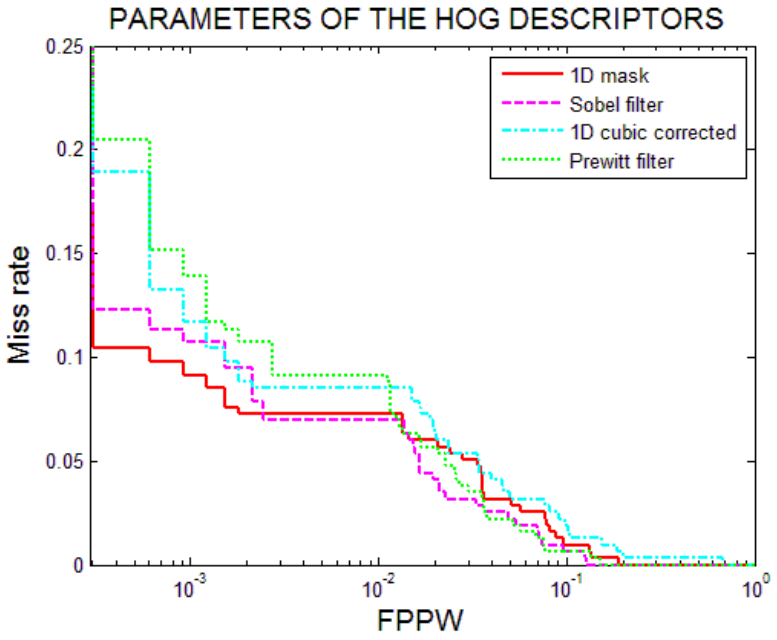


Figure 4.8: Evaluations Different filters: Miss rate vs. False Positive Per Window (FPPW) curves. Lower curves show better performance

We can easily find that simple 1-D mask performs better at $10^{-4} - 10^{-2}$ range of FPPW, but, it performs with a little bit fluctuation at a range of $10^{-2} - 10^{-1}$. Sobel filter is, however, the opposite; so, we the still choose simple 1-D mask consider the time cost and the computing complexity.

Orientation binning

Each pixel within the cell, which can be either rectangular or circular, influences the orientation histogram channel; this is based on the orientation of gradient element centered on it.

The orientation bins are spread over $0° - 180°$ ("unsigned" gradient) or $0° - 360°$ ("signed" gradient) respectively. We tested both signed and unsigned gradients under 8, 9, 18, 36 and 72 bins. The results are listed in Table 4.2.

Signed/Unsigned gradient	**Performance (%) under different orientation bins**				
	8 bins	9 bins	18 bins	36 bins	72 bins
0-360	91.80	91.93	91.29	90.97	90.77
0-180	92.36	92.49	92.22	92.27	92.34

Table 4.2: Data of different parameters of HOG orientation bins

DET curves indicate the performance of different filters for orientation, see Figure 4.10 and 4.11.

After considering the Table 4.2, Figure 4.10 and 4.11, we find that the performance of "signed" gradient is slight bad than the "unsigned" gradient as a whole. Moreover, the performance of different bins of "unsigned" gradient show little difference in our experiments. In addition, the traffic sign is different than other object; it is relatively standard in shape and pattern; we find that a few numbers of bins is enough to describe the features and increase the number of bins will not increase the accuracy of detection or recognition; but will increase the dimension of feature vectors and extraction time. Thus, based on our experiments and the experience value [76], we determined the best parameter for the situation: "unsigned" gradient, using 9 bins.

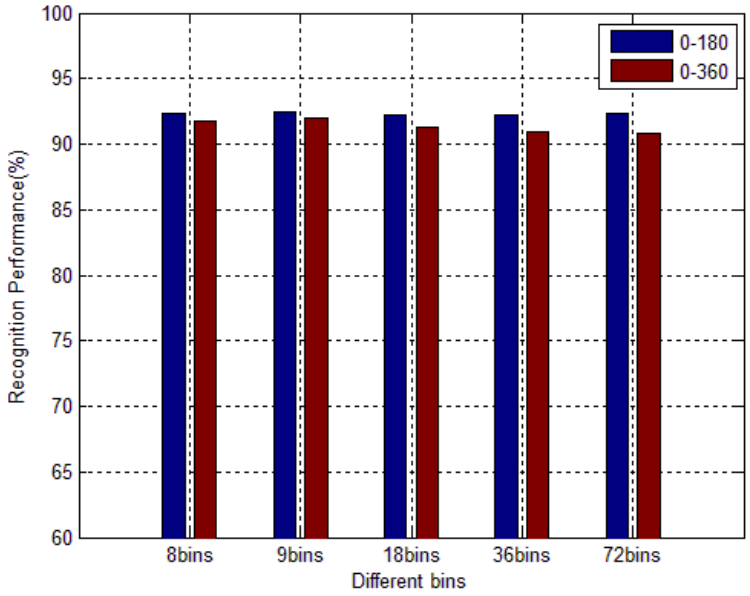


Figure 4.9: Performance of different orientation bins

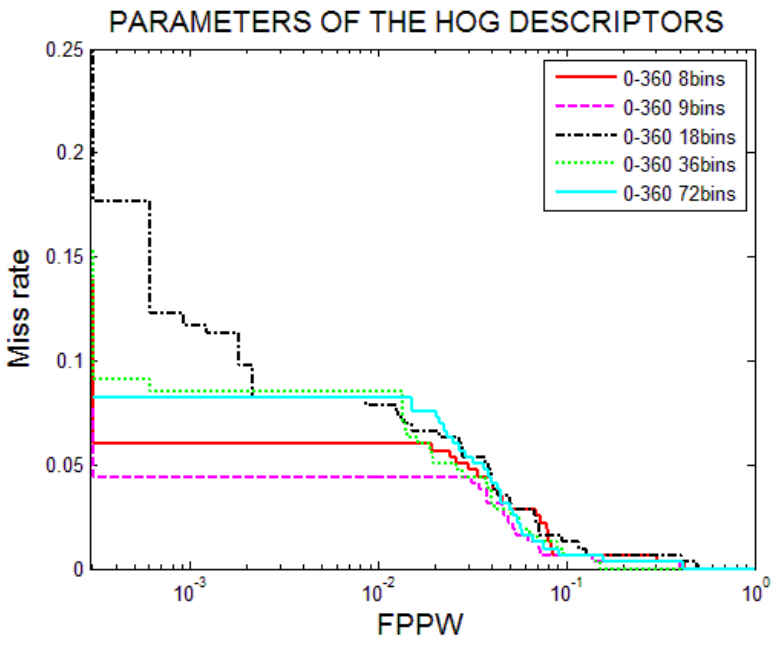


Figure 4.10: Different bins of "signed" gradient evaluation: Miss rate vs. False Positive Per Window (FPPW) curves. Lower curves show better performance.

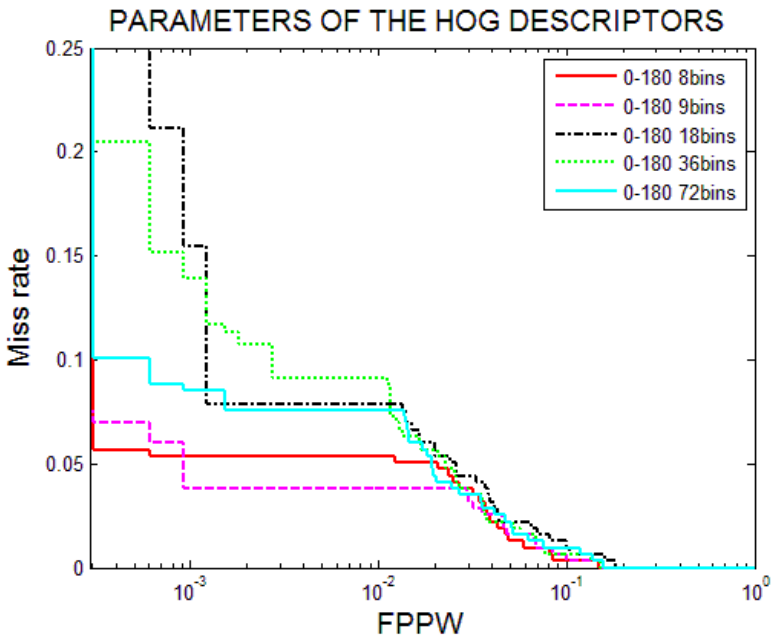


Figure 4.11: Different bins of "unsigned" gradient evaluation: Miss rate vs. False Positive Per Window (FPPW) curves. Lower curves show better performance.

Histogram normalization

The next key step is to block normalization. The norm of the vector contains the histogram contributions. Therefore, we have acquired this step of computing different methods of the HOG. The histogram of each block is then normalized with its respective value. We chose four ways of normalization (see Chapter 4.1.1 for details) in order to compare the results without the normalization results.

Norm methods	**Performance (%)**
L1 sqrt	92.45
L1 norm	91.79
L2 norm	92.33
L2 Hys	93.45
No normalization	80.32

Table 4.3: Data of different normalization methods

In [79], Dalal et al. found that $L1-sqrt$, $L2-norm$ and $L2-Hys$ perform almost

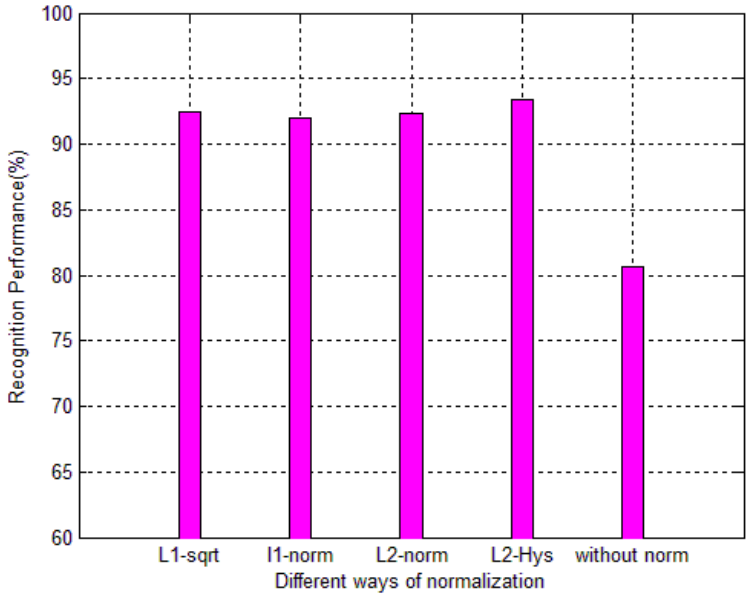


Figure 4.12: Performance of different normalization methods

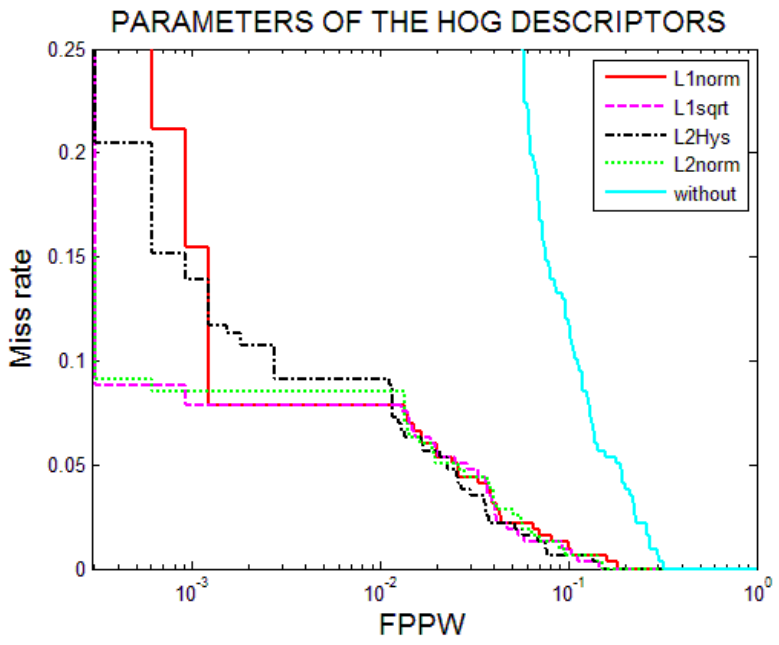


Figure 4.13: Different ways to block normalization evaluation: Miss rate vs. False Positive Per Window (FPPW) curves. Lower curves show better performance.

the same. But, the performance of $L1-norm$ is little bit lower in accuracy. In our experiments, the curves in Figure 4.13 show that, although there is no big difference in the performance of $L1-sqrt$, $L2-norm$ and $L2-Hys$, the performance of $L2-Hys$ is little bit higher based on the information from our dataset. We then chose $L2-Hys$ for in the final training block normalization process.

Block Overlap

Dalal and Triggs study reported that blocks overlapping blocks improves their performance significantly. This is because with block overlapping, each cell contributes more than once to the final descriptor.

However, when this overlap increases (see Section 4.1.1 on histogram normalization and descriptor blocks for detail), the length of the feature vector will increase and it will expend more time. Table 4.4 shows the performance obtained by varying the amount of the overlap.

Overlap (%)	**Stride (pixels)**	**Performance (%)**
0	16×16	88.33
0.5	8×8	91.93
0.75	4×4	92.29

Table 4.4: Data of different overlaps

The curves in Figure 4.15 reflect the detail accuracy.

Figure 4.15 and the Table 4.4 show that, when the stride equals pixels every time (overlap =75%), we got better performance; but, a lot of time was consumed. So, we choose the parameter of as stride equal to pixels (overlap = 50%) in the final training process.

Descriptor blocks

HOG obtains features by separating blocks and cells so that the size of blocks, cells and strides are also important factors. In this section, we demonstrate the influence of cell and block size by testing different combinations. Below, we try different sizes to find a good combination, according to HOG performance.

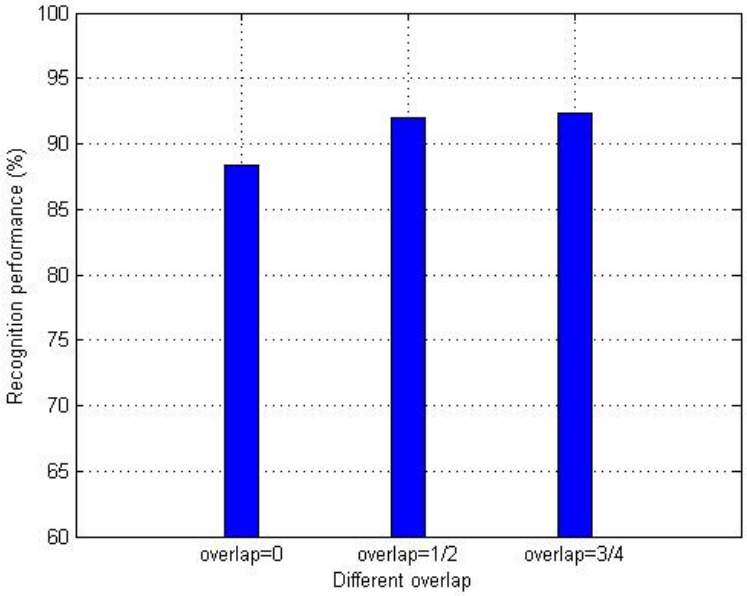


Figure 4.14: Performance of different overlaps

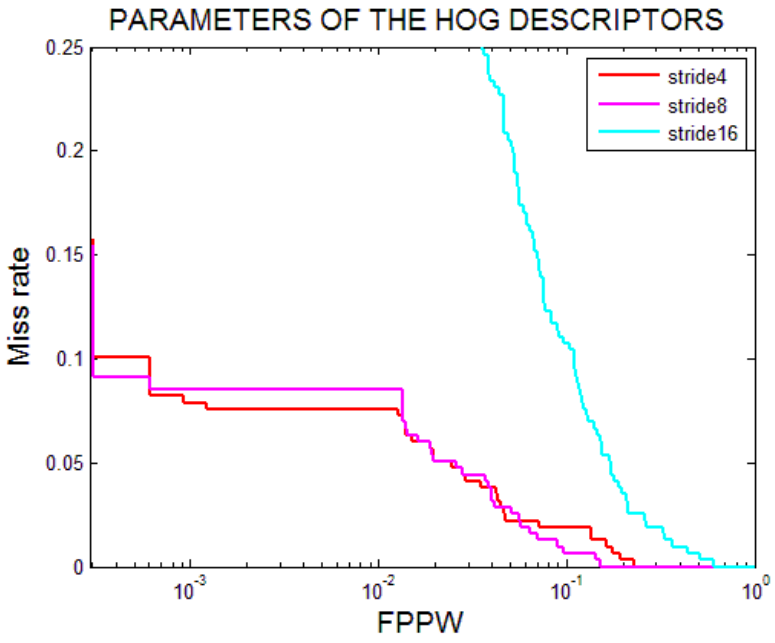


Figure 4.15: Different block overlaps evaluation: Miss rate vs. False Positive Per Window (FPPW) curves. Lower curves show better performance.

The results are shown in Table 4.4 and Figure 4.16 below (image size 40×40). The data in the following Table 4.5 shows sets of the HOG descriptor parameters:

Name	Cell (pixels)	Block (pixels)	Stride (pixels)	Bins	Overlap	Dimension	Performance (%)	Recall rate (%)
HOG1	5×5	10×10	5×5	8	1/2	1568	95.06	76.48
HOG2	5×5	10×10	5×5	9	1/2	1764	95.01	76.26
HOG3	4×4	8×8	4×4	9	1/2	2916	94.03	71.58
HOG4	4×4	8×8	4×4	8	1/2	2592	93.76	70.32
HOG5	8×8	16×16	8×8	9	1/2	576	96.93	85.39
HOG6	8×8	16×16	8×8	8	1/2	512	95.80	80.48

Table 4.5: Data of different sizes of HOG cells, blocks, etc.

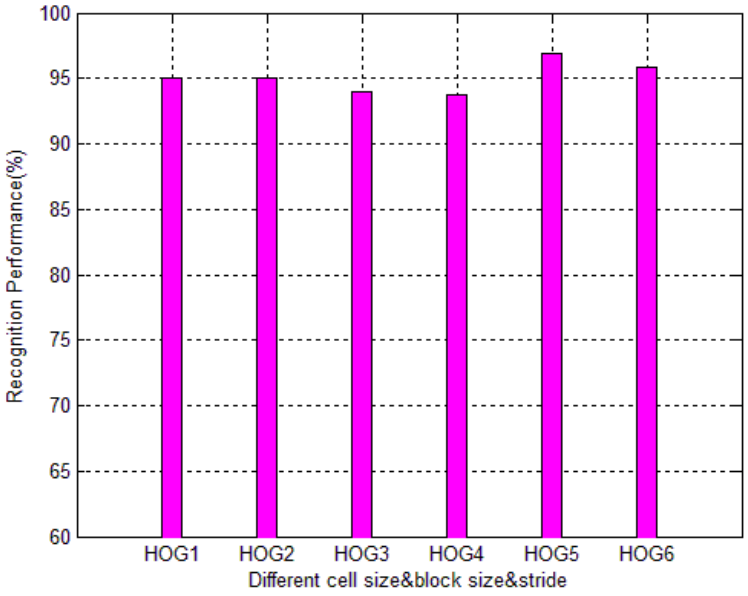


Figure 4.16: Performance of different sizes of HOG cells, blocks, etc.

We can obviously find that HOG 5 has a better performance than other combination in this test. And, we set the block and cell size HOG 5 for our dataset training.

The data above demonstrates that, by fine-tuning the HOG parameters, it is possible to improve the performance considerably for a specific problem. Of course, the performance

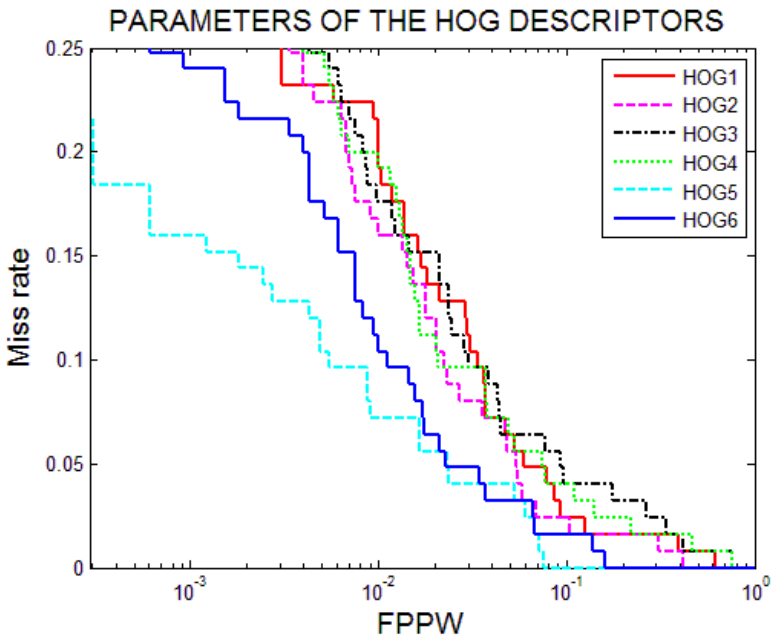


Figure 4.17: Different size of block and cell combination evaluation: Miss rate vs. False Positive Per Window (FPPW) curves. Lower curves show better performance

is based on our dataset, as there is no specific parameter that fits all objects. The only way to find the right parameter is by performing experiments to get a balanced consideration of performance and the time cost incurred when building a descriptor. Different parameters will give different results. Therefore, an extensive study of HOG parameter performance is recommended for any given new problem.

4.3.1 Building an Effective Weight Vector

In [76], the training step consists of using HOG to get the pedestrian feature. The dataset contains positive pedestrian samples and negative non-pedestrian samples. A kind of cell and block-based HOG feature is extracted to describe each image sample; and, according to the author of the article [76], the descriptor has a high dimension (3780 dimensions in a 64×128 window). This may be equated to that appears in Equation 4.11. The SVM algorithm aims to find vector w and threshold b with the minimum value of Equation 4.21.

After the description of HOG and SVM, additional properties of these two algorithms are given as follows. In the case of HOG:

1. HOG focuses on the grid unit, which makes the descriptor subtler and shows superior performance in object detection. HOG maintains invariance of image geometry and optical deformation.

2. In the first paper [76], some subtle human body movements occurred, which would be ignored by HOG.

3. Aside from pedestrians and animals, our traffic signs will not move independently. The characteristics will be more or less stable; thus, traffic signs will then be effectively represented by HOG descriptors.

SVM will provide accurate classifiers and will be noise tolerant. It also can solve high-dimensional problems. This is the reason for why HOG+SVM are used together in the whole system. The rational for this is that since the feature descriptor built by HOG will be a high-dimensional feature, see Figure 4.18

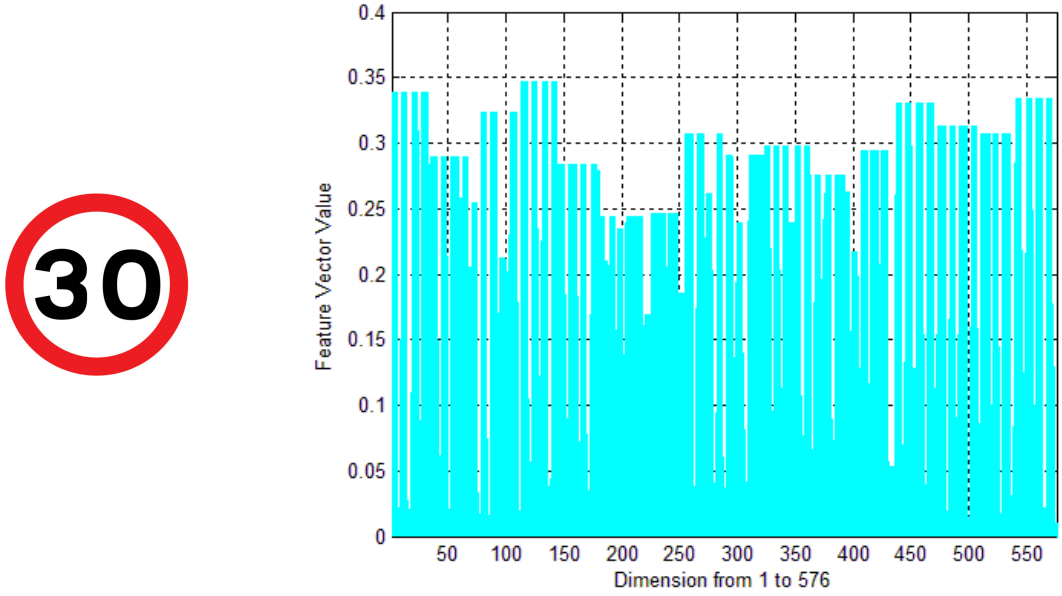


Figure 4.18: Example of final weighted vector for detection

As long as we have the dataset and parameter settings prepared, the image information will be changed into data information so that the computer can learn them.

The first step is to resize the extracted versions of the image into the same size. Because the image is not the same size when it is taken from the different frames, we resized it to 40×40 pixels. This is the suitable size for calculating and training time consumption.

For this experiment, we chose R-HOG block (see Chapter 4.1.1 for details) for our HOG feature extraction. The other parameters for extraction we needed to decide on were:

- Number of unit cells in each block;
- Number of pixels in each cell unit;
- Number of histogram directions for each cell.

Training information demonstrate in Table 4.18.

Size of training images	**Orientation bins**	**Cell (pixels)**	**Block (cells)**	**Dimension**
40×40	9 bins	8×8	2×2	$4 \times 4 \times (2 \times 2 \times 9) = 576$

Table 4.6: Training information

We wrote our program for feature exaction, for the HOG algorithm, using the parameter we found in the previous section. The command parameters for the program are:

- argv[1]: directory of positive images.
- argv[2]: positive images name list file.
- argv[3]: directory of negative images.
- argv[4]: negative images name list file.
- argv[5]: file to store feature vector.
- argv[6]: number of random patches for negative images, default value:10.
- argv[7]: width of images.
- argv[8]: height of images.

The introduction of SVM is already provided in Section 4.3.1. The first training data is then learned by the SVM in order to build classifiers. We used SVM_light [113]and [50], an

optimized version of SVM, to generate the classifiers based on the feature vectors extracted from HOG.

SVM_light contains two modules. SVM_learn is used to construct the hyper plane on the input data. SVM_classify, however, is used to generate the predictions based on the learned classifier.

```
SVM-light Version V6.02
0 # kernel type
3 # kernel parameter -d
1 # kernel parameter -g
1 # kernel parameter -s
1 # kernel parameter -r
empty# kernel parameter -u
576 # highest feature index
3420 # number of training documents
35 # number of support vectors plus 1
1.0927845 # threshold b, each following line is a SV (starting with alpha*y)
```

Figure 4.19: Model of training data after SVM_learn

The final weighted vector used for detection still needs to undergo another calculation step; this computes the weight vector of linear SVM based on the model file. So far, the weighted vector used for detection has been created successfully. This weighted vector is used in detection programs as a command parameter.

4.4 Different Architecture Design for TSDR systems

In this section, we discuss the different ways to detect traffic signs, based on the training stage that was previously discussed. Since we have already chosen the parameters of HOG, we will now set the HOG descriptor.

4.4.1 Two Stages of HOG+SVM Architecture

In our two stages of HOG+SVM architecture, we built different classifiers in order to extract the ROIs and classify different signs for each frame of input videos. The architecture is showed in Figure 4.20.

In Figure 4.20, we first input different testing videos or images, and then built different classifiers for different purposes. The overall classifier is aim to extract the ROIs of each

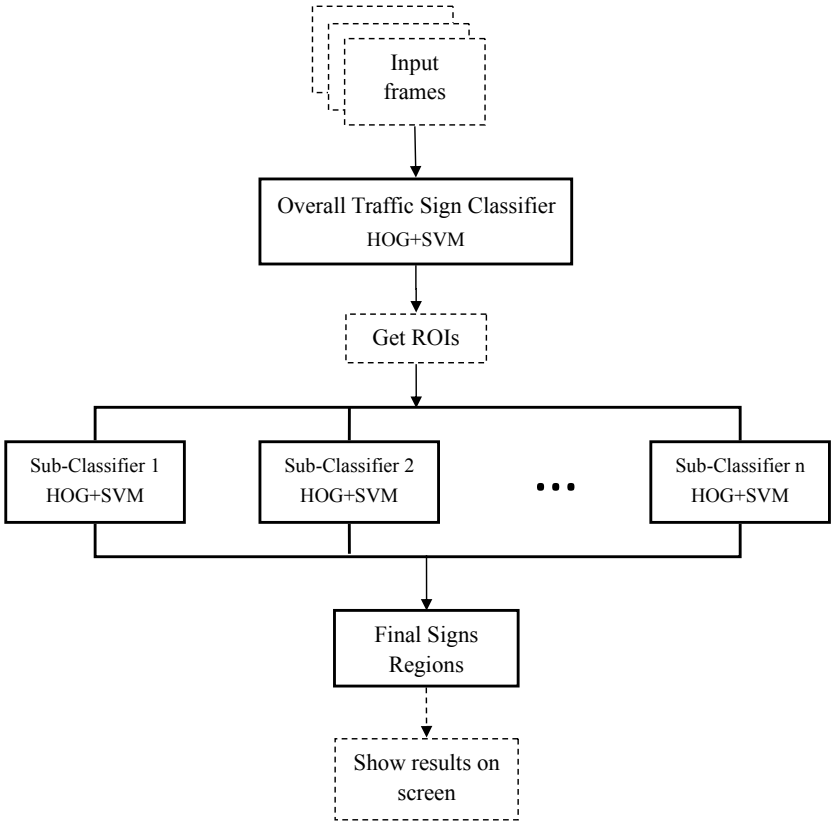


Figure 4.20: Flow chart of HOG+SVM design

frame; and after we get the ROIs, we use small classifiers of different signs for each ROI in order to decide the content of each ROI. During this process, we finally get the recognition results of each ROI and show the correct information on the screen.

When we built the overall classifier, we first trained and tested our vision model on the selected GTSRB dataset [114] and [115] that contains 19,513 training images and 12,042 test images of traffic signs. The positive training dataset was resized to 50×50 pixels. But, the searching window has a size of 40×40 pixels to ensure that the center area of 40×40 pixels is used to compute the features vector. This is why, when we created the dataset, we needed to leave a margin around the traffic sign image and to resize to be slightly larger than 40×40 pixels.

In training, one strategy we used was to train separate classifiers for different signs. Since we needed to recognize each of the signs, the input data had to consist of the overall traffic sign classifiers and every kind of specific traffic sign classifier.

The image of the training part for overall classifiers should be [116]:

- Positive image: all selected classes in the training dataset.
- Negative image: several images of the size 40×40 pixels randomly cut from the dataset and from video clips capturing information along the road.

The input images used to train the second level of classification (for the specific traffic sign classifier) are:

- Positive image: specific signs to be detected.
- Negative image: all other signs and several image patches randomly cut from each image in the dataset and from video clips capturing information along the road.

In the second layer of the classifier, we included all other selected signs as negative samples in order to distinguish traffic sign we wanted to classify from others.

After the descriptor vector for the image was built, the detector was needed. We set the detector window size to 40×40 pixels. When put a test image in this system, a sliding window of 40×40 pixels scrolled over the whole image with 8×8 step sizes; and, a classifier was run for each cropped window.

Comparing SVM with Random Forest Classifiers

Random Forest has many properties reflecting a variety of materials; these properties can produce high-accuracy classifiers. They can handle a large number of input variables. And they can assess the importance of variables while determining the categories. In forest construction, RF can generate no-deviation estimates for internal normalization error.

According to the characteristic of RF, some researchers [117] use this method to classify different traffic signs based on HOG features and the German dataset; this means that RF can be used instead of SVM.

However, the disadvantage of RF is that they need a lot of samples during the training phase. Even though our dataset is updated daily, there is still a difference with the German one. Therefore, the RF method may not suit our needs very well. In any case, the learning element of RF operates more slowly than what we have done with SVM.

RF also has many parameters described in OPENCV function:

```
CvRTParams(
    int max_depth, int min_sample_count, float regression_accuracy,
    bool use_surrogates, int max_categories, const float* priors,
    bool calc_var_importance, int nactive_vars,
    int max_num_of_trees_in_the_forest,
    float forest_accuracy, int termcrit_type
)
```

The initializing parameter of each Random Tree (RT) in RF is as follows:

```
CvRTParams (
    10, 10, 0, false, 15, 0, true, 4, 100, 0.01f, CV_TERMCRIT_ITE
)
```

The value of nactive_vars is an important value that, if changed, would change the accuracy of the test for the whole system. We chose a part number from our dataset and compared it to the parameters of RF to get the best classification rate. See Figure 4.21. The nactive_vars values in experiment equal 4, 8, 10, 20, 25, 50, 51, 53, 60, 100.

According to Figure 4.21, we set the parameters of RF as:

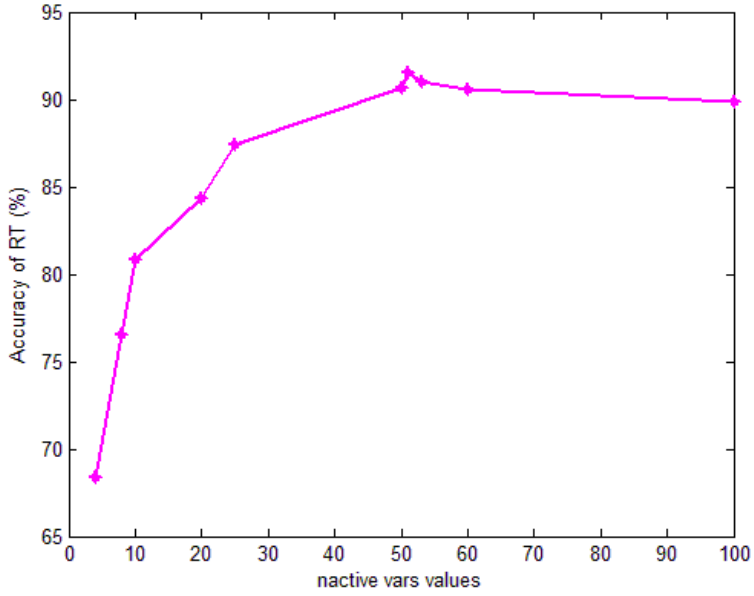


Figure 4.21: Accuracy of RT under different nactive_vars

```
CvRTParams (
    10, 10, 0, false, 15, 0, true, 50, 100, 0.1f, CV_TERMCRIT_EPS
)
```

Datasets	**Sample num**	**Training num**	**Test num**	**Accuracy of RF (%)**	**Accuracy of SVM (%)**
GTSRB	13596	10197	3399	98.7	96.9
Our dataset	4700	3525	1175	92.5	94.1

Table 4.7: Compare data between RF and SVM in different datasets

The Table 4.7 show that when we choose same quantity of sample number and same ratio of training number and test number of images (Training number = 75%× Sample number; Test number = 25%× Sample number) for both RF and SVM classifiers; the accuracy of RF method achieves about 1.6% less than the SVM for our dataset. When the number of sample number is increased (GTSRB); the results of RF will be better than SVM [111]. This situation reflects the characteristic of RF: it needs a large training number

image when build the classifiers. However, because of the sample number in our dataset, we did not choose RF for the following step.

4.4.2 Color Information Extraction Architecture

As we know, in NASLS, there are some speed limit signs that have different meanings, while the number carried out by the sign remains the same. From the first step of HOG, the training images were changed to gray scale images; thus, when we conducted our experiments, the feature vector remained the same (see Figure 4.22).

Figure 4.22: Different example signs (original image and grayscale) of different color backgrounds

In our findings, there will be some missed recognition. For example, the advisory yellow signs will be considered the same as the white one. The misrecognized signs still give a right number of messages for the signs. This is because they have the same speed limit number but the road condition has been ignored. In any case, we are more interested in finding out about the road conditions when driving. Working zone orange signs provide us with this information. When there is an orange sign, this means there will be some construction on the road ahead.

Therefore, it is necessary to separate the signs that have the same number and color, but according to different background colors. The signs with black backgrounds will provide training as usual, since the feature vectors of black and white ones are totally different.

We considered adding the color segmentation at the middle of the two stages before the original step of recognition where we did the color judgement [103]. The purpose of this is to separate the different backgrounds before doing the next part. The structure for this is shown in Figure 4.23.

The strategy of HOG + SVM is the same as the original one. However, after the ROIs are carried out, we extracted the middle part called "ROIc (colored Region Of Interest)."

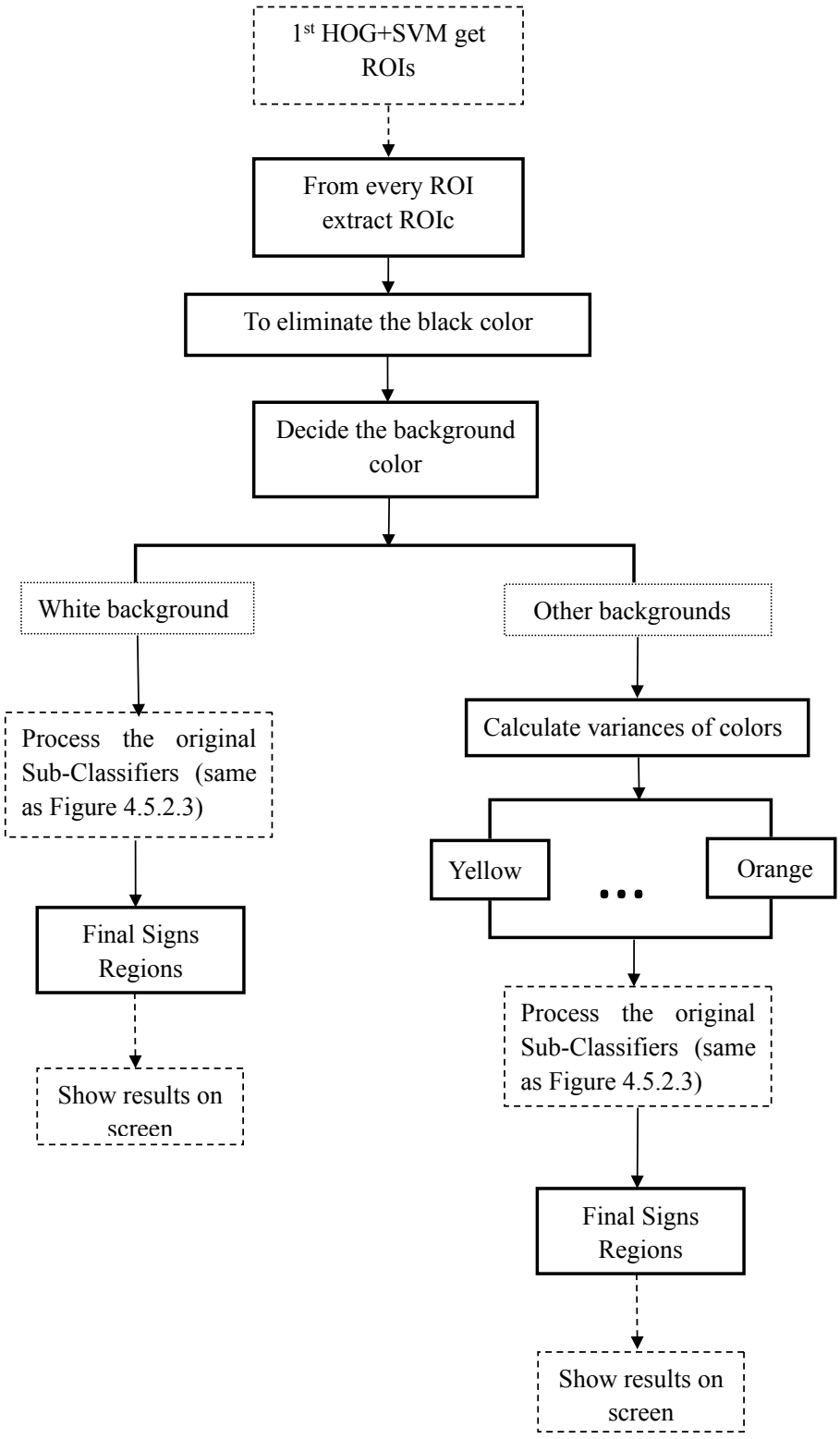


Figure 4.23: Flow chart of color information added architecture design

This could have contained black numbers and the desired unknown color of the sign. Our aim is to eliminate the black numbers in order to avoid their influence on the effect of the color of the rest of ROIc.

In order to eliminate black numbers from the ROIc, because the RGB value of black pixels is (0, 0, 0), we decided to calculate the pixel value of the average channels for the whole ROIc. If the calculated average is close to 0, we assumed the pixel was black. If not, we needed to count the number of non-black values and to calculate the sum of the three channels. If the average of the pixel value of these three is more than 240-250 (RGB value of each channel for white pixels is (255, 255, 255)), we could decide that the background color is white; we could thus process the original recognition step. Otherwise, since the colors of the background in our dataset are yellow and orange, they are almost the same: standard yellow (255, 255, 0) and standard orange (255, 69, 0). In order to separate them clearly, we have to do the statistic for the average value of three channels for all yellow signs and orange signs; this is called the experience values (Er, Eg and Eb). And, in our case, we must then compute the 2 variances of average RGB value (A_r, A_g and A_b) of each unknown colored pixel towards experience values use following Equation 4.25.

$$\begin{aligned}
&A_r = (r_1 + r_2 + ... + r_n)/n \\
&A_g = (g_1 + g_2 + ... + g_n)/n \\
&A_b = (b_1 + b_2 + ... + b_n)/n \\
&(n \quad is \quad the \quad number \quad of \quad pixels \quad in \quad ROIc) \\
&S1^2 = ((A_r - Er_1)^2 + (A_g - Eg_1)^2 + (Ab - Eb_1)^2)/3 \\
&S2^2 = ((A_r - Er_2)^2 + (A_g - Eg_2)^2 + (Ab - Eb_2)^2)/3
\end{aligned} \tag{4.25}$$

And then, compare these 2 variances, if $S1^2 < S2^2$, the background is the first kind of color, Otherwise it should be another one. In our dataset this time, there are only two easy-confused colors: yellow and orange. But if the number of colored backgrounds increased, Equation 4.25 still works. We need to calculate n kinds of variances and choose the minimum variance; the corresponding color of this minimum variance is the correct color background. After this, we moved onto the recognition step. The structure of the color information analysis, which aims to fit our situation, is described in Figure 4.24

Therefore, different color backgrounds that have the same feature vectors are separated

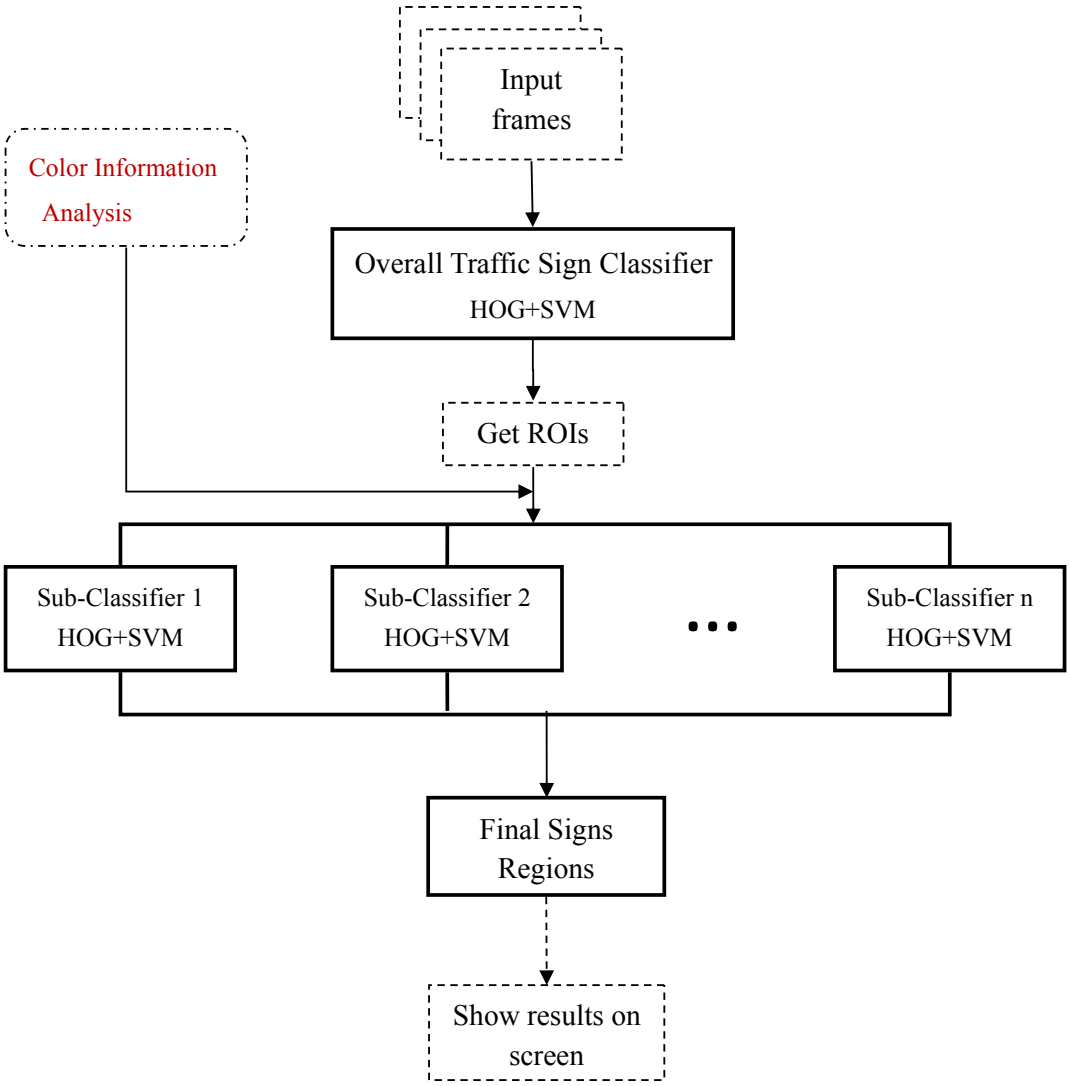


Figure 4.24: Flow chart of color information extraction design

clearly. The color information extraction algorithm can be used for the area in which traffic signs have the same situation as in Canada.

The results are shown in Figure 4.25.

The results in above Figures show that our system can work under different weather conditions; and, when this color extraction part added in, the signs on different backgrounds can be separated clearly.

4.4.3 MSER Improvement Architecture

As discussed in Section 4.2.2, MSER is a useful method for color segmentation, considering how time-consuming the traditional method of HOG can be. Therefore, we have considered MSER instead of the first step of the basic methods because it can find different pieces of stable regions according to different thresholds [118] to increase the processing time.

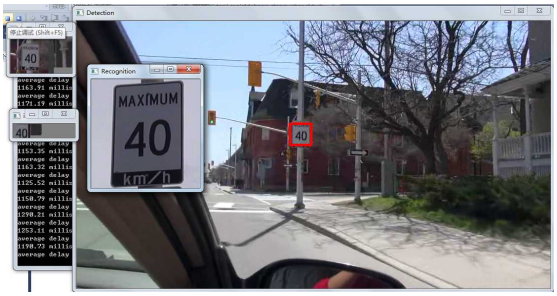
The MSER can be used instead of the first HOG step to get ROIs. Due to the nature of traffic signs characteristics, the region of signs is always a stable region for color or shape. So, retrieving ROIs with the MSER method proves to be much faster.

In an actual video frame, there are many signs with red or blue backgrounds. We need to segment them out and to separate the regions from their backgrounds. When we get the frames from different videos, at first, we need to transform them from RGB into a "normalized R/B" image, as mentioned in [119]

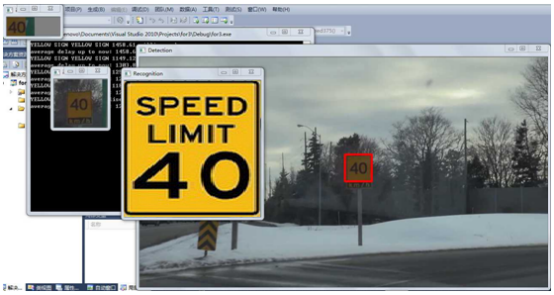
The value of each pixel in the frame needs to be reassigned based on the ratio of the blue channel to the sum of all channels and according to the ratio of the red channel to the sum of all channels. The greater of these two values is used as the pixel value to the normalize R/B image Ω_{RB}:

$$\Omega_{RB} = \max(\frac{R}{R+B+G}, \frac{B}{R+B+G}) \tag{4.26}$$

Pixel values in the frame are higher in the red or blue channel; but, they are lower in other channels. MSERs are then easily found in this frame. MSER offers a robust form of segmented ROIs for traffic signs in complex scenes. These can be computationally expensive. However, they increase the speed compared to the original process using HOG features. This will be introduced in the following section.



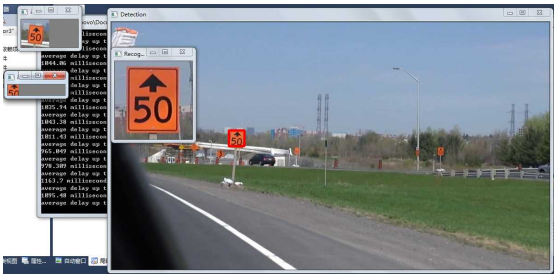
(a) Sunny weather



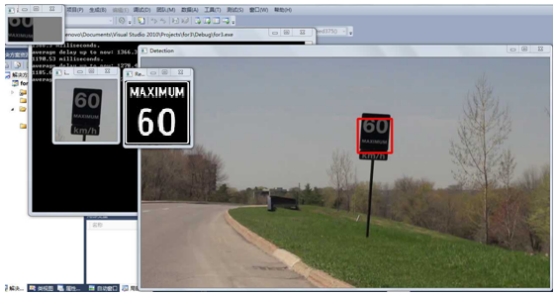
(b) Snowy weather

(c) Sunny weather

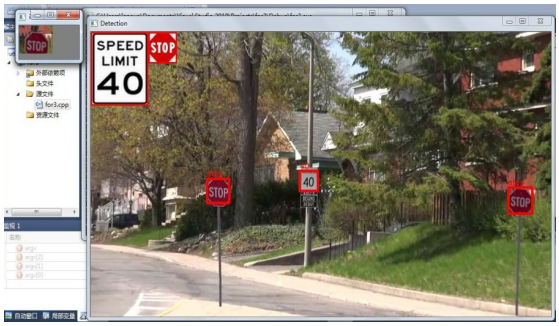
(d) Normal weather



(e) Cloudy weather

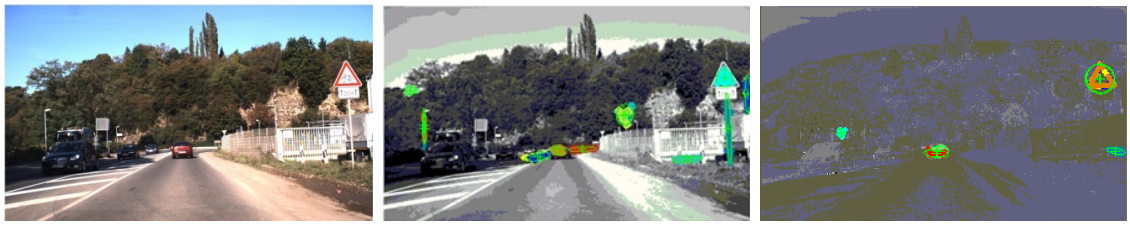


(f) Somber weather



(g) Sunny weather

Figure 4.25: Results of system testing under different weather conditions



(a) Original image (b) MSER method focus on gray image (c) MSER method focus on normalized R/B image

Figure 4.26: Experiment results under MSER analysis to extract and mark ROIs of a scene

When the signs have a white background, MSER uses a grayscale image. The number of region candidates depends on the thresholds of MSER; and, the threshold is set based on the condition of the weather.

Here is the experiment for the extraction of ROIs from MSER used in the following system.

For signs with different background colors, the color segmentation step should also be added after MSER and before recognition. We did not need to separate white and other colors. This is because the threshold of MSER, which is calculated for white and other colors, is different. So, we deleted the first step in the color segmentation process and keep the other steps. In this way, since the recognition part does not change, the accuracy of recognition will not be changed. The structure is represented in Figure 4.27:

The result from MSER is shown in Figure 4.28.

From the Figures above, we can find that the system which replaced HOG by MSER in first step can also reach our goals.

4.5 Data Analysis

In this section, we analyze the accuracy of recognition in Section 4.5.

The accuracy rate achieved when adding the color information extraction algorithm is showed in Table 4.8 and Figure 4.29. (This statistical data is based on the dataset we created).

Selected examples of curves illustrating the accuracy level are as Figure 4.30.

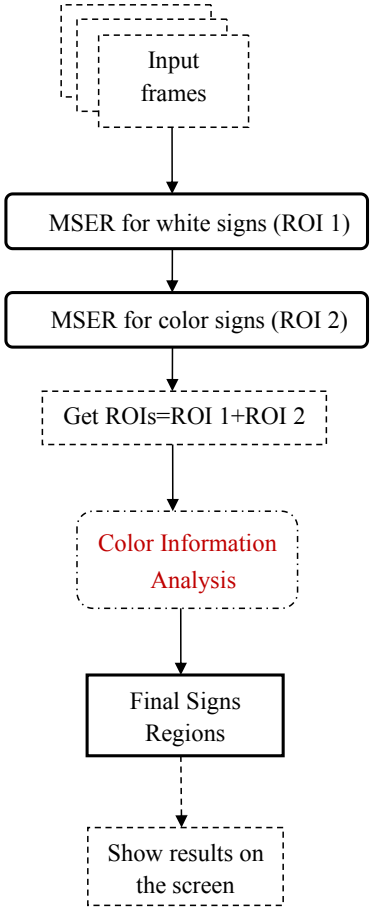


Figure 4.27: Flow chart using MSER

Categ.	**Class**	**Dataset**	**Recog.**	**incorrect**	**Recog.(%)**
White	40	6069	5935	134	97.79
	50	4962	4688	274	94.48
Yellow	40	2933	2876	57	98.06
	50	2877	2808	69	97.60
Black	60	1349	1339	10	99.26
Orange	50	1248	1210	38	96.95

Table 4.8: Data for recognition rate of different colored background signs

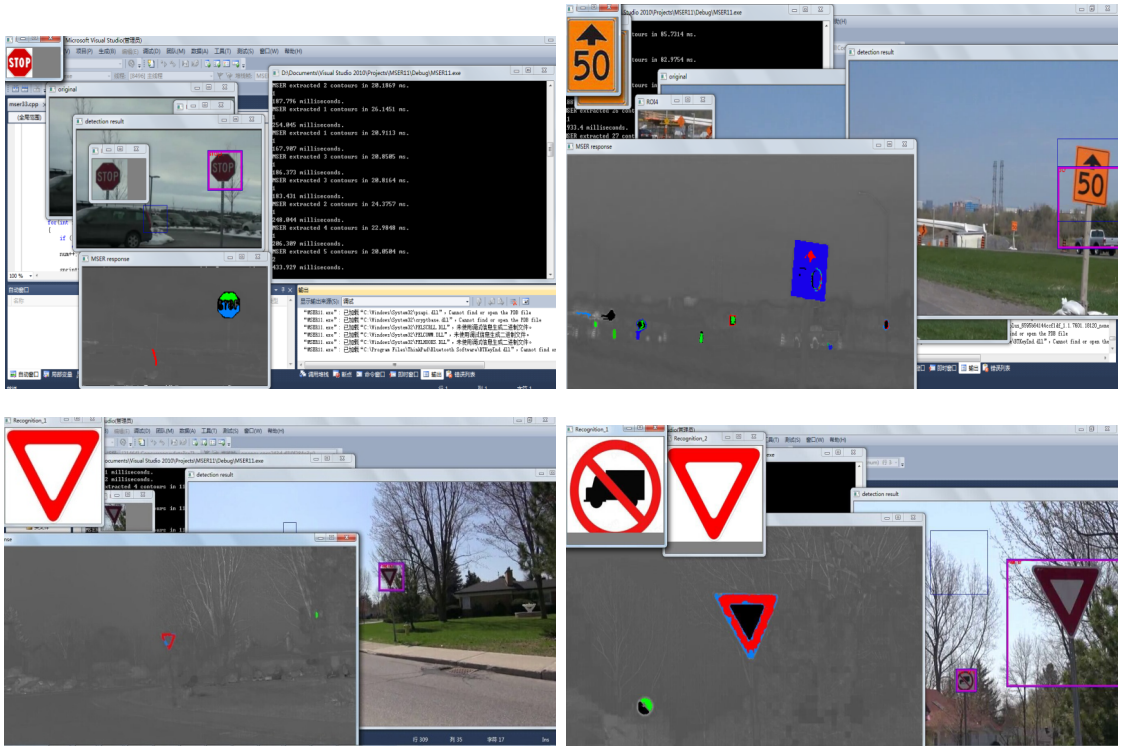


Figure 4.28: Results of MSER testing

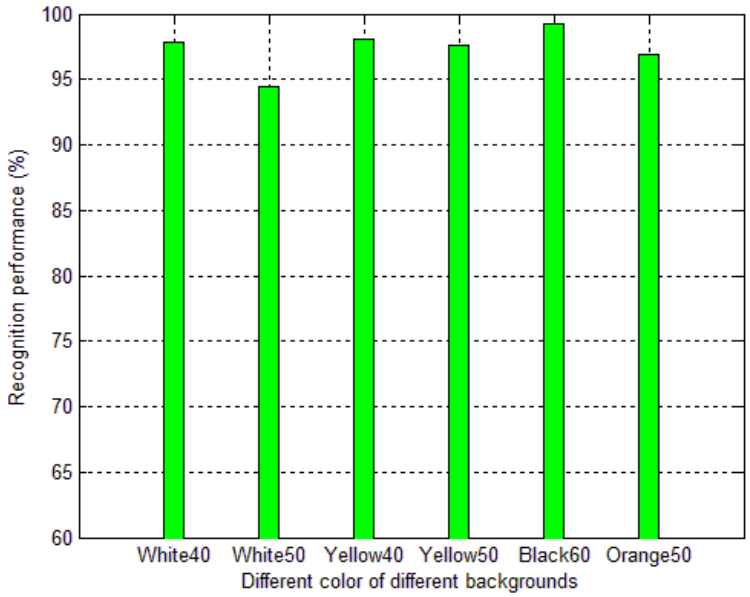


Figure 4.29: Performance of color information extraction system

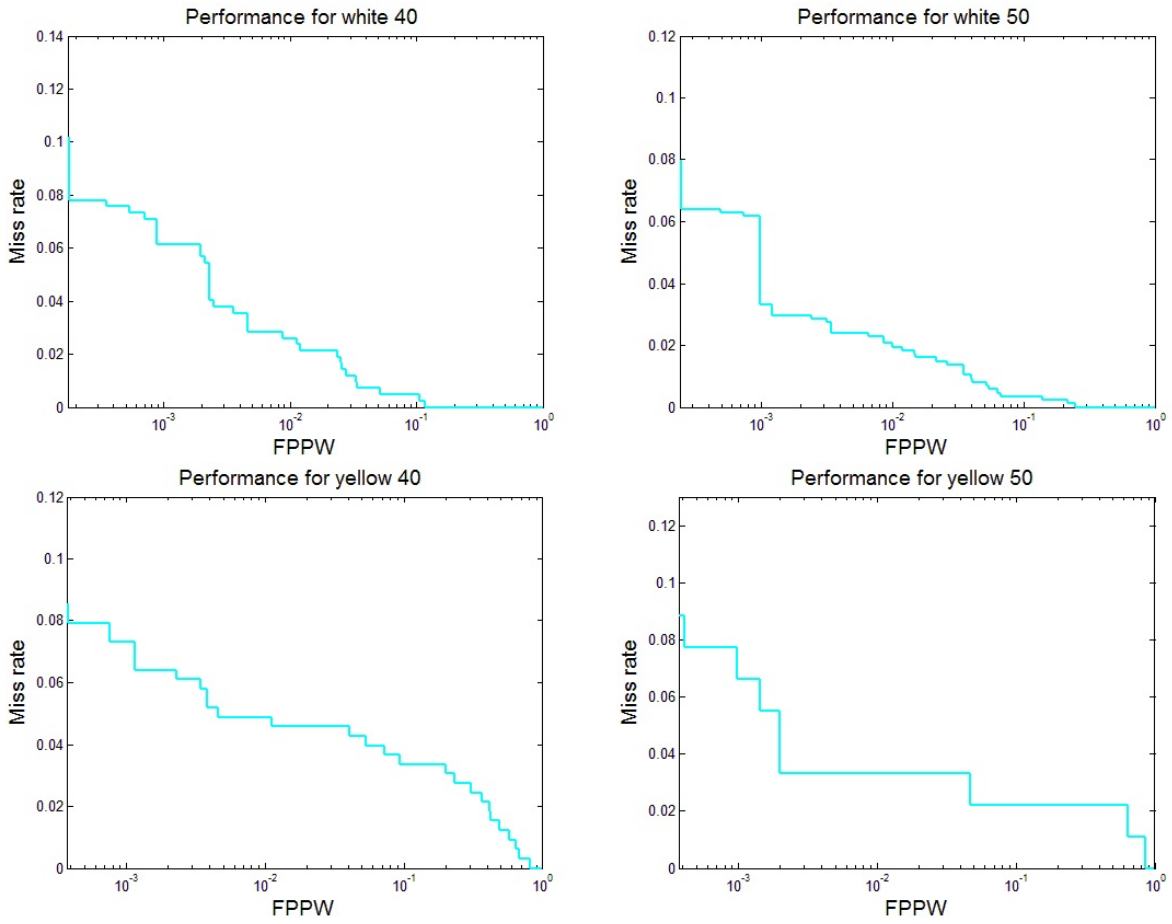


Figure 4.30: Evaluation of performance for different signs recognition: Miss rate vs. False Positive Per Window (FPPW) curves. Lower curves show better performance

Since the experiment is based on our created dataset, the accuracy has an osculation relation to the number of testing frames and training images. The bigger the dataset is and the more videos we have, the more real the results will be.

MSER method is compared with HOG in first stage (obtain of ROIs); since they have the same second stage in our experiments. There are two aspects for our comparison: the accuracy of obtaining ROIs and time-consuming. The condition using the HOG and MSER algorithms is: same day-time videos are recorded by Sony Full HD 1920×1080 pixels camera (video resized to 432×240 pixels)

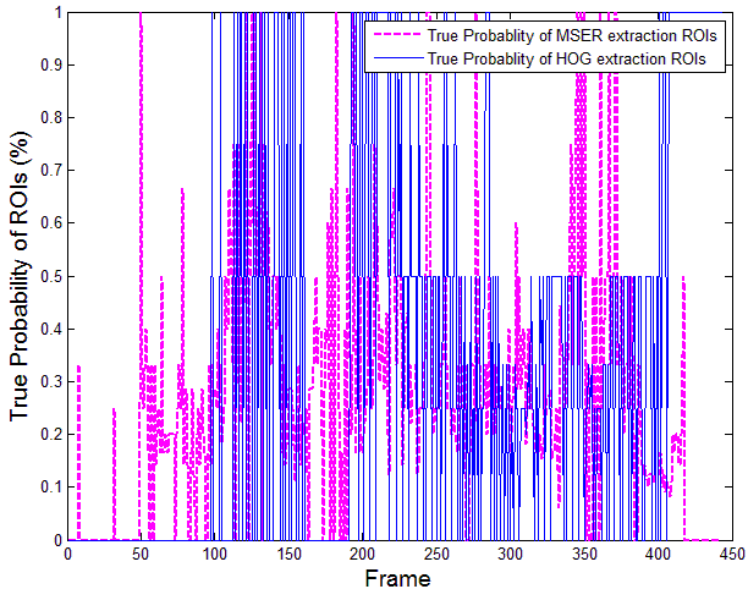


Figure 4.31: True Probability of ROIs extraction in different methods

Figure 4.31 shows the True Probability of ROIs, which means, when these two methods extraction first among of ROIs, the probability of ROIs that truly contains sign (s). We record the numbers of ROIs extraction by these two methods and the numbers of the true signs exactly in those ROIs manually, aim to avoid the effect of the second stage. Then, calculate the True Probability of those ROIs.

Figure 4.32 shows the average True Probability of ROIs extracted by MSER and HOG methods in first stage. We can see that the True Probability of ROIs used HOG is almost 8% higher than MSER. This is because HOG is based on objects features while MSER

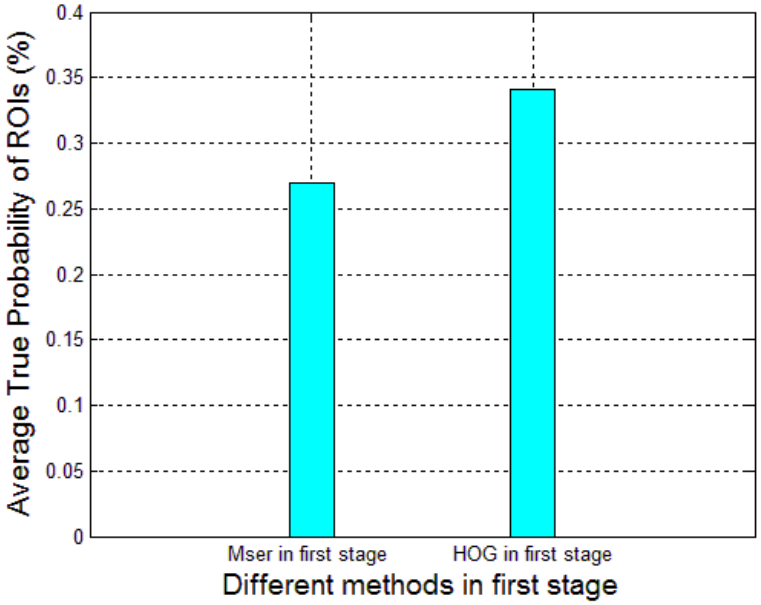


Figure 4.32: Average True Probability of ROIs of differnet methods in first extraction stage

only based on maximum stable regions. Thus, MSER always extracts more the ROIs in first stage than HOG dose, and the more ROIs make the recognition stage analysis harder; and increase the false probability.

But, in another aspect, we compare the time-consuming of these two methods. Figure 4.33 and 4.34 shows the example processing time of HOG and MSER used in first stage of systems for the same test videos.

Figure 4.35 show the extraction process where HOG and then MSER obtain the first ROIs. The time cost of HOG is about 8 times longer than that of the MSER method in each frame, when it extracts first ROIs.

Note that, in Figure 4.36, we can get the total time cost of these two methods. The second stage for HOG and MSER is the same. But, in the HOG method, the time cost of the second stage is less than that of the MSER method. Since HOG is based on objects features, while MSER only based on the stable region, the number of ROIs in the first stage extracted by MSER is more than that which is extracted by HOG. This therefore makes the second stage a little bit slower in MSER method. But in terms of the overview,

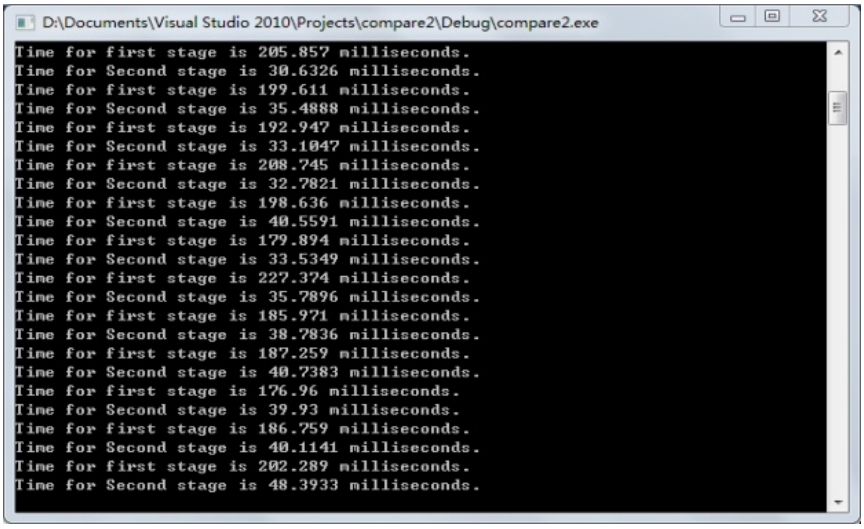


Figure 4.33: Data of video experiments based on HOG first ROIs extraction

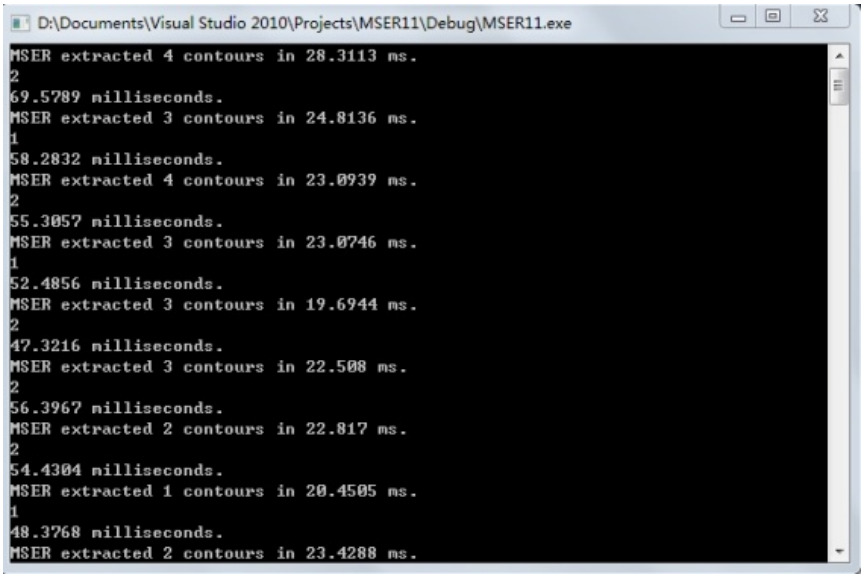


Figure 4.34: Data of video experiments based on MSER get first ROIs extraction

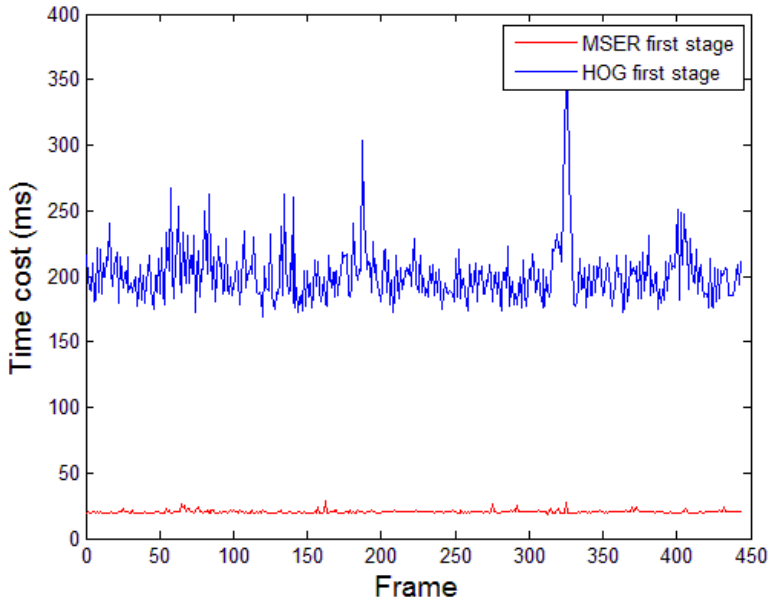


Figure 4.35: Time cost of HOG vs. MSER first extraction of ROIs

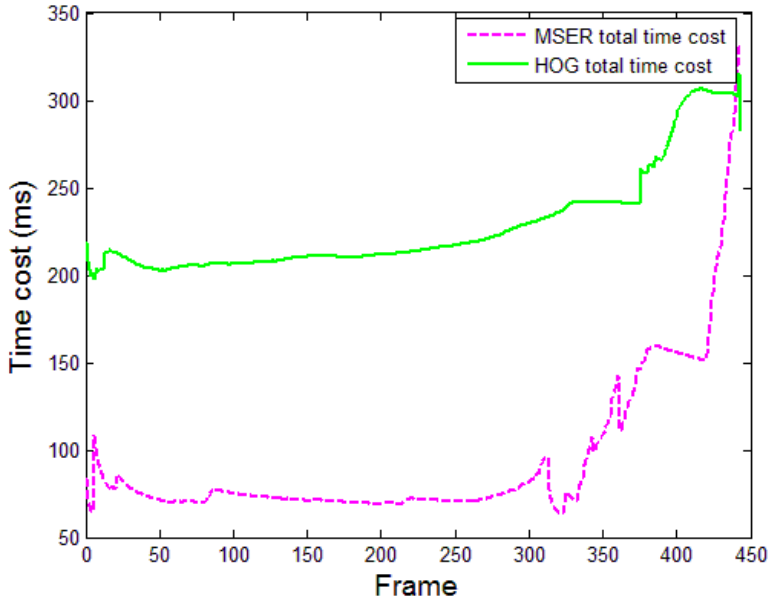


Figure 4.36: HOG vs MSER for total time cost for whole system

the algorithm using MSER is faster than the algorithm which uses only HOG.

Average time delay for the two methods showed in Figure 4.37

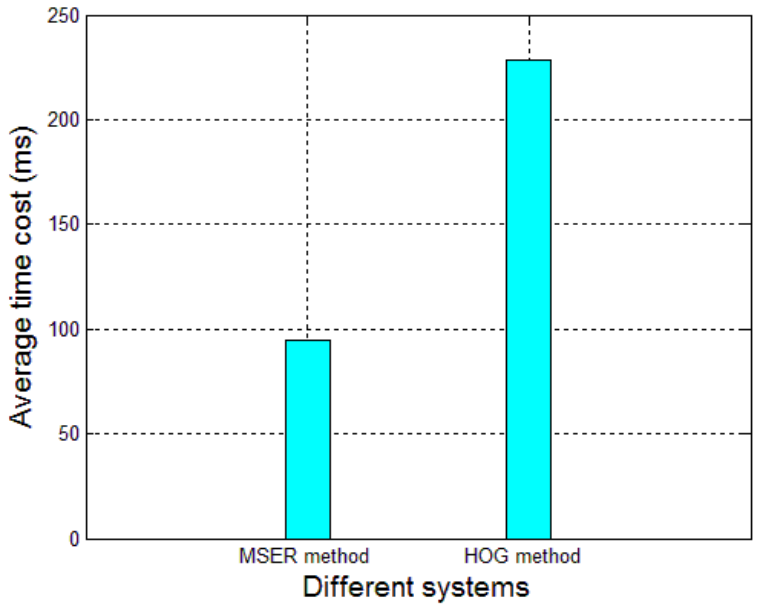


Figure 4.37: Comparison of the average time cost for the two methods

Chapter 5

Traffic Signs Detection and Recognition without Cameras

5.1 Introduction

In previous chapters, different image processing methods concerning on how to detect and recognize traffic signs through videos are described. But when performing tasks like training or searching in the whole frame, a huge deal of computing in CPU will definitely be needed, and, the speed of detection will not be guaranteed.

Thus, in this chapter we introduce a new method to find signs in the video frame. Characteristics of traffic signs do not change frequently; and, the position of the sign is always stable. So we consider using a large dataset to store the position of traffic signs in one or more areas. We store the street name, longitude and latitude coordinates of each sign. When minor changes happen, we only need to update our dataset. Moreover, the number of positions of signs and other characteristics are dramatically huge. We can not only search the data quickly with a single device on a vehicle but also store data using limited space. This situation needs a server for storage needs.

5.2 Algorithm of Computing the Distances on the Earth

The data of signs in our server is found in pairs of coordinates (longitude, latitude) plus road information. The system will return the information of signs nearby, and some other characteristics such as the direction of signs according to vehicle coordinates. When drivers obtain the information of sign coordinates, our system calculate the distance between the sign and the vehicle. When this distance is smaller than a given threshold, a warning message is generated to remind the driver about the encountered sign.

In the next paragraph, we discuss how to calculate the distance between two points by using the real geodetic coordinates of them. In our case, the two points are the position of the vehicle and the position of the sign.

We find that there are 3 ways to calculate the distance between two points [5]:

1. Rectangular plane coordinate [5]: The simplest way to calculate the distance between two points is to put the coordinate of two points in a rectangular plane, ignoring Earths surface. If $A(X1, Y1)$ and $B(X2, Y2)$, $X1$ and $X2$ are latitudes and $Y1$ and $Y2$ are longitude, the distance between A and B is:

$$D = \sqrt{[(x1 - x2)^2 + (y1 - y2)^2]} \tag{5.1}$$

   Although this method is fast enough, it has poor accuracy: it only has a 0.1 level of precision. Therefore, we will not choose this method to calculate our distance between vehicles and signs.

2. Spherical distance algorithm [5]: In this case, the curvature of the earth is generally not directly considered; we treat the earth as a positive sphere for operation. The characteristics of the sphere are known. The shortest distance between two points on the sphere is the length of the inferior arc on a great circle, determined by these two points and the center point of the sphere.

   The algorithm can be described as follows [5]:

   Suppose there are two points $A(X1, Y1)$ and $B(X2, Y2)$; $X1$ and $X2$ are latitude and $Y1$ and $Y2$ are longitudes. In order to seek the spherical distance between A

and B, we first establish a point C at the same latitude as A and the same longitude as B. The coordinate of C is $(X1, Y2)$. O is the center of longitude for the circle of A. E is the center of longitude for the circle of B. We then have the points $OEBC$ coplanar. And the extension lines of CB and OE intersect at F (F is not an extreme point). We make line BD vertically cross OC, in Figure 5.1.

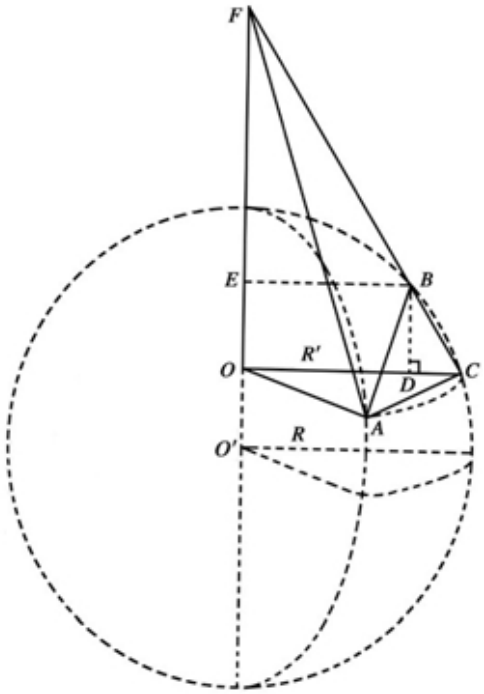


Figure 5.1: Calculation of inferior arc [5]

Since OF is vertical to the plane longitude circle of A and B, we have $OF \perp OC$,$OF \perp BE$,$OF \perp OA$. We will then get $\triangle OCF \sim \triangle BEF$, that is $\dfrac{BF}{CF} = \dfrac{BE}{OC}$

Since $\triangle FOA$,$\triangle FOC$ is Rt$\triangle$ it is easy to get:

$$AF^2 = OA^2 + OF^2 \tag{5.2}$$

$$CF^2 = OC^2 + OF^2 \tag{5.3}$$

As long as A and C are in the same longitude circle, $OA = OC$. That is $AF = CF$. In $\triangle FAC$ let $\angle ACF = \alpha$.

According to the laws of Cosine we can get:

$$AF^2 = AC^2 + CF^2 - 2 \times AC \times CF \times \cos\alpha \tag{5.4}$$

This can be simplified as:

$$\cos\alpha = AC/2CF \tag{5.5}$$

Thus, from:

$$AB^2 = AC^2 + CB^2 - 2 \times AC \times CF \times \cos\alpha \tag{5.6}$$

We can get:

$$AB^2 = AC^2 \times BE/OC + CB^2 \tag{5.7}$$

Considering the definition of longitude and latitude, we have:

$$OC = \cos x1, BE = \cos x2 \tag{5.8}$$

$$BC = 2 \times \sin((x1 - x2)/2) \tag{5.9}$$

$$AC = 2 \times \cos x1 \times \sin(\frac{y2 - y1}{2}) \tag{5.10}$$

From Equation 5.7 and 5.10, we got the following Equation 5.11:

$$AB^2 = 2[1 - \cos(x2 - x1) + \cos x1 \times \cos x2 - \cos x1 \times \cos x2 \times \cos(y2 - y1)] \tag{5.11}$$

And the Arc between A and B is:

$$2 \times \sin^{-1}(\frac{AB}{2}) \tag{5.12}$$

If the radius of the sphere is R, the Arc between A and B is:

$$2R \times \sin^{-1}(\frac{AB}{2}) \tag{5.13}$$

Therefore, when we consider the radius of Earth R, the calculation for the equation of distance of 2 points on earth is:

$$D = 2R\sin^{-1}(\frac{\sqrt{2[1-\cos(x2-x1)+\cos x1\cos x2-\cos x1\cos x2\cos(y2-y1)]}}{2}) \tag{5.14}$$

And the average radius of Earth $R = 6378000 \quad km$.

This algorithm has a moderate calculation speed; and, when used in a smaller distance ($< 100 \quad km$) area, it has good accuracy.

3. Ellipsoid distance algorithm [5]: Since the real Earth is not a strict sphere, a new algorithm needs to be deduced [5]. The strict geocentric coordinate is defined as follows: the coordinate origin is the ellipsoid center of the Earth (cancroids of the Earth). The Z-axis is, however, the short axis of the Earths ellipsoid, pointing to the Earth. The starting meridian of the Earth ellipsoid coincides with the Greenwich meridian plane. The coordinate origin point of X-axis points to the intersection in the starting meridian plane and the Earth ellipsoid equator. Y-axis and X, Z-axis, constitute the right coordinate (scale unit in meters (m)).

   Suppose the coordinates of a known points location are characterized by geodetic coordinates. This would mean that they contain the geodetic longitude L, the geodetic latitude B and the geodetic height H.

   Geodetic longitude L for any point K on the ground or in space is defined in the following. We will make the ellipsoid Earth meridian plane from point K. The angle between the made meridian plane and the starting meridian plane of the Earth is the geodetic longitude L for K. Geodetic latitude B is the angle between the normal KN and the equatorial plane. The height H is the distance from KN to the ellipsoid plane. This is shown in Figure 5.2.

   By the definition of trigonometric functions and the radius length formula of the prime vertical circle on the ellipsoid, we can then get (X, Y, Z) from (L, B, H):

$$if \begin{cases} X = (N+H)\cos B\cos L \\ Y = (N+H)\cos B\cos L \\ Z = [N(1-e^2)+H]\sin B \end{cases} \tag{5.15}$$

   N is the radius of the prime vertical circle: $N = \alpha\sqrt{1-e^2 sin^2 B}$. e is the first

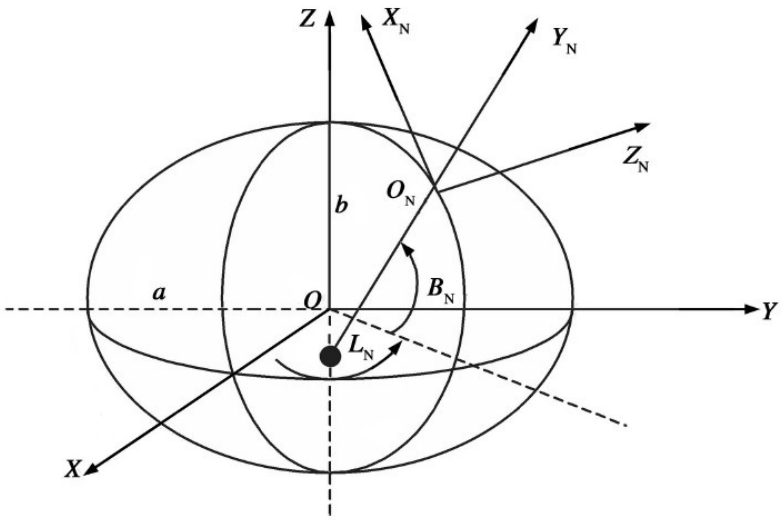


Figure 5.2: The Earth center rectangular coordinate system [5]

eccentricity of ellipsoid: $e = \dfrac{\sqrt{a^2 - b^2}}{a}$, and b is the long and short radius of the ellipsoid. And we can get $e = 0.0817$.

Suppose we get two points $A(L1, B1, H1)$ and $B(L2, B2, H2)$ and we obtain the transfer coordinate $A'(X1, Y1, Z1)$ and $B'(X2, Y2, Z2)$. Therefore, the relationship between two points in space is:

$$D = \sqrt{(X1 - X2)^2 + (Y1 - Y2)^2 + (Z1 - Z2)^2} \tag{5.16}$$

When adding the conversion coordinates to the formula, we can get the shortest linear distance D.

This algorithm has optimal accuracy. However, because we need many parameters when computing distance, the computing speed compared with the previous two others is longer and needs more computing resources. According to the description and characteristic of the three methods, we can obtain the following conclusions shown in Table 5.1.

Thus, in our situation, in a 100km range (assuming that the maximum operating speed of a vehicle in the city is 80km/h), we use the second method after balancing the performance and accuracy for our distance calculation in the system.

Algorithms / Compare	Rectangular Plane Coordinate	Spherical Distance Algorithm	Ellipsoid Distance Algorithm
Time Consuming	shortest	middle	longer
Calculation Complexity	easy	average	complex
Accuracy	low	good	high

Table 5.1: Comparisons of different algorithms in calculation distances

5.3 System Design

The system has two components: the data server and the client-side. We write a large dataset related in which city in the server side, which contains all the information of traffic sign on each road. The server side will then send the information of the signs according to the user vehicle position range. Data filtering is performed on the server side. According to the latency and high packet loss rate of Vehicle Ad-Hoc Networks (VANETs) , it is impossible and unnecessary to allow the car to request the data of the nearby sign frequently. Therefore, the car can request the data using the following strategy: we can let the server respond to the data from the road being traveled by the vehicle. When the driver changes the roads, the car needs to request again. In addition, if the car has a GPS navigation system, we can let the server respond to the signs on the entire navigation route.

The server-side has 3 steps for establishment. First, through information obtained by the GPS, we will get the signs in a grid range. That is, we will separate the maps in different ranges and select the signs in each range. Secondly, using road internet data we find signs on the road being driven on. Thirdly, the speed vector of vehicle and sign orientation vectors are used to define the orientation of signs. All the "face-signs" in the respective road will be provided in groups until the vehicle changes roads or requests data again. Here, we simply by outlining a concept and we will focus more on the client-side.

For the client-side, we have a buffer memory of data which contains only sign information. The position of vehicle is obtained through GPS and an anticipation value is necessary to decide whether the data in the buffer is used up. This is necessary to determine also when the driver changes to a new road and if the client-side needs to ask for new

data.

This concept combines the client-side with the server-side together. The speed of this method will give a brighter prospect than traditional methods, dealing with the image processing step on cars. As well, in the information era, we need to use the benefit of data storage to maximize efficiency.

5.4 Simulation on Client-side

We implemented the simulation on the client-side in a PC, using C++ with Opencv-library together.

First, we took a video on a street in Ottawa which contains the speed limit signs of 50Km/h. We found the longitude and latitude of the signs and the vehicle (using Google maps).

Then, when the vehicle continues to travel along the street, we calculate the distance between the vehicle and appearing signs. We used the formula in Section 5.2, method 2, every 30 ms (this time is used to avoid missing signs, for packet loss may occur) and another threshold necessary for giving warning message. When the distance between the vehicle and signs is smaller than the distance threshold, the warning message is then given to the driver. When the distance is smaller than 2 meters, the message is shine to the driver and then disappear after it has passed by. The data is then deleted from the buffer memory of the client-side. When the car is close to another sign, the same processing steps will happen again.

Client-side simulation:

Figure 5.3 and 5.4 show that when the car is close to the first sign. Figure 5.5 and Figure 5.6 show that when the car is close to the second sign.

Figure 5.3 and Figure 5.5 show the signs category and part of distance statistic data of the closing traffic signs.

Figure 5.4 and Figure 5.6 show the scenarios in the client-side, we can easily observe that the warning message and recognition signs are showed on client side in a safety distance range.

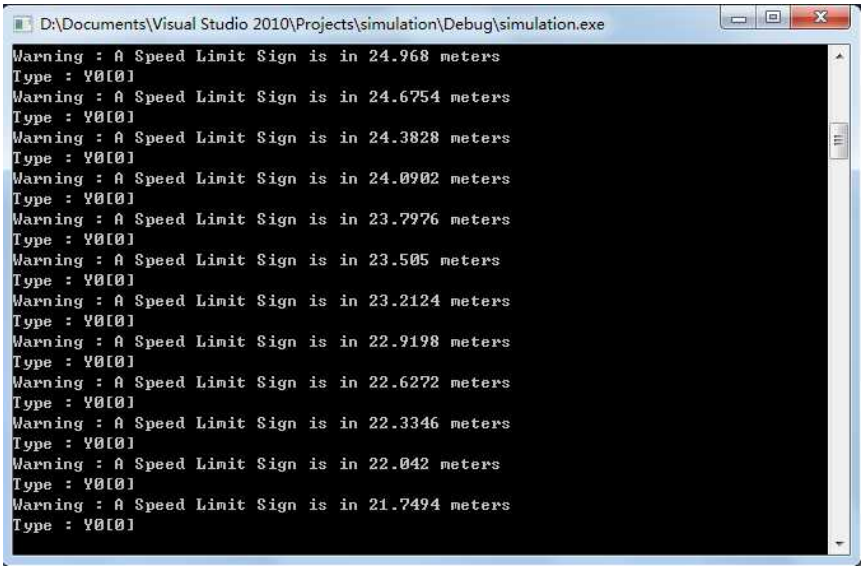


Figure 5.3: Data of first sign position near-by

Figure 5.4: Results of using new method

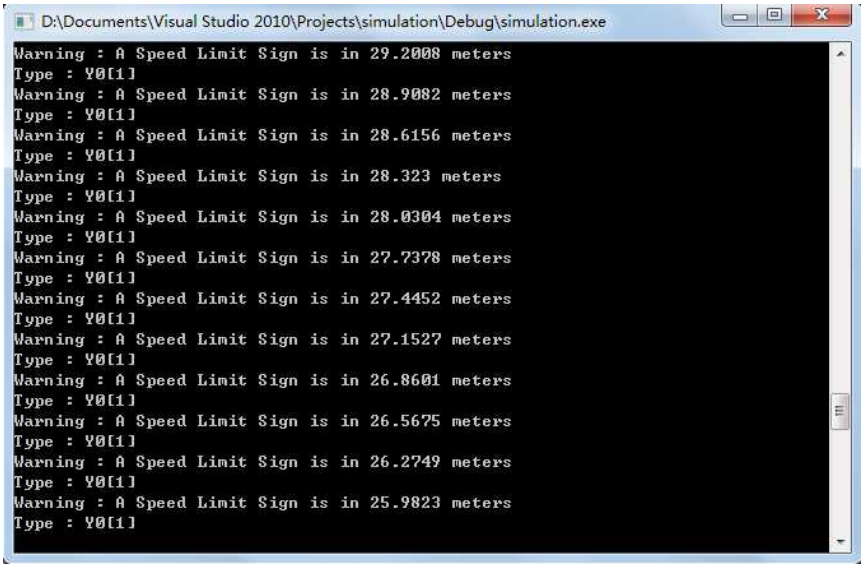


Figure 5.5: Data of second sign position near-by

Figure 5.6: Results of using new method

When compare with the system in Chapter 4 in solving the same problems, the time cost evaluation of these three methods: MSER based system, HOG based system and simulation without cameras is showed in Figure 5.7.

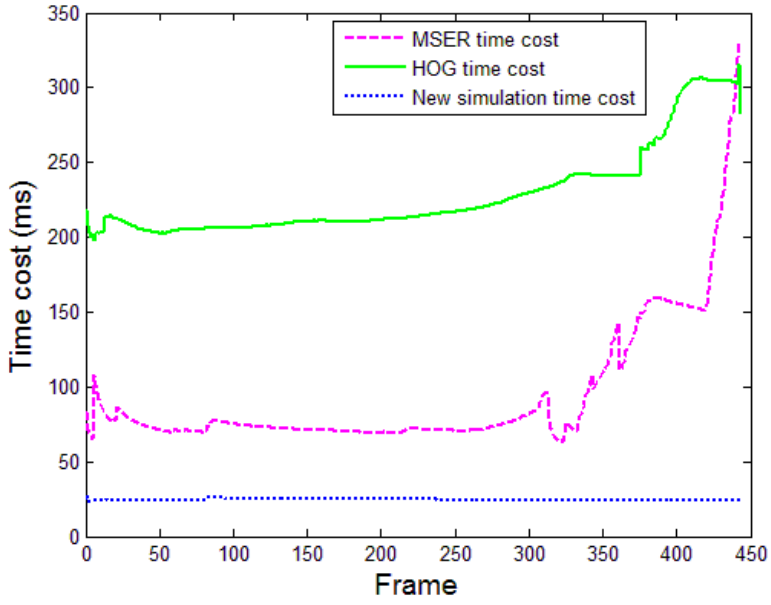


Figure 5.7: Total time cost of three used TSDR systems

Figure 5.7 shows the time consuming of TSRD systems used image processing methods and the methods without cameras based on same testing videos. Figure 5.8 shows the average time cost of different TSDR systems. The average time cost of HOG based system is 2.4 times than the time cost of MSER based system. Moreover, the time cost of MSER based system is 3.8 times longer than the time cost of the method without cameras.

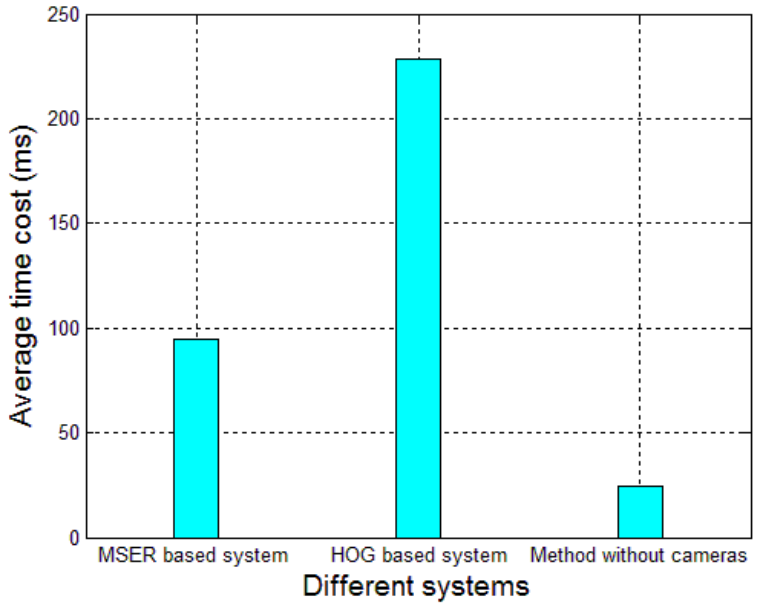


Figure 5.8: Comparison of average time cost of different TSDR systems

Chapter 6

Conclusion and Future Plans

6.1 Conclusion

In this thesis, we presented a new North American traffic sign dataset geared to work with the Canadian traffic system. Using our new traffic sign dataset, some traditional detection and recognition was introduced in the first few chapters based on both object characteristics and statistical methods. As for the statistical methods, we introduced a classical method called HOG and with it, we found a new set of HOG parameters for our new dataset. We used the foundation of traditional methods. We then obtained a new combination using color information extraction, traditional detection and recognition methods together. The purpose of this is to classify different colors of traffic signs; this is especially important for the differently-colored background speed limit signs. After this step, we proposed a new method to decrease the time cost of the traditional one, called MSER, this may decrease the utility of HOG, making the system less time consuming. After the image processing step was completed, we proposed a new concept that produced two parts of the new system: a data storage side and a client-side (we only focus on the client part of the simulation). Based on the navigational information, we were able to provide the information for traffic signs on the roads; and, we were able to reach our goal without using image processing.

The two criterions for evaluating different methods and whole system are:

1. Detection accuracy

2. Detection speed

The two factors are both important in the whole system because they guarantee the effectiveness of the system.

After doing research on all these aspects and testing these three systems, the results show that all of them can recognize traffic signs with a good accuracy rate. The MSER based system is faster than the one using only HOG and SVM. The method without camera is the faster one for solving the same problems encountered in the image processing stage.

6.2 Future Plans

Currently, although our dataset was updated daily, the number of images is still not enough. We need to find a way to increase the number of high-quality images and to focus on accommodating different weather conditions. This is because there are still various luminance conditions on the roads. As well, different lighting scenarios should be added for testing. A new feature (i.e. Haar features, LBP features, etc.) descriptor should be added to improve the time cost. However, the color information extraction, which we did, should not be ignored. The background color information is important, unless a new technical appears.

The method without camera which in Chapter 5 requires a big dataset, contains the coordinate information for sign position on the server side. For the image processing method and for other new methods, we need to make a feedback system. This needs to be done in case the signal of the position cannot be found. Additionally, if there is missing data, the image processing should be used to guarantee the stability of the whole system. This part will present a big challenge in the future; but, the balancing of these two methods will make a better system overall.

References

[1] *RGB Color Space*. http://photo.hanyu.iciba.com/upload/encyclopedia_2/88/95/bk_889574aebacda6bfd3e534e2b49b8028_vheBLq.jpg.

[2] *HSV Color Space*. http://www.tsi.telecom-paristech.fr/pages/enseignement/ressources/beti/correl_couleur/images/couleur-HSV2-hexcone.jpg.

[3] George Otto. *Organizing Color*. http://viz.aset.psu.edu/gho/sem_notes/color_2d/html/primary_systems.html.

[4] Intel. *Color Models*. https://software.intel.com/sites/products/documentation/hpc/ipp/ippi/ippi_ch6/ch6_color_models.html.

[5] Yu Wei. The distance between two points under the consideration of the curvature of the earth. *Aero Weaponry*, (3):7–11, 2008.

[6] Hidehiko Akatsuka and Shinichiro Imai. Road signposts recognition system. *Training*, 2013:12–11.

[7] Lanlan Liu Shuangdong Zhu and Xiaofeng Lu. Colors-geometric model for traffic sign recognition. *Computationla Engineering in Systems Application IMACS Multi-conference*, 2:2028–2032, 2006.

[8] Nasser Kehtarnavaz, NC Griswold, and DS Kang. Stop-sign recognition based on color/shape processing. *Machine Vision and Applications*, 6(4):206–208, 1993.

[9] Marie De Saint Blancard. Road sign recognition: A study of vision-based decision making for road environment recognition. In *Vision-based vehicle guidance*, pages 162–172. Springer, 1992.

[10] S Estable, J Schick, F Stein, R Janssen, R Ott, W Ritter, and Y-J Zheng. A real-time traffic sign recognition system. In *Intelligent Vehicles' 94 Symposium, Proceedings of the*, pages 213–218. IEEE, 1994.

[11] Giulia Piccioli, Enrico De Micheli, and Marco Campani. A robust method for road sign detection and recognition. In *Computer VisionECCV'94*, pages 493–500. Springer, 1994.

[12] Guo-Qing Wei and Song De Ma. Implicit and explicit camera calibration: Theory and experiments. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(5):469–480, 1994.

[13] Qiu Maoling, Ma Songde, and Li Yi. Overview of camera calibration for computer vision. *Acta Automatica Sinica*, 26(1):43–55, 2000.

[14] National Laboratory of Pattern Recognition (China). *Camera Calibration*. Chinese Academy of Sciences.

[15] Roger Y Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *Robotics and Automation, IEEE Journal of*, 3(4):323–344, 1987.

[16] *Camera Calibration*. `http://baike.baidu.com/view/380491.htm?fr=aladdin`.

[17] Peter Eisert, Eckehard Steinbach, and Bernd Girod. Multi-hypothesis, volumetric reconstruction of 3-d objects from multiple calibrated camera views. In *Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on*, volume 6, pages 3509–3512. IEEE, 1999.

[18] Olivier D Faugeras, Q-T Luong, and Stephen J Maybank. Camera self-calibration: Theory and experiments. In *Computer VisionECCV'92*, pages 321–334. Springer, 1992.

[19] Marc Pollefeys, Reinhard Koch, and Luc Van Gool. Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters. *International Journal of Computer Vision*, 32(1):7–25, 1999.

[20] Peicheng Hu, Ning Li, and Liangliang Zhao. A novel approach for camera calibration based on vanishing points. In *The 13th National Academic Conference on image and graphics*, pages 1–4, 2006.

[21] Song De Ma. A self-calibration technique for active vision systems. *Robotics and Automation, IEEE Transactions on*, 12(1):114–120, 1996.

[22] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.

[23] Andrzej Ruta, Yongmin Li, and Xiaohui Liu. Real-time traffic sign recognition from video by class-specific discriminative features. *Pattern Recognition*, 43(1):416–430, 2010.

[24] Stefano Marsi, Gaetano Impoco, Anna Ukovich, Sergio Carrato, and Giovanni Ramponi. Video enhancement and dynamic range control of hdr sequences for automotive applications. *EURASIP Journal on Advances in Signal Processing*, 2007, 2007.

[25] Greg Ward. High dynamic range imaging. In *Color and Imaging Conference*, volume 2001, pages 9–16. Society for Imaging Science and Technology, 2001.

[26] Doctorat Paristech, Mme Bernadette Dorizzi, M. Christian, M. Patrick, Siarry Examinateur, M. Bogdan, Stanciulescu Examinateur, M. Fawzi, and Nashashibi Examinateur. Multiclass object recognition for driving assistance systems and video surveillance, .

[27] JB Arens, AR Saremi, and CJ Simmons. Color recognition of retroreflective traffic signs under various lighting conditions. *Public Roads*, 55(1), 1991.

[28] Lukas Sekanina and Jim Torresen. Detection of norwegian speed limit signs. In *ESM*, pages 337–340, 2002.

[29] Hsiu-Ming Yang, Chao-Lin Liu, Kun-Hao Liu, and Shang-Ming Huang. Traffic sign recognition in disturbing environments. In *Foundations of Intelligent Systems*, pages 252–261. Springer, 2003.

[30] Jim Torresen, Jorgen W Bakke, and Lukas Sekanina. Efficient recognition of speed limit signs. In *Intelligent Transportation Systems*, pages 652–656, 2004.

[31] Arturo De La Escalera, Luis E Moreno, Miguel Angel Salichs, and José María Armingol. Road traffic sign detection and classification. *Industrial Electronics, IEEE Transactions on*, 44(6):848–859, 1997.

[32] Shuang dong Zhu, Zhany Yi, and Lu Xiao-feng. Detection for triangle traffic sign based on neural network. In *Vehicular Electronics and Safety, 2005. IEEE International Conference on*, pages 25–28, Oct 2005. doi: 10.1109/ICVES.2005.1563608.

[33] Jun Miura, Tsuyoshi Kanda, and Yoshiaki Shirai. An active vision system for real-time traffic sign recognition. In *Intelligent Transportation Systems, 2000. Proceedings. 2000 IEEE*, pages 52–57. IEEE, 2000.

[34] S Vitabile, A Gentile, and F Sorbello. A neural network based automatic road signs recognizer. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 3, pages 2315–2320. IEEE, 2002.

[35] Hirofumi Ohara, Ikuko Nishikawa, Shigeto Miki, and Noboru Yabuki. Detection and recognition of road signs using simple layered neural networks. In *Neural Information Processing, 2002. ICONIP'02. Proceedings of the 9th International Conference on*, volume 2, pages 626–630. IEEE, 2002.

[36] Zhu Shuangdong and Jiang Tian-tian. Intelligence approach of traffic sign recognition based on color standardization. In *Vehicular Electronics and Safety, 2005. IEEE International Conference on*, pages 296–300, Oct 2005. doi: 10.1109/ICVES.2005.1563660.

[37] Xiaohong W Gao, Lubov Podladchikova, Dmitry Shaposhnikov, Kunbin Hong, and Natalia Shevtsova. Recognition of traffic signs based on their colour and shape features extracted using human vision models. *Journal of Visual Communication and Image Representation*, 17(4):675–685, 2006.

[38] Luis David Lopez and Olac Fuentes. Color-based road sign detection and tracking. In *Image Analysis and Recognition*, pages 1138–1147. Springer, 2007.

[39] Andrzej Ruta, Fatih Porikli, Shintaro Watanabe, and Yongmin Li. In-vehicle camera traffic sign detection and recognition. *Machine Vision and Applications*, 22(2):359–375, 2011.

[40] King Hann Lim, Kah Phooi Seng, and Li Minn Ang. Intra color-shape classification for traffic sign recognition. In *Computer Symposium (ICS), 2010 International*, pages 642–647. IEEE, 2010.

[41] Claus Bahlmann, Ying Zhu, Visvanathan Ramesh, Martin Pellkofer, and Thorsten Koehler. A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 255–260. IEEE, 2005.

[42] Hilario Gómez-Moreno, Saturnino Maldonado-Bascón, Pedro Gil-Jiménez, and Sergio Lafuente-Arroyo. Goal evaluation of segmentation algorithms for traffic sign recognition. *Intelligent Transportation Systems, IEEE Transactions on*, 11(4):917–930, 2010.

[43] Thanh Bui-Minh, Ovidiu Ghita, Paul F Whelan, Trang Hoang, and Vinh Quang Truong. Two algorithms for detection of mutually occluding traffic signs. In *Control, Automation and Information Sciences (ICCAIS), 2012 International Conference on*, pages 120–125. IEEE, 2012.

[44] Saturnino Maldonado Bascón, Javier Acevedo Rodríguez, Sergio Lafuente Arroyo, A Fernndez Caballero, and Francisco López-Ferreras. An optimization on pictogram identification for the road-sign recognition task using svms. *Computer Vision and Image Understanding*, 114(3):373–383, 2010.

[45] Hasan Fleyeh. Shadow and highlight invariant colour segmentation algorithm for traffic signs. In *Cybernetics and Intelligent Systems, 2006 IEEE Conference on*, pages 1–7. IEEE, 2006.

[46] Wen-Jia Kuo and Chien-Chung Lin. Two-stage road sign detection and recognition. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 1427–1430. IEEE, 2007.

[47] Saturnino Maldonado-Bascon, Sergio Lafuente-Arroyo, Pedro Gil-Jimenez, Hilario Gomez-Moreno, and Francisco López-Ferreras. Road-sign detection and recognition based on support vector machines. *Intelligent Transportation Systems, IEEE Transactions on*, 8(2):264–278, 2007.

[48] Zhiyong Huang and guangmin Sun. Traffic sign segment based on rgb vision model. *Intelligent Transportation Systems, IEEE Transactions on*, 21(10):147–152, 2004.

[49] Alvy Ray Smith. Color gamut transform pairs. In *ACM Siggraph Computer Graphics*, volume 12, pages 12–19. ACM, 1978.

[50] Hasan Fleyeh. Road and traffic sign color detection and segmentation-a fuzzy approach. *Red*, 250:207, 2005.

[51] Wen-Jia Kuo and Chien-Chung Lin. Two-stage road sign detection and recognition. In *Multimedia and Expo, 2007 IEEE International Conference on*, pages 1427–1430. IEEE, 2007.

[52] Arturo De La Escalera, Luis E Moreno, Miguel Angel Salichs, and José María Armingol. Road traffic sign detection and classification. *Industrial Electronics, IEEE Transactions on*, 44(6):848–859, 1997.

[53] Min Yao. *Digital image processing*. Mechanical Industry Press, 2006.

[54] CG Kiran, Lekhesh V Prabhu, VA Rahiman, K Rajeev, and A Sreekumar. Support vector machine learning based traffic sign detection and shape classification using distance to borders and distance from center features. In *TENCON 2008-2008 IEEE Region 10 Conference*, pages 1–6. IEEE, 2008.

[55] S Lafuente-Arroyo, P Garcıa-Dıaz, FJ Acevedo-Rodrıguez, P Gil-Jiménez, and S Maldonado-Bascón. Traffic sign classification invariant to rotations using support vector machines. *Proceedings of Advabced Concepts for Intelligent Vision Systems, Brussels, Belgium*, 2004.

[56] WG Shadeed, DI Abu-Al-Nadi, and MJ Mismar. Road traffic sign detection in color images. In *Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003 10th IEEE International Conference on*, volume 2, pages 890–893. IEEE, 2003.

[57] Woong-Jae Won, Minho Lee, and Joon-Woo Son. Implementation of road traffic signs detection based on saliency map model. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 542–547. IEEE, 2008.

[58] Jun Miura, Tsuyoshi Kanda, and Yoshiaki Shirai. An active vision system for real-time traffic sign recognition. In *Intelligent Transportation Systems, 2000. Proceedings. 2000 IEEE*, pages 52–57. IEEE, 2000.

[59] A. Mammeri, A. Boukerche, and M. Almulla. Design of traffic sign detection, recognition, and transmission systems for smart vehicles. *Wireless Communications, IEEE*, 20(6):36–43, December 2013. ISSN 1536-1284. doi: 10.1109/MWC.2013.6704472.

[60] Miguel Angel Garcia, Miguel Angel Sotelo, and Ernesto Martín Gorostiza. Traffic sign detection in static images using matlab. In *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA'03. IEEE Conference*, volume 2, pages 212–215. IEEE, 2003.

[61] S-H Hsu and C-L Huang. Road sign detection and recognition using matching pursuit method. *Image and Vision Computing*, 19(3):119–129, 2001.

[62] Michael Shneier. Road sign detection and recognition. In *Defense and Security Symposium*, pages 623016–623016. International Society for Optics and Photonics, 2006.

[63] Cosmin Grigorescu and Nicolai Petkov. Distance sets for shape filters and shape recognition. *Image Processing, IEEE Transactions on*, 12(10):1274–1286, 2003.

[64] Nick Barnes and Alex Zelinsky. Real-time radial symmetry for speed sign detection. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 566–571. IEEE, 2004.

[65] Gareth Loy and Nick Barnes. Fast shape-based road sign detection for a driver assistance system. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 70–75. IEEE, 2004.

[66] Gareth Loy and Nick Barnes. Fast shape-based road sign detection for a driver assistance system. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 70–75. IEEE, 2004.

[67] Chun-Ta Ho and Ling-Hwei Chen. A high-speed algorithm for elliptical object detection. *Image Processing, IEEE Transactions on*, 5(3):547–550, 1996.

[68] Aryuanto Soetedjo and Koichi Yamada. Fast and robust traffic sign detection. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 2, pages 1341–1346. IEEE, 2005.

[69] Lutz Priese, Jens Klieber, Raimund Lakmann, Volker Rehrmann, and Rainer Schian. New results on traffic sign recognition. In *Intelligent Vehicles' 94 Symposium, Proceedings of the*, pages 249–254. IEEE, 1994.

[70] Chiung-Yao Fang, Sei-Wang Chen, and Chiou-Shann Fuh. Road-sign detection and tracking. *Vehicular Technology, IEEE Transactions on*, 52(5):1329–1341, 2003.

[71] Fatin Zaklouta and Bogdan Stanciulescu. Real-time traffic-sign recognition using tree classifiers. *Intelligent Transportation Systems, IEEE Transactions on*, 13(4): 1507–1514, 2012.

[72] Uwe Franke, Frank Lindner, Frank Paetzold, Dariu Gavrila, Steffen Görzig, and Christian Wöhler. Autonomous driving goes downtown. *IEEE Intelligent systems*, 13(6):40–48, 1998.

[73] Yuji Aoyagi and Toshiyuki Asakura. A study on traffic sign recognition in scene image using genetic algorithms and neural networks. In *Industrial Electronics, Control, and Instrumentation, 1996., Proceedings of the 1996 IEEE IECON 22nd International Conference on*, volume 3, pages 1838–1843. IEEE, 1996.

[74] Fabien Moutarde, Alexandre Bargeton, Anne Herbin, Lowik Chanussot, et al. Robust on-vehicle real-time visual detection of american and european speed limit signs, with a modular traffic signs recognition system. In *proc.*, 2007.

[75] Andrzej Ruta, Yongmin Li, Fatih Porikli, Shintaro Watanabe, Hiroshi Kage, and Kazuhiko Sumi. A new approach for in-vehicle camera traffic sign detection and recognition. In *MVA*, pages 509–513, 2009.

[76] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[77] Gary Overett, Lachlan Tychsen-Smith, Lars Petersson, Niklas Pettersson, and Lars Andersson. Creating robust high-throughput traffic sign detectors using centre-surround hog statistics. *Machine Vision and Applications*, 25(3):713–726, 2014. ISSN 0932-8092. doi: 10.1007/s00138-011-0393-1. URL `http://dx.doi.org/10.1007/s00138-011-0393-1`.

[78] Yuan Xie, Li-feng Liu, Cui-hua Li, and Yan-yun Qu. Unifying visual saliency with hog feature learning for traffic sign detection. In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 24–29. IEEE, 2009.

[79] Xu Qingsong, Su Juan, and Liu Tiantian. A detection and recognition method for prohibition traffic signs. In *Image Analysis and Signal Processing (IASP), 2010 International Conference on*, pages 583–586. IEEE, 2010.

[80] Jiang Gangyi, Zheng Yi, and Choi Tae Young. Morphological skeleton analysis of traffic signs on road. In *Systems, Man, and Cybernetics, 1996., IEEE International Conference on*, volume 1, pages 70–75. IEEE, 1996.

[81] Jesmin F Khan, Reza R Adhami, and Sharif MA Bhuiyan. Image segmentation based road sign detection. In *Southeastcon, 2009. SOUTHEASTCON'09. IEEE*, pages 24–29. IEEE, 2009.

[82] Pedro Gil Jiménez, Saturnino Maldonado Bascón, Hilario Gómez Moreno, Sergio Lafuente Arroyo, and Francisco López Ferreras. Traffic sign shape classification and localization based on the normalized fft of the signature of blobs and 2d homographies. *Signal Processing*, 88(12):2943–2955, 2008.

[83] Jun Miura, Tsuyoshi Kanda, and Yoshiaki Shirai. An active vision system for real-time traffic sign recognition. In *Intelligent Transportation Systems, 2000. Proceedings. 2000 IEEE*, pages 52–57. IEEE, 2000.

[84] Lars Petersson, Luke Fletcher, Nick Barnes, and Alexander Zelinsky. An interactive driver assistance system monitoring the scene in and out of the vehicle. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3475–3481. IEEE, 2004.

[85] Hasan Fleyeh and Mark Dougherty. Road and traffic sign detection and recognition. In *10th EWGT Meeting and 16th Mini-EURO Conference*, pages 644–653, 2005.

[86] Min Shi, Haifeng Wu, and Hasan Fleyeh. Support vector machines for traffic signs recognition. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 3820–3827. IEEE, 2008.

[87] S Lafuente-Arroyo, P Gil-Jimenez, R Maldonado-Bascon, F Lopez-Ferreras, and S Maldonado-Bascon. Traffic sign shape classification evaluation i: Svm using distance to borders. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 557–562. IEEE, 2005.

[88] Kun-ming WANG and Zhong-ren XU. Study on method of traffic signs recognition based on neural network and invariant moments [j]. *Application Research of Computers*, 3:092, 2004.

[89] Benjamin Hoferlin and Klaus Zimmermann. Towards reliable traffic sign recognition. In *Intelligent Vehicles Symposium, 2009 IEEE*, pages 324–329. IEEE, 2009.

[90] Alexandre Bargeton, Fabien Moutarde, Fawzi Nashashibi, and Benazouz Bradai. Improving pan-european speed-limit signs recognition with a new global number segmentation before digit recognition. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 349–354. IEEE, 2008.

[91] Hasan Fleyeh and Mark Dougherty. Traffic sign classification using invariant features and support vector machines. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 530–535. IEEE, 2008.

[92] Chia-Hsiung Chen, Marcus Chen, and Tianshi Gao. Detection and recognition of alert traffic signs. *Publication of Standford University*, 2008.

[93] Alexandre Bargeton, Fabien Moutarde, Fawzi Nashashibi, and Benazouz Bradai. Improving pan-european speed-limit signs recognition with a new global number segmentation before digit recognition. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 349–354. IEEE, 2008.

[94] Wei Liu, Yonghua Wu, Jin Lv, Huai Yuan, and Hong Zhao. Us speed limit sign detection and recognition from image sequences. In *Control Automation Robotics & Vision (ICARCV), 2012 12th International Conference on*, pages 1437–1442. IEEE, 2012.

[95] P Gil-Jimenez, S Lafuente-Arroyo, H Gomez-Moreno, F Lopez-Ferreras, and S Maldonado-Bascon. Traffic sign shape classification evaluation. part ii. fft applied to the signature of blobs. In *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*, pages 607–612. IEEE, 2005.

[96] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[97] Mirko Meuter, Anton Kummert, and S Muller-Schneiders. 3d traffic sign tracking using a particle filter. In *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*, pages 168–173. IEEE, 2008.

[98] S Lafuente-Arroyo, S Maldonado-Bascon, P Gil-Jimenez, J Acevedo-Rodriguez, and RJ Lopez-Sastre. A tracking system for automated inventory of road signs. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 166–171. IEEE, 2007.

[99] Paul R Kalata. The tracking index: A generalized parameter for α-β and α-β-γ target trackers. *Aerospace and Electronic Systems, IEEE Transactions on*, (2):174–182, 1984.

[100] Jan Eichhorn and Olivier Chapelle. Object categorization with svm: kernels for local features. 2004.

[101] Caltech. *Caltech 101 dataset*. `http://www.vision.caltech.edu/Image_Datasets/Caltech101/`.

[102] San Diego University of California. *Yale Face Database*. `http://vision.ucsd.edu/content/yale-face-database`.

[103] Abdelhamid Mammeri, Azzedine Boukerche, Jingwen Feng, and Renfei Wang. North-american speed limit sign detection and recognition for smart cars. In *Local Computer*

Networks Workshops (LCN Workshops), 2013 IEEE 38th Conference on, pages 154–161. IEEE, 2013.

[104] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks*, 32:323–332, 2012.

[105] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10): 761–767, 2004.

[106]

[107]

[108]

[109]

[110]

[111] Doctorat Paristech, Mme Bernadette Dorizzi, M. Christian, M. Patrick, Siarry Examinateur, M. Bogdan, Stanciulescu Examinateur, M. Fawzi, and Nashashibi Examinateur. Multiclass object recognition for driving assistance systems and video surveillance, .

[112] T. Gritti, Caifeng Shan, V. Jeanne, and R. Braspenning. Local features based facial expression recognition with face registration errors. In *Automatic Face Gesture Recognition, 2008. FG '08. 8th IEEE International Conference on*, pages 1–8, Sept 2008. doi: 10.1109/AFGR.2008.4813379.

[113] Thorsten Joachims. *Support Vector Machine*. `http://svmlight.joachims.org/`.

[114] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, number 1288, 2013.

[115] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1453–1460. IEEE, 2011.

[116] Abdelhamid Mammeri, Azzedine Boukerche, Jingwen Feng, and Renfei Wang. Traffic sign detection, recognition and transmission system for smart vehicles. In *GlobalCom*, 2013.

[117] Fatin Zaklouta, Bogdan Stanciulescu, and Omar Hamdoun. Traffic sign classification using kd trees and random forests. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 2151–2155. IEEE, 2011.

[118] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10): 761–767, 2004.

[119] J. Greenhalgh and M. Mirmehdi. Real-time detection and recognition of road traffic signs. *Intelligent Transportation Systems, IEEE Transactions on*, 13(4):1498–1506, Dec 2012. ISSN 1524-9050. doi: 10.1109/TITS.2012.2208909.

[120] N Kehtarnavaz and A Ahmad. Traffic sign recognition in noisy outdoor scenes. In *Intelligent Vehicles' 95 Symposium., Proceedings of the*, pages 460–465. IEEE, 1995.