**A Driving Assistance System:**

# Real–time Speed Limit Sign Recognition System

by

**Wen Liu**

A dissertation submitted to the University of Dublin, Trinity College,

In partial fulfilment of the requirements for the degree of

Master of Science in Computer Science.

(Mobile and Ubiquitous Computing)

**Supervisor: Dr. Kenneth Dawson-Howe**

***University of Dublin, Trinity College***

***August 2011***

# Declaration

*I declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.*

Signed: ——————————————————

*WEN LIU*

Date: 26th August 2011

# Permission to lend and/or copy

*I agree that Trinity College Library may lend or copy this dissertation upon request.*

Signed: ——————————————

*WEN LIU*

Date: 26th August 2011

# Acknowledgements

I would like to thank my dissertation supervisor, Dr. Kenneth Dawson-Howe for all his advice, help, guidance and encouragement throughout the progress of my dissertation. Dr Dawson-Howe has always been helpful and enthusiastic and I could not have finished my study without his assistance.

I would like to thank UbiCom class for their comments, encouragements and advices. I thank the course director Dr. Stefan Weber for lending Android phone to me for video collecting.

On a more personal note, I would like to express my gratitude to my family, landlady Mrs. Marry Fenlon and our neighbour Mr. Val Roche for all their support during this year. Especially, I would like to thank my newly married wife, Rui Wu, for persisting with me and helping me to collect testing videos and pictures.

# Abstract

Real-time speed limit sign recognition is an important component of driver assistance and intelligent transport systems. It would remind driver and intelligent driving system to keep the current speed within the speed limit specified by speed limit signs.

In this thesis, a robust approach for achieving real-time automatic detection and recognition of speed limit sign is presented. The system analyses images collected via a digital camera on the dashboard and will notify driver when a speed limit sign is detected. The system consists of three major steps, which are colour segmentation, sign detection and sign recognition. In the first two stages, colour segmentation and linear Support Vector Machine (SVM) are utilized to detect and extract possible speed limit sign candidates from scenes based on their colour and shape features. A series of rules is then applied to eliminate those candidates which do not have same visual features as a standard speed limit sign (i.e. number of black blobs on the ellipse, whether the ellipse contains a “Km/h” segment. etc.). In sign recognition stage, a new scan-line based digit character recognition algorithm is proposed and used to recognise the reading of speed limit sign. Experimental results in different weather conditions and image qualities, including sunny, cloudy, rainy weather, fine, poor and damaged image quality demonstrate that most speed limit signs can be detected and recognised by the system correctly with a very high accuracy which is 92.20% and a fast average processing speed which is 0.077 sec. per video frame (size 640 X 480) on a laptop computer.

# Table of Contents

# List of Figures

# 1 Introduction

## 1.1 Motivation

Traffic sign, especially speed limit sign recognition is an important task for a driver support system. Being able to automatically detect and recognise speed limit signs would be a very important component of intelligent vehicle driving system. The speed limit signs are used to guide the drivers to drive under the maximum speed. This will efficaciously reduce traffic accidents caused by over-speed. However, a driver may not notice a particular speed limit sign due to tiredness, distraction or lack of concentration. In this case, an automatic speed limit sign recognition system may be helpful to make drivers aware of speed limit information they have missed.

Some of ongoing commercial researches are concentrated on developing an automatic speed limit sign recognition system to increase the driving safety. The on-board car camera system developed by VDO (Siemens VDO)[1] gives us a good image of on-board speed limit sign recognition system (see Figure 1-1). The system uses a single video camera to scan the road for speed limit signs constantly. When the camera identifies a speed limit posted on a sign, the current driving speed is checked against that of the sign. The driver is warned if they are travelling above the speed limit detected. Furthermore, as an additional safety precaution, VDO intend to enable the system to project the information onto a heads-up display on the lower-half of the vehicle's windshield, and also integrate with adaptive cruise control technologies into the system, which would automatically slow down the vehicle if the driver is speeding.

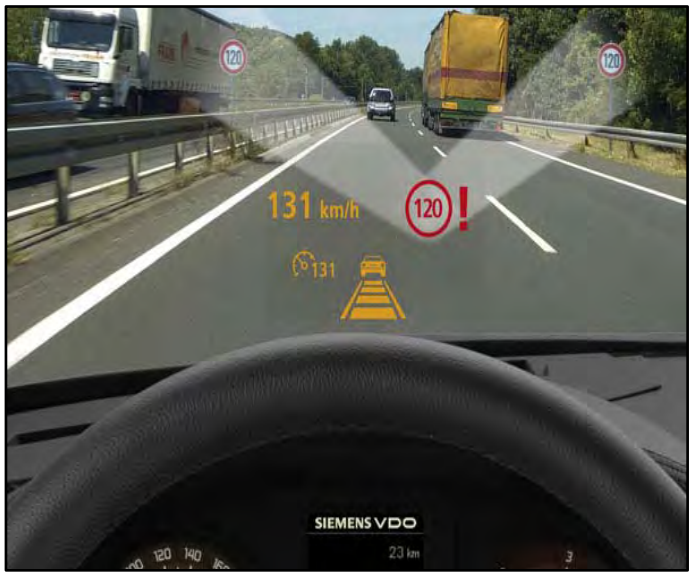



*Figure 1-1 Siemens VDO scans the road for speed limit signs and adjusts the speed accordingly.*

[1] VDO Global Website URL: www.siemensvdo.ro/ (Last visit: 06/08/11)

However, according to the latest researches in this field [3, 4, 6, 7], there is still room for improvement. The field is eager for new ideas and approaches to increase recognition accuracy, system reliability, portability and processing speed. Hence, developing a speed limit sign recognition system which would improve some of the features above is becoming the research goal of this project.

## 1.2 Research goal

The research goal of this dissertation is to design and develop a real-time automatic speed limit sign detection and recognition system, which will competently perform sign recognition task with high processing speed and accuracy. Exploring new ideas, approaches and / or algorithms and using them to enhance the system performance is another goal of the research.

Furthermore, in order to test the feasibility of running such a recognition system on hand-held mobile devices (i.e. Android cell phones), the system should also be able to demonstrate the same level of performance with limited hardware resources and lower image quality.

## 1.3 Dissertation structure

The second chapter – “State of the art”, highlights the developed technologies and concepts behind the speed limit sign recognition system. It provides the underpinnings of the various work and research that has been done in the field. Most of them are recently published on high standard academic papers and /or widely used in existing recognition systems. These technologies have been organized under three different sections which represent three major components of a typical speed limit sign recognition system respectively to conduct a better comparison. A conclusion which summarizes the study, and identifies the potential problem of existing technologies and possible solutions is drawn at the end of the chapter.

In the “System design” chapter, the general structure of the system, and how its components are linked together is described. The comparisons between alternative techniques and algorithms can also be found in the chapter. “Implementation and Results” chapter covers the process of building the product speed limit sign recognition system, the implementation of algorithms and the products of each development stages. Various experiments on the product system using video/image in different qualities/conditions, their results and evaluations of the research are described in “Evaluation and Discussion” chapter. The final chapter of this dissertation concludes the finding and product of the research. Relevant future research and works are also stated in this chapter. The references are listed in the bibliography and an “Appendices” which contains supplementary materials of the research can be found at the back of the document.

# 2 State of the Art

## 2.1 Introduction

The various works and researches that have been done on road sign recognition system in the last decade and some of them were aimed to build systems which achieve real-time processing [4, 8-14]. Most of these real-time systems consisted of a camera planted on the windscreen, monitoring outside environment, and a computer (usually a laptop), which reads and processes video frames from the camera (see Figure 2-1).

In this chapter, an overview of existing systems, techniques, and regulations which are used to implement the speed limit sign recognition will be presented. These speed limit sign recognition systems, in most cases, consist of two main stages: (1) sign detection and (2) sign numeral digits recognition. Furthermore, the sign detection stage is usually divided into two sub-stages, which are segmentation based on colour information, and speed limit sign detection according to the shape of region.

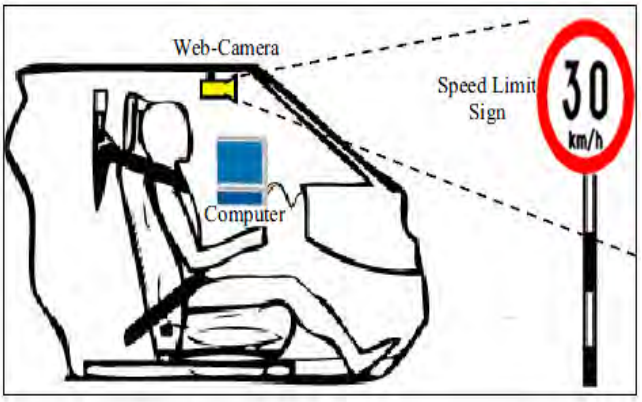



*Figure 2-1 A typical real-time speed limit sign recognition system.*

Firstly, in order to smooth the sign detection, colour segmentation stage extracts the candidate regions, which are the blobs containing most similar colour of an outside boundary of speed limit sign. Those regions are in white colour and distinguish from uninterested back ground pixels, which are in black colour. Ideally, the amount of pixels in candidate regions will be much less than in the original image. Then, these candidate regions are classified into circular pattern and non-circular pattern in the sharp detection stage – only the circular regions can be delivered to the final stage. The circle detection process is also considered as a critical determinant of both processing speed and recognition accuracy of the system [10, 15]. In the last recognition stage, the contents within the circular region will be tested. Contents in each region will be segmented and scaled into small blocks (e.g. 20*20 pixel [4] or same size as the templates if using template matching algorithms [12]). Each of these blocks is a blob in

non-white colour and it is scanned to recognise whether it is a digit character. After all blobs are recognised, and if the result shows into there are numerical characters, the result will be treated as the reading on the speed limit sign in current video frame and will be used to cue the driver via visual display (i.e. LED screen beside the dashboard) and/or via acoustic signals.

The following sections in this chapter describe the most popular techniques and algorithms used in above stages respectively.

## 2.2 Colour segmentation

Circle detection is usually considered as a calculating expensive process, most existing systems applied colour based segmentation to the image / frame before circular sign detection stage. The colour segmentation stage thresholds the image with a specified colour space and/or mechanism. The result image will be regions in white pixels which indicating the objects have similar colour with the outside boundary of speed limit sign. A variety of colour spaces have been used to do threshold process.

### 2.2.1 RGB / Greyscale

RGB is the most straight forward colour space for image processing. It consists of three additive primary colour channels: red, green and blue. However it is seldom used in direct colour segmentation part in this area [5].

In[16] and [8], the system sets up an image filter with RGB colour space to extract three colours: red, white and black from the given image. Their colour information will be stored respectively. The idea of this algorithm is to test if a candidate region is a speed sign by checking if there is a white region in a red region and if there is a black region in that white region.

In [6], the images are converted from standard RGB image into greyscale image directly. The reason for this is they assumed that the white area in the speed limit signs are unique and often appear brighter than other regions[6]. They threshold the greyscale image to generate the binary image and then find out most likely white region as the candidate for further processes. Paper [4] also uses greyscale image at the start of system to get the edge image for candidate sign region segmentation.

A significant drawback of RGB colour space is high sensitivity to the light condition [3, 5, 7, 17]. The segmentation result could vary in different light conditions due to weather and / or shadows. An appropriate fixed threshold value is also impossible to specify with this colour space.

### 2.2.2 HSV / HSL / HSI colour space

One of the most common colour spaces in computer vision area is HSI, as well as in road sign detection and recognition field [17]. This colour space comprises three components: Hue, Saturation and Intensity.

In [3, 7, 9], the original image is firstly converted from RGB to HSI colour space and then use a fixed threshold value to extract the colour regions of interest. The threshold values are chosen by analyzing the hue and saturation information of the image, usually according to the histograms (see Figure 2-2).

The [17] proposed a non-linear transformation to obtain the region of interest (RoI). Two colour look up tables (LUTs), which contain hue and saturation information respectively are used to threshold the image.

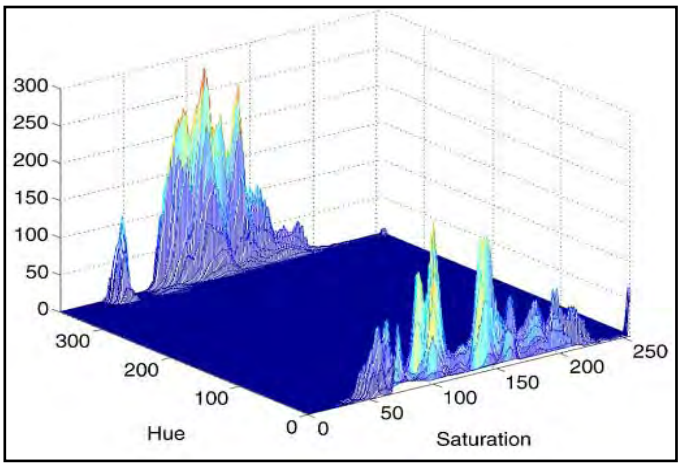



*Figure 2-2 Hue and Saturation histogram for red sign colour. [5]*

The HSI colour space has less variation to light condition than RGB. Hence the effect of light can be reduced to an acceptable level, the fixed threshold value can be applied for segmentation. A remarkable fact is that there is no uniform transformation formula for RGB – HSI conversion. All these papers are using different formula to convert the image to HSI colour space.

### 2.2.3 YCrCb / YUV colour space

The YCrCb colour space is composed of luminance, chrominance of red channel and chrominance of blue channel. The advantage of this colour model is that it separate luminance from chrominance information [2]. Therefore, the YCrCb can also be constant in different light and weather conditions.

In [12], a filter is set up to extract similar colour regions to red circle of the speed limit sign with a fixed threshold value. The filter is trained with sample pictures of signs in real-world. Paper [2, 14] converts image into YCrCb and builds space distribution colour models (see Figure 2-3) to identify optimized threshold values. Two optimized threshold values are used in these papers to provide robust isolation of red objects.

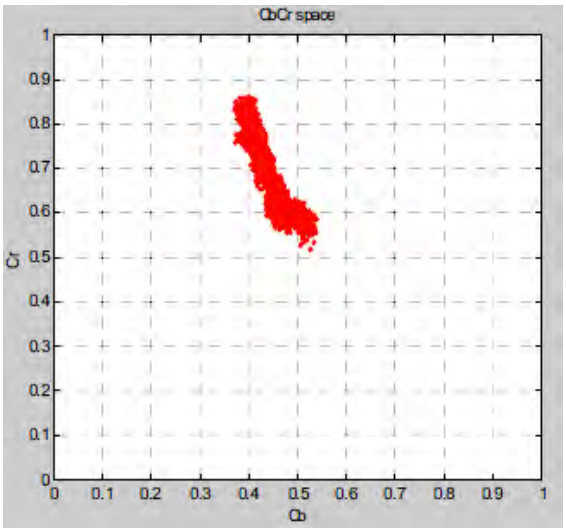



*Figure 2-3 Space distribution of red colour in YCbCr.[2]*

The YCrCb is widely used in speed sign recognition, not only because it resists variations of light condition but also the conversion formula is straightforward and the red chrominance is an important feature.

### 2.2.4 Achromatic / Chromatic components

Hence the pattern of a legal speed limited sign is a white round panel within a larger red circle, paper [5, 13, 18] go one step further to segment white panel of the sign with image achromatic decomposition. They proposed to treat white panel and the red circle surrounding it as achromatic colour and chromatic colour respectively. After red colour segmentation was processed based on hue information, the equation proposed in[13] is used to determine an achromatic / chromatic colour:

$$f(R,G,B) = \frac{(|R-G|+|G-B|+|B-R|)}{3D} \quad (1)$$

the *D* is the degree of extraction of an achromatic colour. Value of *D* can vary in different conditions – however, 20 is the most frequently used value [5, 13, 17, 18]. A colour is considered as achromatic colour if *f(R,G,B)* <= 1 and as chromatic colour otherwise. According to this process, a larger chromatic (red) region with a smaller achromatic (white) region in it has a good probability to be a speed limit sign and will be passed to circle detection phase as a RoI. Similarly, in Paper [17], a variation of equation (1) is used as one of four energy functions of the system to generate the chromatic image.

The Achromatic / Chromatic colour segmentation is an efficient way to eliminate amount of region passed to circle detection stage and increase the accuracy of detection performance.

## 2.3 Circular sign detection

The circle detection is applied to RoIs generated in previous colour segmentation phase to test whether a RoI is a circle. If it is, we can be positively sure that the RoI is a pattern of traffic sign. This process eliminates those RoIs which have similar size and colour feature with a speed limit sign to smooth the detection. A typical example of eliminated RoI could be a region of "YIELD" sign (see Figure 2-4) in the binarized image, which passed the colour segmentation due to high similarity in colours.

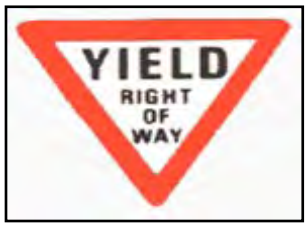



*Figure 2-4 A YIELD sign with high similarity in colours with a speed limit sign.*

### 2.3.1 RANSAC for circle

The Random Sample Consensus (RANSAC) algorithm [19] is proposed by Fischler et al. in 1981. It is one of the most widely used techniques for model fitting in industry area[20]. Basically, the algorithm takes a dataset (i.e. the contour), randomly select a sub-set of sample features (i.e. points) and try to fit a geometric shape model (i.e. a circle). Then, compare the model with the whole feature set. The RANSAC process terminated if sufficient number of features from the dataset satisfy the model with a tolerance. In[14], the following steps are used to carry out the circle detection[20]:

*Step1: Select a set of p=3 contour points randomly (three points are required to determine a circle).*

*Step2: Construct a circle through these points.*

*Step3: Count the number of points which lie within an error tolerance of* $\epsilon_{max}$ *to the circle (called inliers)*

*Step4: If the number of inliers is greater than some threshold* $n_{min}$ *, do least squares circle fitting for all inliers, else repeat the above process, i.e. start again at step1, until a maximum number* $m_{max}$ *of trials is reached.*

As the algorithm is quite straightforward and there are only a few regions passed colour segmentation stage, the processing speed is fast enough for real-time detection. Paper[7] also claimed that the system has been successfully tested under various light and weather conditions with good performance. Furthermore, the RANSAC algorithm is also able to detect the circle object even with incomplete contour. This is an extremely important feature for speed sign detection not only because the various light conditions, but also some uninterested object may block part of contour of speed limit sign. These are very common scenes in the urban environment where most of speed signs exist.

### 2.3.2 Gradient / edge based circle detection

The gradient /edge based algorithm is a traditional circle detection technique. Especially, in speed limit sign detection field, this algorithm has an obvious advantage – because of the sign consists of two parts in distinct colours (red and white). This makes the edge between these two parts even easier to detect. In general, the algorithm calculates edge image of the regions first, and then, the gradient of each edge pixel is calculated. The core of this algorithm is to test if there is a centre point within the region which pointed by all edge gradient directions and has a constant distance to all edge pixels[10].

In [12, 17], the authors applied original gradient / edge based circle detection algorithm to their systems. Paper [12] set a search area around each candidate region. For every search area, they applied Canny edge detector to it to extract the edge image. They proposed that if an edge is a part (i.e. an arc) of a circle, the centre of the circle should exist on the line which passes the edge and has the same direction as the gradient of the edge. They calculate such a line for each edge in a region and vote the line in the search area. The region is considered containing a circle if there is a prominent peak exceeding some threshold in the area. A distance transform is applied to the edges rather than consider their gradient direction directly in paper [17]. They use Hausdorff distance[21], which indicates how two shapes differ to test the edge image of region. There is a novel edge-voting principle used to detect a circle in [3]. In this algorithm, a circle is divided into four phases (see Figure 2-5). They use two 3*3 Sobel operators (horizontal and vertical directions) to extract edges of images firstly, and then for each point on the edge, they assign a phase value to it according to gradients of that edge point, using the equation (2).

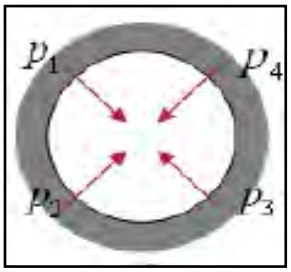



*Figure 2-5 Four phases of a circle [3].*

$$P_m = \begin{cases} 1, & if\ G_x > 0\ and\ G_y > 0 \\ 2, & if\ G_x < 0\ and\ G_y > 0 \\ 3, & if\ G_x < 0\ and\ G_y < 0 \\ 4, & if\ G_x > 0\ and\ G_y < 0 \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

It ends up with a phase matrix of the region, and an edge variation mask, which its size is the same with phase matrix and the values in this edge variation mask are $v \mid v \in \{1, 2, 3, 4, 0\}$. The pattern of this mask is similar to Figure 2-5 and it applied to the phase matrix to conduct the edge comparison matrix by multiplying itself with the phase matrix. Finally, values in edge comparison matrix are summed to compare with the threshold value $T_{mask}$. If sum is larger than, or equal to $T_{mask}$, it can then be confirmed that a circle is allocated in current region and no circle otherwise. After the circle is detected, authors also proposed a centre-voting principle to

allocate the centre of that circle [3], therefore, extract the circle. Hence the detection is based on gradient / edge of image, it has a good resistance to various light conditions and less sensitivity to noise.

### 2.3.3 Circular templates matching

In object detection and recognition area, template matching is always considered as a simple and straightforward algorithm. As well as in speed limit sign recognition, template matching has been used in many previous researches [2, 8, 16, 18, 22]. Basically, template matching is a process that compares an unknown sample with specified templates and calculates similarities between them to classify the unknown sample. In [18], a library of pictures of various speed limit signs are utilized as templates to eliminate the ambiguity caused by various template selection schemes and only one template is involved in the matching of each sample at a time. The similarity of current match is measured by the minimum distance between their tangent function. If the similarity exceeds the threshold value, this sample will be accepted as a matched speed limit sign sample. Otherwise, next temple in the library will be involved in the matching. The same algorithm is used in [16] and [8]. The authors set up a template library with various sizes of speed limit sign (i.e. in size 32*32, 38*38, 50*50 etc.). Systems reject samples which do not come out with an acceptable similarity.

Paper [2] proposed some improvement to the original template matching algorithm. They scale the sample to the size of template and apply matching between templates and sample to identify the sample with highest cross correlation. Correlation describes the convolution theorem and the attendant possibility of efficiently computing correlation in the frequency domain using the Fast Fourier Transform (FFT) [2]. For example, a sample *f(i,j)* is given and a template *g(i,j)* is involved in the matching. The system place the template at a location *(i,j)* on the sample image to detect its presence at that point by comparing gray scale values. Since it is rare that two gray scale values match exactly, they measure the dissimilarity of these two values. In order to obtain all locations and instances of the template applied to the sample, the template is shifted over the sample and measured at every point in the sample. Therefore, for a template *(m,n)*, use formula (3) to calculate the cross correlation:

$$\mathrm{N}(m,n) = \frac{\sum_{\mathrm{i}=0}^{m}\sum_{\mathrm{j}=0}^{n} g(i,j)\,t(i-m,j-n)}{\left(\sqrt{\sum_{i=0}^{m}\sum_{\mathrm{j}=0}^{n} g(i,j)^2} \times \sqrt{\sum_{i=0}^{m}\sum_{\mathrm{j}=0}^{n} t(i-m,j-n)^2}\right)} \qquad (3)$$

where *i* and *j* are coordinates of a point in sample image. After computations are done for every possible position over the entire sample image, the local maxima value can be determined, and compared to threshold *T*. The benefit of this improved algorithm is that the difference of luminance between template and sample does not affect the result. This algorithm is also used in

system proposed in [22]. However, the drawback of circle template matching is also obvious. It is very sensitive to the distortions, which are quite frequently occurring under different condition or using different camera to monitor scenes.

### 2.3.4 Circular Hough transformation and its variation

Circular Hough Transformation is a typical approach to detect a circle[23]. It transforms object from image space to accumulated Hough space to find the greatest local maxima. The Hough space is an accumulator which accumulates evidence of the likelihood of a circle being present using equation $(i-a)^2+(j-b)^2 = r^2$ where $r$ is a given radius of the circle, $(i,j)$ and $(a,b)$ are circle centre in image and Hough space respectively. However, the greatest issue for Hough transformation is the processing speed as it is always considered as an expensive algorithm. Although research for reducing the computational cost for this algorithm is still ongoing, many systems in speed limit sign recognition field use this mechanism to detect sign because its high accuracy and ability of partial circle object detection.

In some recent system such as in [6, 15], Hough Transformation is used associated with Adaboost algorithm [24] to detect speed limit signs. The Adaboost is a powerful machine learning algorithm. It has a cascaded structure which consists of several sub-processing stages, and a set of Haar-like features is given to every stage. If the sample blob fails to pass any sub-processing stage, it will be rejected as an uninterested region. The Adaboost algorithm is trained with a huge amount of positive and negative patterns captured in various environments and conditions to build up this cascade detector. A sample passed through the trained cascade is detected to be a candidate region of circular Hough transformation process. The amount of candidate regions is ideally small and hence the Hough transformation can be carried out in very little processing time. The performance of detection, due to these Adaboost – Hough system, is improved on both efficiency and accuracy aspects. Furthermore, the system is also proved to be insensitive to bad conditions [15]

A novel variation of circular Hough transformation is proposed in [9] called Fast Radial Symmetry Transform, and it is also used to detect speed limit signs in [4]. The proposed algorithm executes in order $kp$, where $k$ is the number of searched discrete radius and $p$ is the amount of pixels. As an opposite, original Hough transformation executes in order $kbp$ where $p$ votes on all circles over a discrete set of radius $k$ which could pass through the $p$, and $b$ comes from the discretization of bins on the gradient of circular tangents that could pass through this point. The variation eliminated $b$ by getting edge point gradient directly from the output of Sobel edge detector. Hence, the computation of radial symmetry is reduced and resulting circle map is also simplified by a dimension. It turns out that the system takes advantage of Hough Transformation with a much faster processing speed [4]. The algorithm is suitable for real-time

process that takes 13.2 ms to detect circles from a 240*320 image [9].

### 2.3.5 Linear Support Vector Machine

Support Vector Machine (SVM) was introduced by Vapink [25]. It is a set of related supervised learning mechanisms that used to recognise and analyse patterns. In road sign recognition field, Paper [5] introduced a novel shape detection algorithm based on linear SVM, and this system has also been proved to be an effective way of detecting road signs in [26].

In the system, blobs that passed through colour segmentation stage are classified by the linear SVM. The SVM takes DtBs (Distance to Boundary) as feature vectors for the inputs. These four DtB vectors are obtained from each sample blob. The DtBs are the distance from the external edge of the blob to its bounding box boundary [5]. To obtain a DtB, every blob is normalized to a size fixed bounding box first. A set of scan lines (20 lines in the system) search the outer edge of the blob starting at the boundary of the box, from left to right side. The line stops when it hit the edge and next line starts to scan in the same way. After all scan lines finish scanning, the DtB vector is formed based on lines first hit locations. The Figure 2-6 shows an example of DtBs of a sample blob where *D1, D2, D3* and *D4* are presenting top, right, bottom and left DtB respectively. Then, these four DtB vectors are fed SVMs to classify the shape of sample blob as a circle or not, by holding four favorable votes. The sample blob will be rejected as a “noisy shape” if the voting result is lower than a specified threshold value. At the end, paper [5, 26] claimed that the detection process using SVM is also invariantable to rotation, translation and scale, beside its high effiency and accuracy.

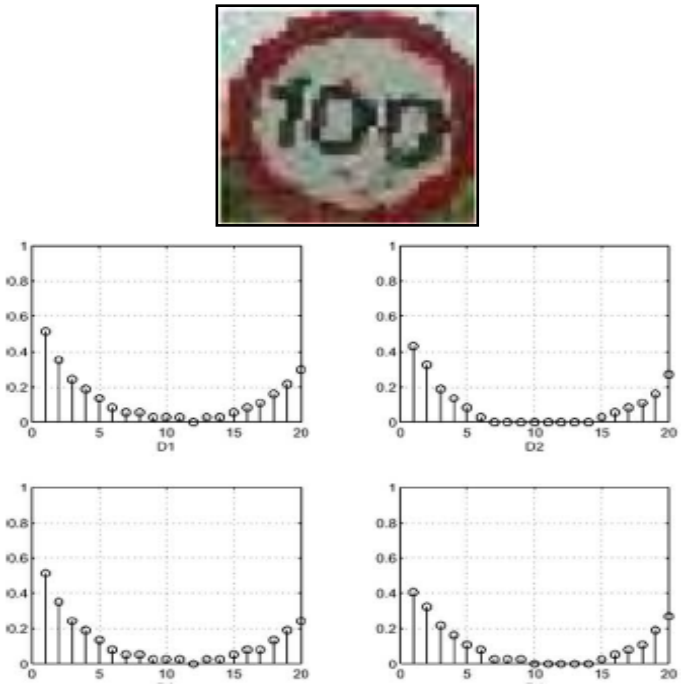



*Figure 2-6 Four DtB vectors of a blob that contains a speed limit sign (above) [5].*

## 2.4 Speed limit sign recognition

This step is known as the last stage in most speed limit sign recognition systems. The blobs of interests are passed to the recognition module where the Optical Character Recognition (OCR) process is to take place. In speed limit sign recognition field, there are only ten digit characters (0 ~ 10) needing to care about. The common approaches are using a trained Artificial Neural Networks to classify the sample blobs [2, 3, 14, 16] or through Fuzzy template & local feature matching[4, 15]. Moreover, scan-line based digit recognition is also widely used in document acquisition and automated meter-reading [1, 27, 28]. This technique can also be used to recognise digits on the speed limit sign. The result of this process is treated as the reading of the speed limit sign. System informs driver this result via visual and / or acoustic signals. Thus, a process cycle of speed limit sign recognition is finished.

### 2.4.1 Artificial Neural Networks

An Artificial Neural Networks (ANN) is a mathematical or computational model which consists of an interconnected group of artificial neurons. It processes data using a connectionist approach, akin to the vast network of neurons in the human brain. Nowadays, modern ANNs are non-linear statistical data modelling tools and widely used to detect patterns in data or model complex relationships between inputs and outputs.

The recognition stages in paper [2, 14, 16] are implemented by using a multilayer feed forward neural network. The basic feed forward network performs a non-linear transformation of input data in order to approximate the output data. This neural network contains three different layers: input layer, hidden layer and output layer [29]. Each layer consists of numbers of nodes. The input layer is passive which means it doesn’t modify the data and the other two layers are active. For example, every node in input layer is the colour value (usually ‘0’ for black and ‘1’ for white) of a single pixel in a binary image. Each value from the input layer is duplicated and sent to all of the nodes in hidden layer. This is called a fully interconnected structure. The hidden layer is usually about 10% the size of the input layer [29]. The values entering the hidden node are multiplied by weights, which a set of predefined value for hidden nodes. After all input values are multiplied by the weight, they will be added to produce a single sum. Before leaving the node, this number is passed through a nonlinear mathematical function called a sigmoid. This is an ‘S’ shaped curve that limits the node's output. That is, the input to the sigmoid is a value between -∞ and +∞, while its output can only be between 0 and 1. Each of these outputs from the hidden layer are duplicated and applied to the output layer. The active nodes in this layer will combine the entering values together to generate the outputs, which are usually

possibilities of output categories. The category with most likelihood is the final output of this feed forward ANN. Before using the ANNs to process data, it must be optimized through various training algorithms [29]. In paper [2], the ANN is trained using binary images of digits 0~9 with size 32 X 32 pixels, which means there are 1024 nodes in its input layer. The model contains 15 and 10 neurons in its hidden and output layer respectively. The 10 output neuron consists of 9 desired output neurons, which are representing 9 most common readings of speed limit sign (15, 30, 40, 50, 60, 70, 80, 90 and 110 km/h), and one undesired output neuron which representing non-digit input (see figure 2-7). After all output layer neurons get their result, the final output will either be a speed limit sign reading (for example 50 km/h) or non-digit reading (i.e. not a speed limit sign) depends on the most likelihood possibility output. Paper [14, 16] also use the same ANN model to recognise the readings of speed limit sign. In paper [14], the input layer of ANN contains 400 neurons (20 X 20 pixel samples). The size of the hidden layer was set by empirical experimentation, which shows that the 30 neurons in hidden layer giving the best performance. There are 12 neurons in the output layer corresponding to the following signs types: 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 km/h, national-speed limit (in UK/Poland) and false positive (i.e. not a sign). Each output value is the classification likelihood between 0 and 1 for the corresponding class (i.e. type) of sign and the largest value indicates the output of ANN.

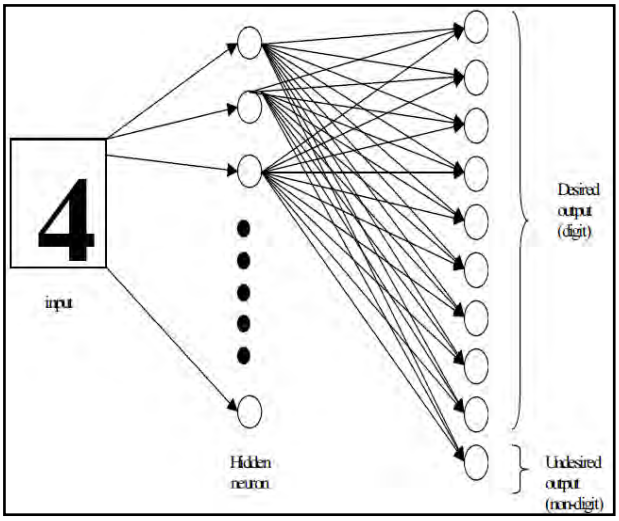



*Figure 2-7 A sample ANN structure for speed limit sign recognition [2].*

Researchers claim at the end of paper [2, 14, 16] that the system has achieved a very accurate performance with 90 to 92% recognition accuracy. The recognition speed of ANN models is also acceptable for real time processing. However, the cost of precise ANN recognition is large amount of training samples and building a model of complex multilayer structure.

### 2.4.2 Fuzzy template & local feature matching

Contrary to ANN approaches, template matching has no need of collecting samples and training the model. However, The traditional template matching algorithms are not suit for recognizing the readings of speed limit sign due to image blur, weather or illumination condition [4, 15]. To overcome that, Paper [4] proposed a algorithm which combines template matching and local feature matching together. Firstly, the system applies fuzzy template matching method to coarse

recognise the sign number character. Then, to avoid mismatching and achieve robust recognition performance, a fine recognition process based on local feature vector is applied. The approach of first part of the system is template matching using Normalized Cross Correlation as the matching criterion, which has been described in section "2.3.3 Circular template matching". After this, the features of the sample image will be extracted to form the local feature vectors.

The reason for applying the local feature vector is to identify the similar characters under blurs and different luminance. The features of character can be number / position of holes [15] and distribution of character pixel (different from background pixel) at the top / bottom of the image [4]. An example of distinguish character '4' and '6' on the speed limit sign is given in [4]. Because the numbers of hole for '4' and '6' are both one, the feature to identify the correct character is the difference between distributions of white pixel (background of the road sign) at the bottom. To form the local feature vector, the system search background pixel at the bottom of the image. The search of a column terminated with character pixel (non-background pixel) encountered. Then, system counts the number of background pixel passed and starts to search the column next. After all columns have been searched, counts of background pixel are normalized to size 20 according to the searching order (i.e. left to right) to consist local feature vector. (See figure 2-8.) The local feature vector of fuzzy template can be obtained in the same way. Then, the Euclidean Distance of local feature vectors between character image and fuzzy template can be calculated by using the following equation (4):

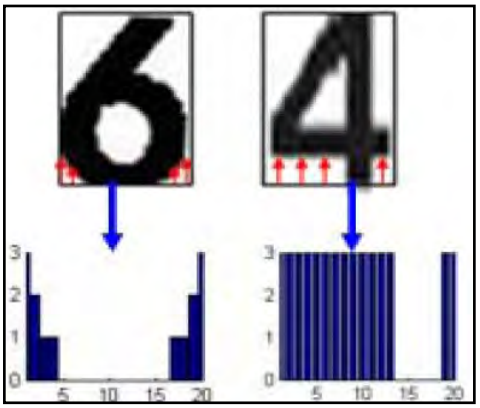



*Figure 2-8 Two local feature vectors for digit "6" (left) and "4" (right). The blue stripes represent the number of background pixel in each column.[4]*

$$\text{Distance}(i) = \sqrt{\sum_{j=1}^{20}(z_j - z_j^i)^2} \ , (i = 4, 6) \qquad (4)$$

Where $Z^i = \{z_1^i, \ldots, z_{20}^i\}$ is the feature vector of the fuzzy template *i.* and $Z = \{z_1, \ldots, z_{20}\}$ is the local vector of the character to be recognised. Hence, the reading of the speed limit sign is finally recognised as the best matching result which has the smaller Euclidean distance. Similarly, the system in paper [15] uses this algorithm in recognition stage with some other local features (i.e. position of top / bottom row of character pixel, number of left / right column across of character pixel).

This algorithm retains the simplicity and flexibility of the template matching. It enhances the reliability and improves performance of digit recognition.

### 2.4.3 Scan-line based digit recognition

Scan-line based OCR technique is widely used in automated meter-reading (i.e. automated record readings from ammeter dial plate [27, 28]) and document acquisition because its simplicity and fast recognition speed. In fact, this algorithm is suit for various purposes – including speed limit sign recognition [1, 28]. The idea of this algorithm is using multiple vertical and / or horizontal scan-lines to go through the input image and mark those non-background pixels encountered. These pixels and relevant patterns are critical features which identify the out digit character by matching with pre-defined digit character (0 ~ 9) feature matrixes. The digit character with the highest similarity with critical features is the output of system. If the similarity is below the threshold value (i.e. 95%), the input is treated as a non-digit character.

Paper [1] gives a detailed example of how this efficient algorithm works. Firstly, the system detects the left (*X min*), right (*X max*), top (*Y max*) and bottom (*Y min*) boundary of the input character image. Then, a vertical scan line is drawn through the middle point of image width (i.e. through point which X = *X min + [X max – X min]/2*). All non-background pixels the scan line encountered are marked as 1, 2,….*n,* from top to bottom of the character (see figure 2-9). For example, there are 6 critical points for images contain digit '2' and '8'.

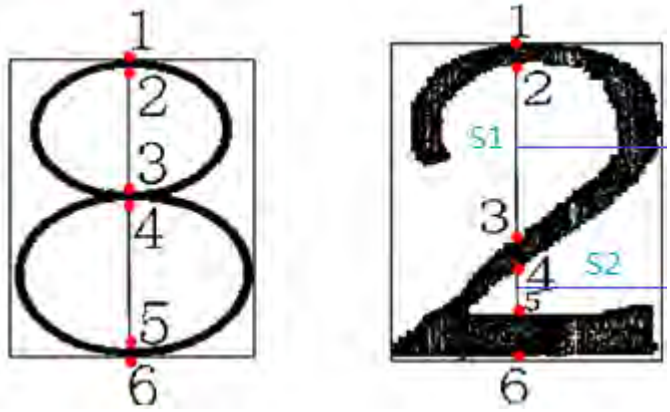



*Figure 2-9 The distribution of critical points the vertical scan-line encountered, and two horizontal scan-lines for digit '8' and '2'.[1]*

After marking critical points, the algorithm to determine the character is as follows:

1): if there are only two critical points (minimum of possible number of critical points), the pattern in the input image is digit character **'1'** (i.e. the feature matrix for digit '1' is "one critical point").

2): if the number of critical points is more than six (i.e. seven critical points), the pattern is not a digit character (i.e. return "None").

3): otherwise, if the number of critical points is more than three (i.e. four critical points), for all even critical point, except the last one (i.e. critical points No.6 for '8'), do inner edge tracing. Mark *Feature1* = "true" if first tracing success and *Feature2* = "true" if second tracing success. Otherwise, mark "false". (i.e. for pattern '8', the first tracing starts at critical point No.2 and the second starts at No.4. Both *Feature1* and *Feature2* are marked as 'true'. Similarly, for pattern '2', both are marked as 'false').

4): otherwise, if the number of critical points is six, *Feature1* = "true" and *Feature2* = "true", the pattern is digit character **'8'.**

5): otherwise, if the number of critical points is six, *Feature1* = "true" and *Feature2* = "false", the pattern is digit character **'9'.**

6): otherwise, if the number of critical points is six, *Feature1* = "false" and *Feature2* = "true", the pattern is digit character **'6'**.

7): otherwise, if the number of critical points is four, critical point No.4 is not *Y min* and *Feature1* = "true", the pattern is digit character **'4'**.

8): otherwise, if the number of critical points is four, critical point No.4 is *Y min* and *Feature1* = "true", the pattern is digit character **'0'**.

9): otherwise, if the number of critical points is four, *Feature1* = "false" and *Feature2* = "false", the pattern is digit character **'7'**.

10): otherwise, if the number of critical points is six, *Feature1* = "false" and *Feature2* = "false", do two horizontal scan-lines (towards right) through the middle point of critical point No. 2 & No.3 and through the middle point of No.4 & No.5 respectively (See figure2-9 above). Mark *Feature 3* = "true" if S1 encountered any non-background pixel and *Feature 4* = "true" if S2 encountered non-background pixel. Otherwise, mark "false".

11): if the number of critical points is six, *Feature1* = "false", *Feature2* = "false", *Feature 3* = "false" and *Feature 4* = "true", the pattern is digit character **'5'**

12): otherwise, if the number of critical points is six, *Feature1* = "false", *Feature2* = "false", *Feature 3* = "true" and *Feature 4* = "true", the pattern is digit character **'3'**

13): otherwise, if the number of critical points is six, *Feature1* = "false", *Feature2* = "false", *Feature 3* = "true" and *Feature 4* = "false", the pattern is digit character **'2'**

14): otherwise, the pattern in the input image is a non-digit character, returns "None".

Through these 14 conditions above, the pattern of the input image can be identified as either a single digit character or non-digit. Since the reading of a speed limit sign could be vary in number of digit (i.e. 5 km/h – one digit, 10 km/h – two digits and 115 km/h – three digits). However, if the number of digit is 3, the algorithm is applied at most two times to generate the final reading as the first digit is always '1'.

Some obvious features of this algorithm include fast process speed, stronger resistance to image blurs and variation of font. The concept of this algorithm is straightforward and easy to implement. The similar technique is used in systems proposed by paper [27, 28]. The difference is that in paper [28], a image thinning and burrs deleting process are applied to the input image to be recognised before drawing scan-lines in order to reduce the effect of noise. It is not really necessary because the width all three scan-lines is only 1 pixel.

## 2.5 Summary of literature review

In this State of the Art chapter, all needing algorithms and techniques for building a real time speed limit recognition system has been studied and identified. Features, advantages and drawbacks of these techniques are also identified. They are formed into different sections according to their functionalities and appearance in different stage in existing systems. The study gives a fundamental idea and concept to produce robust speed limit sign recognition system with appropriate techniques and algorithms.

On the other hand, several problems are spot from this literature review. Firstly, the most of existing systems [2-4, 6-8, 13, 14, 16, 28] are only able to identify input patterns from fixed number of different types of speed limit sign. For example, system can identify the readings from 20 km/h, 30 km/h, 40 km/h, 50 km/h, 60 km/h, 70 km/h, 80 km/h, 90 km/h and 100 km/h will fail to recognise a valid 110 km/h sign in front. Secondly, a better recognition performance can be achieved by applying some tunings to current systems. For example, the detection of speed limit sign can be smoothed by searching for circular achromatic regions with appropriate size instead of searching for outer red circle. Furthermore, some small medium processes can be added to the system to improve the recognition accuracy (i.e. applying a local character segmentation step before passing characters to recognition stage to prevent mismatching caused by dirty-misconnection between digits.) Low flexibility is another common limitation of current systems. This is caused by the ANN algorithms or template matching because they require pre-training or collection of various sample templates. These algorithms can be replaced by some others which don’t require these pre-works (i.e. scan-lines based algorithm) to overcome this drawback.

Last but not least, the less portability is a common issue existing systems. Although the latest palm size device (i.e. cellular phone) is much powerful than before and they are holding more market share, none of current speed limit sign recognition system is designed for, or implemented on Smart phone platform (i.e. Google Android platform). Since 1GHz CPU frequency, 512 MB memory and 5 Mega pixel digital camera is the current hardware standard of Android Smart phone, there is motivation to implement real-time speed limit sign recognition system which running on Smart phones.

In conclusion, the State of the Art shows various techniques and potential space for development in this field. Moreover, to overcome the problems of existing systems, or at least reduce their effect becomes part of the goals of this research project.

# 3 System Design

## 3.1 Overview of the design

The result of State of the art study shows that all existing speed limit sign recognition system, regardless of what techniques they used, consist of three major processing stages, which are Colour segmentation, Speed limit sign detection and Recognition / result output stages. Hence, the product system of this research is also built based on this pattern (See figure 3-1).

After reading a new frame from the on-board camera or sample input video, the input frame will be segmented into numbers of smaller regions based on the colour information. The reason for applying this process is to reduce the effort wasted on searching speed limit sign over the entire image. Instead, the system only searches for signs within red colour regions, because the speed limit sign in reality is always surrounded by red colour.

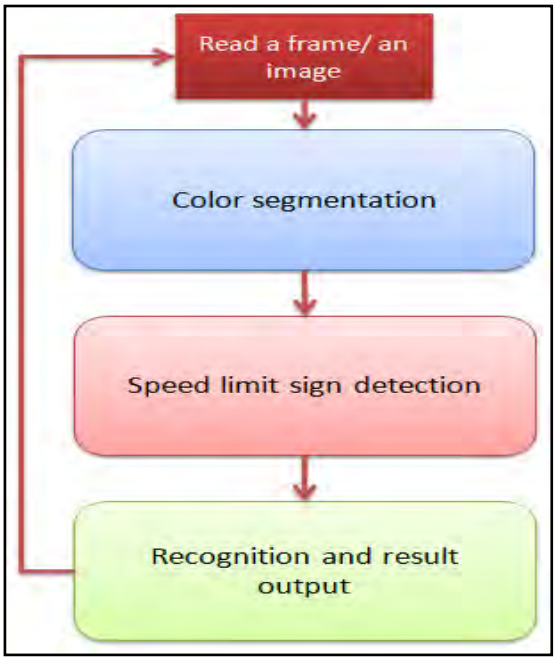



*Figure 3-1 System contains three main processing stages.*

The sign detection stage is the process that identifies speed limit signs from those candidate regions. It firstly identifies all circles and / or ellipses from these regions and then, for any circle and ellipse detected, the pixels within these circles will be binarized with black and write colours only. The pixel with RGB (Red, Green and Blue colour) value above the threshold will be marked with black colour. Otherwise, it will be marked with white colour. This will distinguish characters on (and /or contents of) the sign from the background because all speed limit readings are written in black colour on an ellipse in white. The sign detection is then applied on the binarized region to identify whether it is a speed limit sign by checking its visual features. For example, if there is no black blob found on an ellipse, this ellipse is considered as a non-speed limit sign object, and will be discarded. Only those ellipses with similar visual features with speed limit signs survive as candidates of the recognition stage.

In the last processing stage of the system, the optical digit character recognition is applied to identify digit readings from those black blobs on candidate ellipses. The results of readings of recognition will be evaluated before output. All illegal readings (i.e. “05”, “250”, “1”, any non-digit character, or readings with probabilities lower than the threshold) will be rejected and discarded. When the final legal speed limit reading is generated, driver will be informed via both visual and acoustic signals.

In general, there are two most important tasks for the proposed system, which are eliminating all fake speed limit sign objects in the scene successfully and recognizing the specified speed limit on the sign correctly. To achieve these tasks, each processing stage is divided into several sub-steps (See figure 3-2 below).

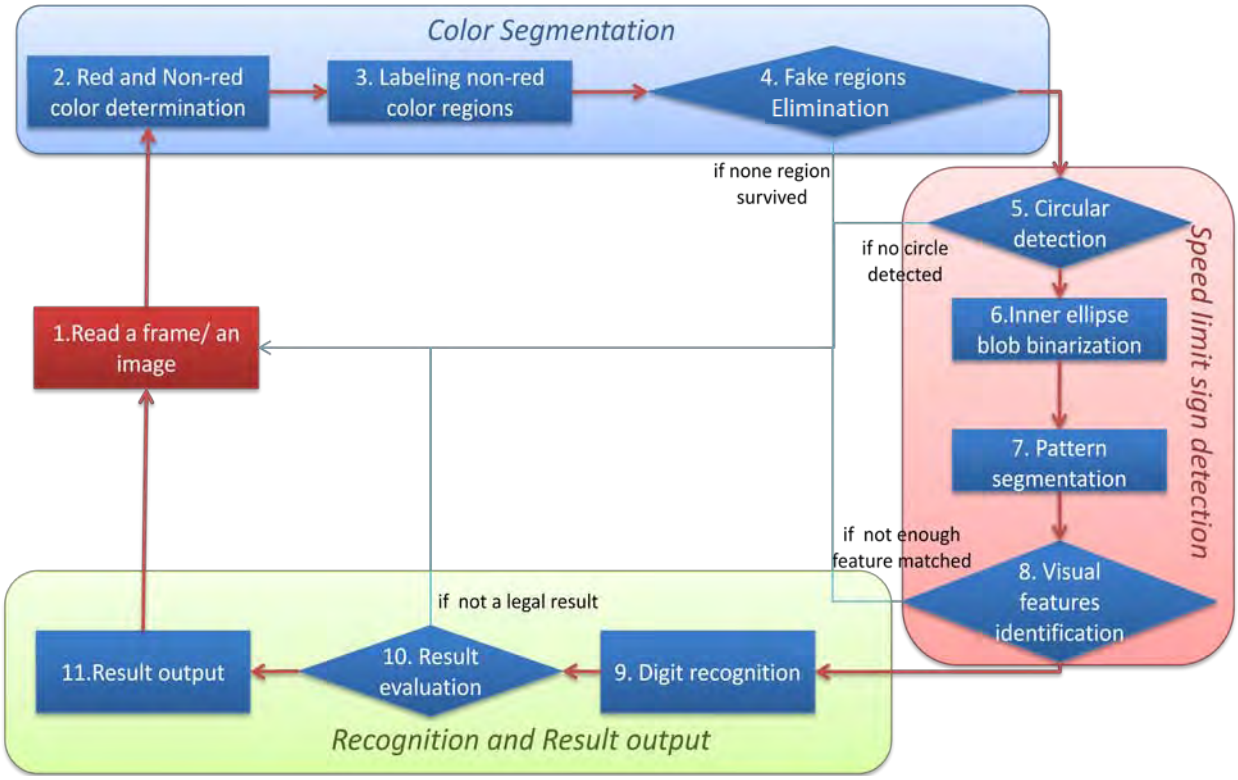



*Figure 3-2 A general flow chart of the proposed system*

These eleven critical steps together compose a full processing circle of a speed limit sign recognition routine. The steps in rectangle are regular steps that pass their results to subsequent steps. Those steps in diamonds are steps which judge whether proceeding to next step or aborting current routine by reading the next video frame. Routine aborted when no legal candidate or result to proceed to next step. In other words, no speed limit sign is detected in the scene.

Descriptions of the designs and techniques used in each of these steps are given in the preceding sections.

## 3.2 System component

### 3.2.1 Colour segmentation

As descriptions in previous section, the colour segmentation is used to distinguish regions in interested colour from those uninterested regions. In this case, regions surrounded by red colour are considered as potential candidate regions which may contain speed limit sign. (See figure 3-3).

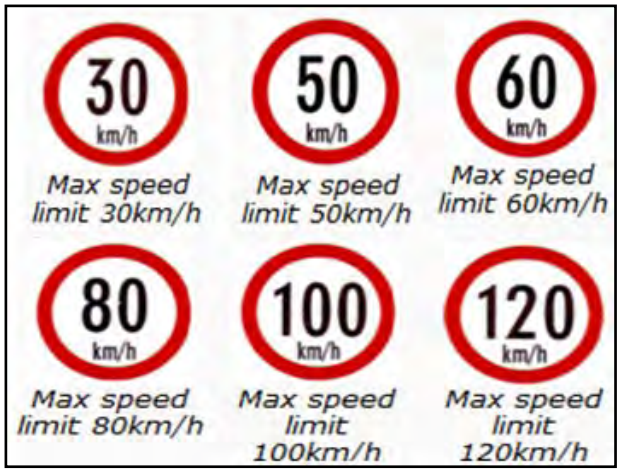



*Figure 3-3 Some typical speed limit signs in Ireland*

In order to achieve this, the input sample image will be firstly converted into YCrCb colour space, which is one of the most widely used colour space in computer vision field [2, 8, 12, 14]. The reason why this colour space was chosen is because it separates luminance from chrominance (lightness from colour). This will increase resistance to variation of lightness. Another advantage of this technique is the equation (5) of converting image from RGB colour space to YCrCb space is unique and quite straightforward [2]:

$$\begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (5)$$

Hence the Y coordinate is strongly dependent on brightness, the red colour segmentation is only considering the Cr (red-difference chroma) and Cb (blue-difference chroma) values. Based on previous researches and testing, the distribution of red colour in YCrCb space was shown in figure 2-3 in Chapter 2. According to researches of paper [2, 8], a pixel can be considered as a red pixel if its Cr, Cb values are meeting the following conditions:

$$\begin{cases} 0.5012 \leq P_{(Cr)} \leq 0.851 \\ \text{and} \\ 0.3811 \leq P_{(Cb)} \leq 0.5192 \end{cases} \quad (6)$$

Figure 3-4 shows that a good result of red colour detection in YCrCb colour space can be obtained according to (6).

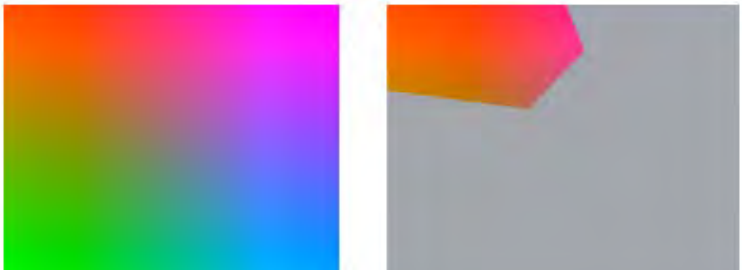


*Figure 3-4 Original image (left) and the result (right) after red colour detection.*

To separate red colour regions from non-red ones, the pixels meet condition (6) will be marked as black colour, and all other areas are marked as white (See Figure 3-5).


*Figure 3-5 Two original images (left side) and the result images (right side) which mark red component in original images with black colour.*

After red components have been separated from non-red components, non-red regions are in white and red regions are in black. For this system, a non-red region implies a white area which every pixel is connected with each other and not been isolated by any black (red) pixel. For example, for those two result images in Figure 3-5, both of them contain two non-red regions – the inner white ellipse and the region contains residual white pixels.

As the target of segmentation stage is to identify different non-red colour regions, the system will label them with different colours – one colour for one region and the RGB value of that colour will be used as the identification of the corresponding non-red regions. To achieve this, the connected component analysis algorithm [30] is applied to scan the result binary image (image with black and white colour only) with an 8-adjacency mask. (See figure 3-6). The central pixel 0 is considered to be adjacent to all other 8 pixels. The mask will go through the binary image row by row and column by column (within each row).

| | | |
|---|---|---|
| 2 | 3 | 4 |
| 1 | 0 | 5 |
| 8 | 7 | 6 |

*Figure 3-6 An 8 adjacency mask.*

For every pixel 0 in the mask, if it is a white pixel (indicating a non-red pixel in original image), all its previous adjacent pixels (pixel 1 ~ 4) will be checked. If all these pixels are background pixels (pixels in black colour), a new colour label is assigned to pixel 0. Otherwise, pixel 0 is assigned with a colour label picked from one of its previous adjacent pixels to show that they are connected together. Meanwhile, if any other previous adjacent pixels have different labels, those labels are noted as equivalent colour labels. After going through the entire image, equivalent labels are set to same with changing their colour value to same (See figure 3-7). Hence, non-red component regions are labelled with unique colour labels respectively. Colour values (RGB value) of these labels are stored for later use.

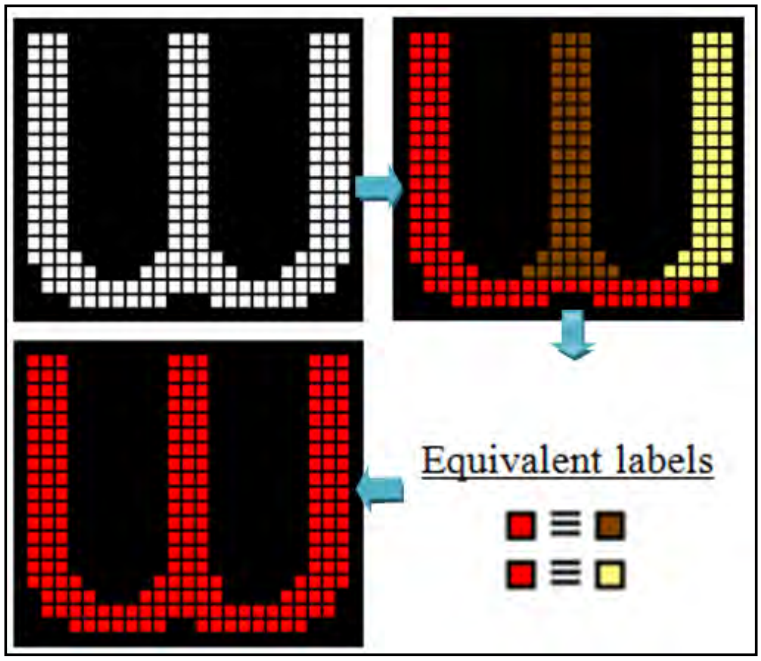



*Figure 3-7 Connected component labelling algorithm. Binary image (upper left) has been scanned with 8-adjacent mask (upper right) and all equivalent labels found (bottom right) are set to same colour (bottom left).*

These labelled regions will be passed to the next stage to check whether they contain speed limit sign. However, before going into the sign detection stage, the size of these regions will be checked to eliminate fake regions which are regions with inappropriate size. During colour labelling process, the width and height of each region are calculated. Subtracting region's left most column index from its right most column index gives us the width of the region and subtracting the top row index from bottom row index gives the height of the region. Only those regions with width and height fit the condition (7) can go into the detection stage:

$$\begin{cases} W_{\min} \le W_{\mathrm{r}} \le W_{\max} \text{ and } H_{\min} \le H_{\mathrm{r}} \le H_{\max} \\ \text{and} \\ \mathrm{abs}(W_{\mathrm{r}} - H_{\mathrm{r}}) \le T \end{cases} \quad (7)$$

Where $W_r$ and $H_r$ are width and height of the region. $W_{min}$, $W_{max}$, $H_{min}$ and $H_{max}$ are threshold values indicate minimum and maximum of an appropriate region width and height respectively. T is the threshold of maximum difference which region's width and height can have.

The first stage of process is ended with passing candidate regions to detection stage, with none red and / or fake regions in them. It smoothes and enhances the performance of processes in the following stage. For example, all red objects (i.e. red billboards and buildings etc.) and non red objects with inappropriate size (i.e. sky, road surface etc.) in the scene are discarded before further process.

### 3.2.2 Speed limit sign detection

The main task of speed limit sign detection stage is identifying speed limit signs within those candidate regions got from colour segmentation stage. This stage consists of 3 processes, which are circular detection, content extraction\ segmentation and sign identification.

Circular detection is the process which checks whether a candidate region is a circle \ ellipse. Only circular candidate regions can go through to further processes and all non-circular regions are discarded.

During State of the art studying period, I found that some of existing systems [2, 7, 8, 12, 16, 18] use Template matching technique to detect outer red circular pattern of the sign. However, these systems could fail when a speed limit sign appeared on the red background (i.e. in front of a red building, see figure 3-8) because the red circle of the sign is melted into the background and thus does not match with the template.


*Figure 3-8 Template matching technique could fail to detect red circle surrounds the speed limit sign when the red circle melted into the background red area (from left to right: original image, binary image after red colour extraction and the template image).*

Hough-based circular transform is another traditional approach to detect circular objects in digital images [9, 15, 22]. However, latest researches [4-6] considered Hough transform as computationally complex and memory hungry for real-time processing over large images.

In order to overcome those drawbacks, the proposed system applies another robust approach, called Linear Support Vector Machine (LSVM) [5, 26], to detect circular object. The system concentrated on detecting non-red ellipse from non-red regions instead of searching outer red ring. Top contour and bottom contour of each labelled candidate regions are scanned to form 2 DtBs (Distance to Boundary), top DtB and bottom DtB. Every scan starts from the upper left

corner of the colour segmented image. For every second column, go down the column row by row to detect the first encountered pixel labelled with the target label colour. If no target region pixel found in the column, moves to scan the second next column. If a target region pixel is detected, its row index is recorded and then moves to scan the second next column. The scan stops when no target region pixel found in any following column, or reached the right most column of the image. Thus, the top DtB of the target region is formed by recorded row indices. In other words, top DtB implies the top contour of the region. The bottom Dtb of the region is formed in the same way with starting from the lower left corner and go up every second column. An example of this algorithm can be found in figure 3-9.

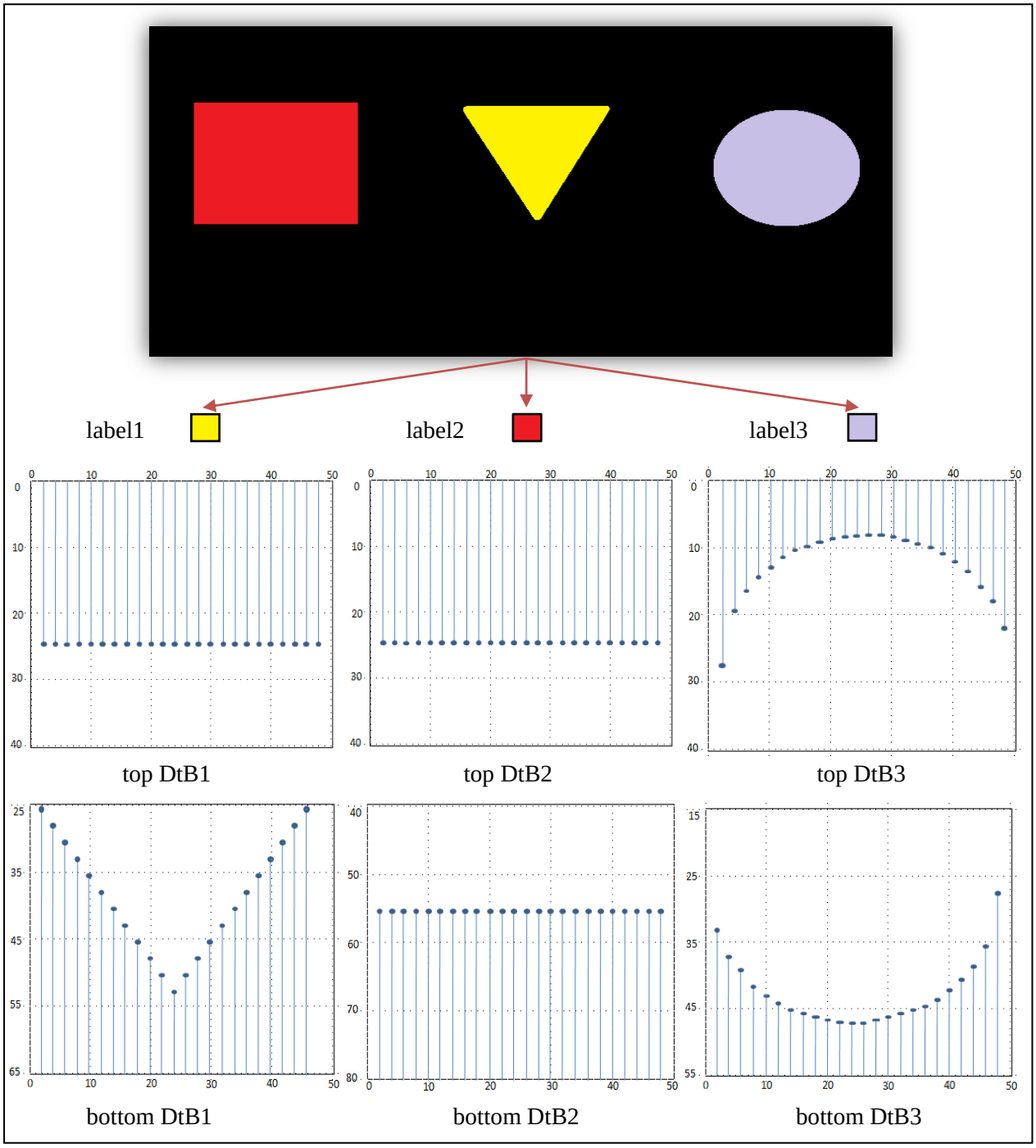



*Figure 3-9 An example of generating DtBs of regions*

Every vertical line in a DtB is a linear vector representing the distance to the boundary of the image. LSVM takes DtBs as input to decide whether the region is a circle by analyzing difference between two neighbouring vectors, symmetry and peaks of DtBs. The system output will be a probability of circle according to similarity between DtB pattern and circular arc. If the probability is greater than the threshold (i.e. 80%), the corresponding non-red region is considered as a circle \ ellipse. Thus, the system is now able to identify non-red ellipses surrounding by red colour.

Unlike some systems in previous researches [8, 16] which could confuse a speed limit sign with other regulatory traffic signs, the proposed system is designed to discard any other traffic sign but speed limit sign. After non-red ellipses have been identified, some of regulatory traffic signs can be eliminated (if any appears in the scene). However, we can't be sure that speed limit signs are located because some other traffic signs which have high similar modality with speed limit sign in reality haven't been eliminated yet (see figure 3-10). In order to make sure that only speed limit sign can be detected by the proposed system, the content of identified non-red ellipse is analysed.


*Figure 3-10 After non-red ellipses have been identified, regulatory traffic signs in the first two rows can be eliminated by the system because the ellipses are split into two different regions by the red backslash strip.*

Firstly, candidate ellipses are extracted from original image (according to the height and width of the corresponding region), and convert it to gray-scale (single channel) image. Then, Optimal Thresholding (Ridder Calvard method) [31] is applied to obtain the binary image of the ellipse. Optimal Thresholding selects a threshold value that is statistically optimal, based on the contents of the image. The algorithm of this method is:

1. Set $T^0 = T^{initial}$, $t = 0$.
2. At step $t$, compute $\mu^t{}_B$ and $\mu^t{}_O$ as the mean background and object gray-level respectively, where segmentation into background and object at step $t$ is defined by the threshold value $T^t$ determined in the previous step.

$$\mu^t{}_B = \frac{\sum_{(i,j)\in \text{background}} f(i,j)}{\#\text{background\_pixel}} \qquad (8)$$

$$\mu^t{}_O = \frac{\sum_{(i,j)\in \text{object}} f(i,j)}{\#\text{object\_pixel}} \qquad (9)$$

3. Set $T^{(t+1)}$ $$T^{(t+1)} = \frac{\mu^t{}_B + \mu^t{}_O}{2} \quad (10)$$

   $T^{(t+1)}$ is the updated threshold.
4. If $T^{(t+1)} = T^t$, halt, and output $T^t$, otherwise $T^t = T^{(t+1)}$ and return to step 2.

In practice, algorithm above is applied to the histogram (with range 0~255) of the gray-scale image. The output threshold value is used to distinguish background from object by changing pixel below $T^t$ to white (background) and pixel above to black (object). Hence, the binary image of the content pattern is generated. The result binary image will then be segmented via the connected component analysis algorithm described in "Colour segmentation" section. The content patterns (in black colour) on the ellipse will be segmented and labelled with different colour labels according to their connectivity (see figure 3-11).

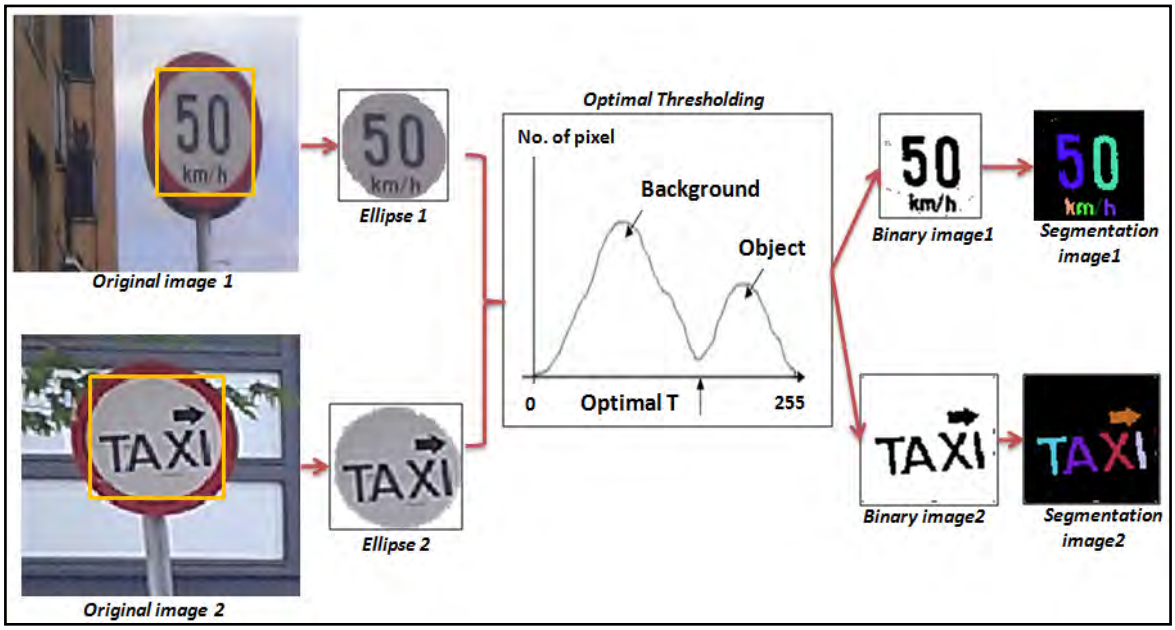



*Figure 3-11 Component segmentation of two ellipses: two ellipses are extracted from original images. Binary images are generated via Optimal Thresholding respectively and components (black blobs) of images are labelled with different colour to form corresponding segmentation images.*

Location and amount of labelled blobs in segmentation image will be checked next to identify speed limit sign. In the system, every segmentation image is divided into two sections, which are a larger upper section (from top boundary to 1/3 of image height) and a smaller lower section (from 1/3 of image height to bottom). If there are 1 ~ 3 large labelled blobs found in upper section with roughly same height, and a pattern is found lying at the middle of the lower section which consists of some smaller blobs (or one larger blob),

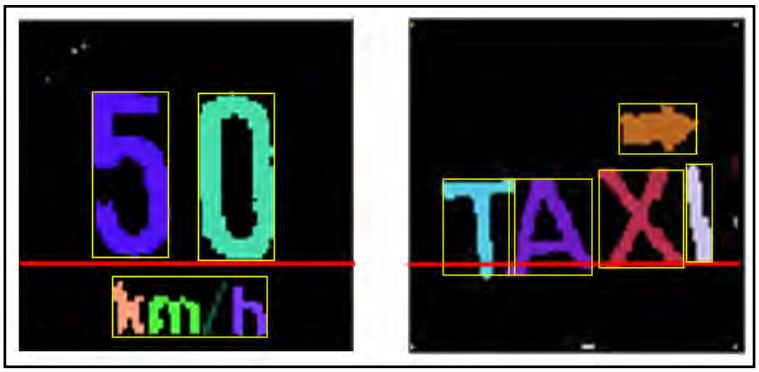


*Figure 3-12 Segmentation images are checked to identify speed limit sign. Red line is the boundary of upper and lower section.*

the corresponding ellipse is considered as a speed limit sign. For example, in figure 3-12, the segmentation image on the left has the same modality as the speed limit sign, and is considered as a speed limit sign. On the other hand, the image on the right does not contain any flat large pattern (i.e. “Km /h”) in the lower section and too much blobs (5 blobs) have been found in the upper section. Therefore, the ellipse is not a speed limit sign and will be discarded.

By applying above scheme, most of traffic signs and other objects with high similarity of a speed limit sign are eliminated. Blobs in the upper section (i.e. numerical digits) of those survived ellipses will be sent to recognition stage.

### 3.2.3 Recognition and result output

As the last stage of the system, Recognition stage is responsible to recognise numerical digit on the sign and cue the driver with visual and acoustic signals. Artificial Neural Network (ANN) [2, 3, 8, 14, 16] and Template Matching based [12, 15] are two traditional approaches to achieve digit recognition in this field. However, there are still several problems with these techniques. The ANN requires large amount of training to operate before applying. It also requires high processing time for a well-trained robust neural network. For Template Matching, the performance is varying in different fonts, damage and rotation of speed limit sign. The recognition speed performance is also poor. Furthermore, the Template Matching will definitely fail to recognise an infrequent speed limit sign if no such a template is in its database (see figure 3-13). In order to overcome these problems, a new scan-line based digit recognition algorithm is proposed. The original version of scan-line based algorithm is reviewed in the “State of the Art” chapter of this thesis. The main advantage of this method is that it is able to run in real time and good resistance to variation of font and image quality. This technique also does not require any pre-processing or training. The drawback of the original version is that the output of recognition is binary result, which means either true or false (i.e. the character is either “8” or not). In other words, the algorithm does not involve any level of tolerance. This could become a potential issue which may cause misclassification. The proposed system overcomes this by introducing probability scheme to recognition algorithm.


*Figure 3-13 An occasional “8 Km/h” speed limit sign.*

The new scan-line based algorithm scans character \ pattern with three vertical scan lines instead of with one in the original system. The three vertical scan lines are: middle line, left line (middle column of left boundary and middle line) and right line (middle column of middle line and right boundary). Pixels at the border between object and background which encountered by scan lines

will be marked, and especially, those pixels encountered by middle line are marked as Critical Points (CPs). The number of border pixels encountered by each scan line, together with their location (i.e. row index) is stored as one of the "*Features*" of the blob. After this Feature is obtained, the inner edge tracing is applied according to no. of CPs.

Inner edge tracing is the process that traces the inner edge of the pattern to decide whether the pattern is closed. A tracing is successful if it has traced around the inner edge and back to the start pixel. As CPs always appear as pairs, the amount of CP is always an even number. If there are 4 CPs, inner edge tracing is applied between C2 and C3. If there are 6 CPs, an extra tracing is applied between C4 and C5. The result of tracing is recorded as one of the "*Features*". For example, a successful inner edge tracing between C2 and C3 will mark "*Feature 2*" to "*true*" and an unsuccessful tracing will mark "*Feature 2*" to "*false*". Similarly, "*Feature 3*" will be marked according to the result of tracing between C4 and C5. After this, any "*false*" result will trigger the right-side edge detection with a horizontal scan-line start from the middle point between the same CPs. (i.e. if inner tracing has failed between C2 and C3, the right-side edge detection will start at the middle point between C2 and C3). The horizontal scan-line goes straight to the right, and it stopped when meeting an edge pixel of the pattern, which indicates a successful detection; or meeting the right boundary of the pattern, which indicates a failed detection. The result of edge detection is also marked as one of "*Features*" ("*true*" for success and "*false*" for failure). These discriminating "*Features*" are the keys to the recognition being successful in classification.

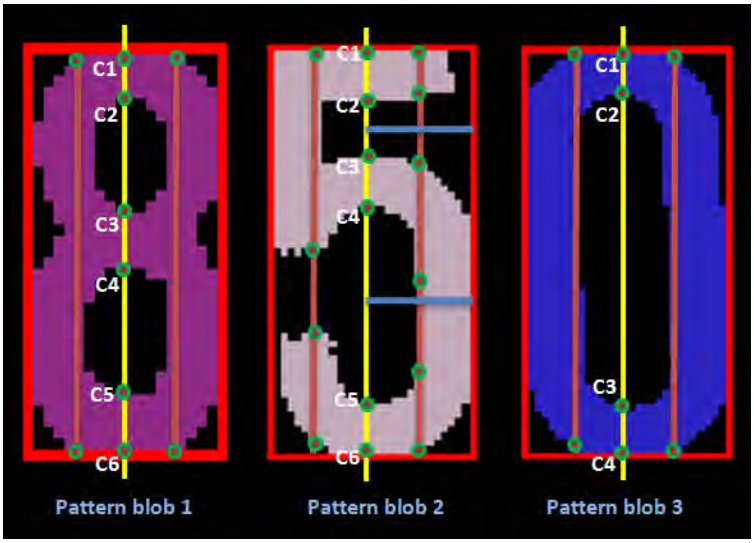



*Figure 3-14 An example shows three blobs after being scanned by scan-lines.*

Figure 3-14 shows that three pattern blobs after being scanned by scan-lines. Outer boxes in red show the boundary of each blob. Scan-lines in yellow are middle lines and two scan-lines in rosé red represent left and right lines of each blob. Every border point is marked with "●" and the CPs are labelled with C1~C6. Two horizontal lines in blue are horizontal scan-lines for right edge detection after both C2-C3 and C4-C5 inner edge trackings failed. "*Feature*" tables of these patterns are filled and shown in Table 1 below:

| | # CP | # LBP | # RBP | IET1 | IET2 | RED1 | RED2 | CCB | LCB | RCB |
|---|---|---|---|---|---|---|---|---|---|---|
| Blob 1 | 6 | 2 | 2 | True | True | N/A | N/A | True | True | True |
| Blob 2 | 6 | 4 | 6 | False | False | False | True | True | True | True |
| Blob 3 | 4 | 2 | 2 | True | N/A | N/A | N/A | True | True | True |

# LBP = No. of Left Border Point
# RBP = No. of Right Border Point

IET1 = Inner Edge Tracing (C2-C3)
IET2 = Inner Edge Tracing (C4-C5)
RED1 = Right Edge Detection(C2-C3)
RED2 = Right Edge Detection(C4-C5)

CCB = (first and last)CPs are Close to Boundary
LCB = (first and last) LBPs are Close to Boundary
RCB = (first and last) RBPs are Close to Boundary

*Table 3-1 Feature table of the blobs in Figure 3-14.*

Value of each field in the table will be matched with the corresponding field in *"Features of Standard Numerical Digit"* table (see Table 2) to generate the result of the classification with a probability. The probability is changing according to whether a field is matched with the corresponding field in Standard Feature Table. A matched pair will increase the probability of the corresponding numerical digit, and decrease all probabilities of unmatched digits. One thing I would like to point out is that the fields in red colour indicate "*crucial features*" which carry more weight than other features.

| Digits | # CP | # LBP | # RBP | IET1 | IET2 | RED1 | RED2 | CCB | LCB | RCB |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4 | 2 | 2 | True | N/A | N/A | N/A | True | True | True |
| 1 | 2 | 2 | 2 | N/A | N/A | N/A | N/A | True | False | True |
| 2 | 6 | 4 | 4 | False | False | True | False | True | True | True |
| 3 | 6 | 4 | 4 | False | False | True | True | True | True | False |
| 4 | 4 | 2 | 2 | True | N/A | N/A | N/A | False | False | True |
| 5 | 6 | 4 | 4 | False | False | False | True | True | True | True |
| 6 | 6 | 2 | 4 | False | True | False | N/A | True | True | True |
| 7 | 4 | 4 | 2 | False | N/A | True | N/A | False | True | False |
| 8 | 6 | 2 | 2 | True | True | N/A | N/A | True | True | True |
| 9 | 6 | 4 | 2 | True | False | N/A | True | True | True | True |

*Table 3-2 The Feature table of Standard Numerical Digits.*

For example, the values in every field of row "Blob 1" (see Table 1) made perfect match with fields in row "8", which means that the corresponding blob is classified as digit "8" because full match makes the highest probability. Meanwhile, along with the increase in probability of digit "8", probabilities of other digits which have been found in unmatched pairs, are decreased: probabilities of digit "0", "1", "4" and "7" are decreased because they cannot match with field "#CP"; probabilities of digit "1", "2", "3", "5", "6" and "7" are decreased because they cannot

match with field “IET1” etc. It turns out that the blob1 is classified as “8” with 100% probability. The blob3 is classified as digit “0” (also with 100% probability) in the same way. For blob 2, although there is no perfect match with any digit, the blob is recognised as “5” with a significant high probability (i.e. 95%) because there is only one unmatched pair in uncritical feature field “#RBP”.

However, as the main target of this thesis is to produce a robust system with recognition speed limit sign from video, the final output of the system should not depend on the recognition result of one single frame. Instead, an accumulated probability is introduced to calculate the final output. The final output depends on the probability of all possible outcomes accumulated from the reference outcome (i.e. speed limit *X* is identified at frame *n*) to the last outcome (i.e. frame *n+10*). In other words, for any speed limit sign *X* detected with probability *Y* at frame *n*, keeps tracking the sign in subsequent frames and accumulating the probability until : (1). The accumulated probability reaches the threshold *T* (i.e. *T=90%*) or (2). 10 frames have been processed since frame *n*. If the accumulated probability *f* is greater than *T*, the speed limit *X* is considered to be the final output. The equation of accumulated probability is:

*At any frame f,* $n \leq f \leq n+10$

$$P_f(X) = P'_f(X) \times \alpha + P_{f-1}(X) \times (1-\alpha) \qquad (11)$$

Where $P_f(X)$ is the accumulated probability of speed limit sign *X* at frame *f*. $P'_f(X)$ is the probability of *X* recognised at frame *f*. $\alpha$ is the weight factor and $\alpha = 0.4$.

After a sign *X* is considered as the final result of the recognition, the reading of *X* will be checked before cueing the driver. If the reading is an unrealistic speed limit (i.e. “05 Km/h”, “210 Km/h” etc.), the result will be discard and the driver will not be notified. Otherwise, the latest speed limit, together with a screenshot (see figure 3-15), is displayed on LED screen beside the dashboard and the cue tone is played via onboard speaker. In this way, the driver will be notified with both visual and acoustic signals.


*Figure 3-15 Final result of the recognition, together with a screenshot will be displayed on the screen.*

# 4 Implementation and Results

The main function modules and all critical algorithms \ techniques have been introduced in the last chapter. This chapter presents the implementation of function modules which comprise the proposed system. The proposed system is developed in Microsoft Visual C++ with Microsoft Visual Studio 2008 environment. An open source computer vision library – OpenCV 2.1 is utilised to help reading video sequences \ images from sample database.

The general class diagram of the system is presented in Figure 4-1 below. This diagram shows the major classes, crucial functions contained in these classes and how classes are linked with each other.

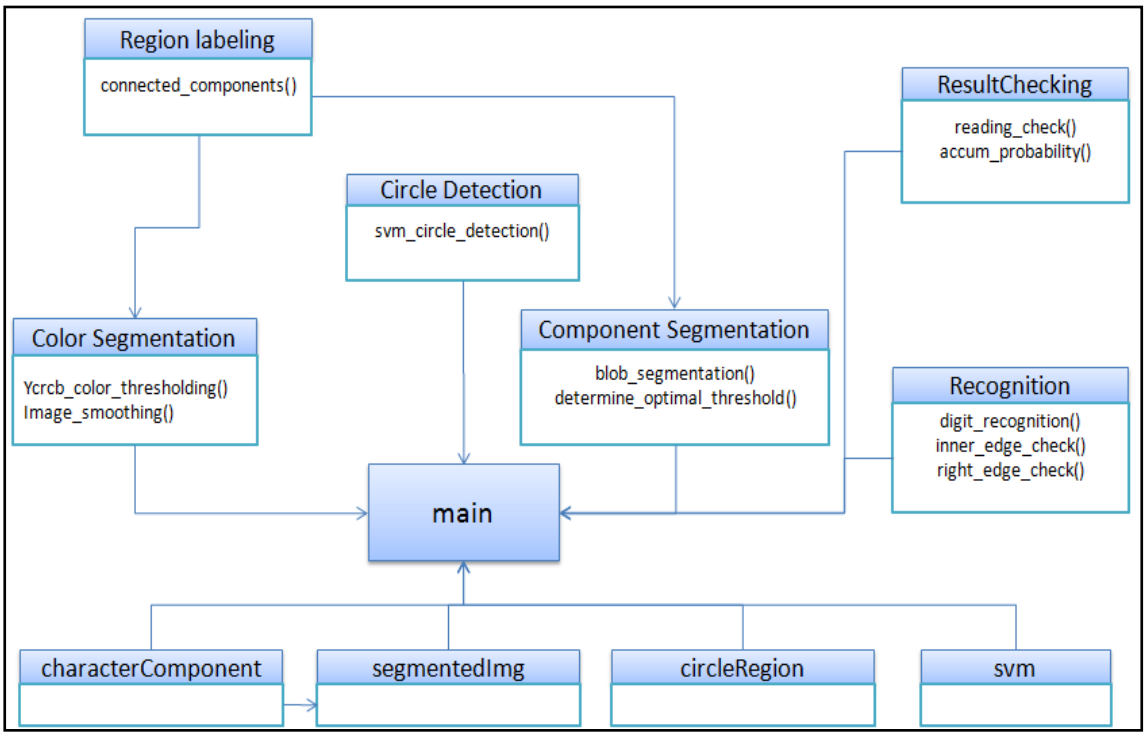



*Figure 4-1 A general class diagram of the proposed system.*

Four empty classes at the bottom of figure 4-1 are four crucial data structures which I create to store result data after being processed by functions and pass to subsequent functions. `svm` is the structure holds data of linear vectors which form DtBs. Field in this structure includes a 2 dimensional array holding index of every vector and row index of the vector, and an array holding colour label of the target region (R,G,B). `circleRegion` holds data of circular region which passed through circular detection. Filed in `circleRegion` include 4 integers holding boundary of the region (top row, leftmost column, width and height), an integer holding probability of the region is a circle and an array holding colour label of the region. `segmentedImg` holds a reference of the segmented image of the ellipse, a rectangle which

identifies the border of the ellipse, and a vector (container) of characterComponents. characterComponent is the structure which holds boundary (top row, right column, bottom row and left column), probable digit character, probability array of all possible digits (0~9), array of CPs, array of LBPs and RBPs data of a pattern blob in segmented ellipse.

Instances of these structures are created (released and reinitialised) in Main class. The Main class is the entry point of the system. It handles passing variables and instances to functions and managing of process sequence. It is also responsible for presenting recognition result to the driver. The implementation of major functions in every function module is described in the following paragraphs of the chapter, following the progress of recognition.

## Colour Segmentation

The colour segmentation is the first process of the system. In order to extract and label all non-red segments from the original input image \ frame, the image needs to be converted from RGB to YCrCb colour space and binarised first. Function Ycrcb_color_thresholding() has been built for this. According to the conversion matrix (5) in section 3.2.1, the conversion is straightforward (see figure 4-2).

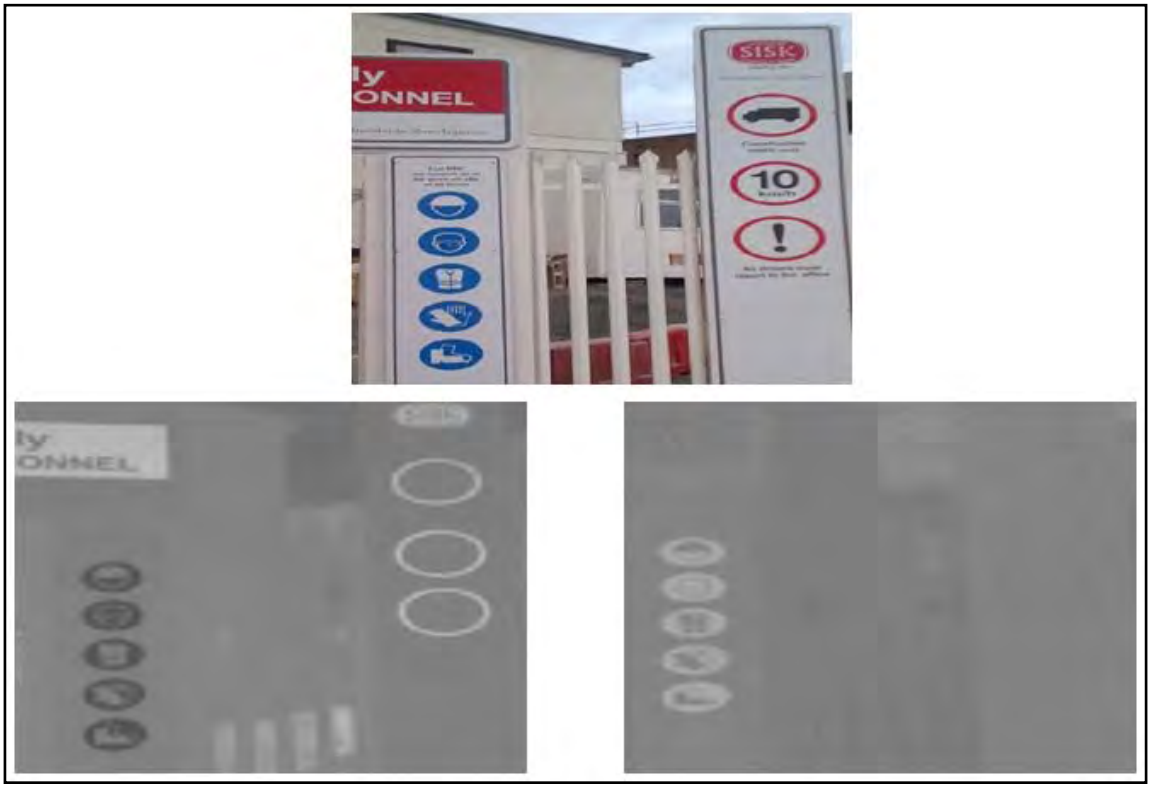



*Figure 4-2 An original image (top), the Cb channel (bottom left) and the Cr channel (bottom right) after converting it to YCrCb space. The brighter area identifies red-like region (larger blue-difference chroma) in Cb channel and blue-like region (larger red-difference chroma) in Cr channel.*

However, the condition equation (6) which decides whether a pixel is red in YCrCb space doesn't work very well because it is too sensitive and cannot identify red colour when it is much brighter than common. This could cause miss-segmentation because of unclosed regions. Hence, the equation (6) is replaced by the modified equation (12) below:

$$\begin{cases} 0.54 \le P_{(Cr)} \le 0.87 \\ \text{and} \\ 0.37 \le P_{(Cb)} \le 0.52 \end{cases} \quad (12)$$

The new threshold values are obtained by experiment and observation. The binarization result is much better than using equation (6).

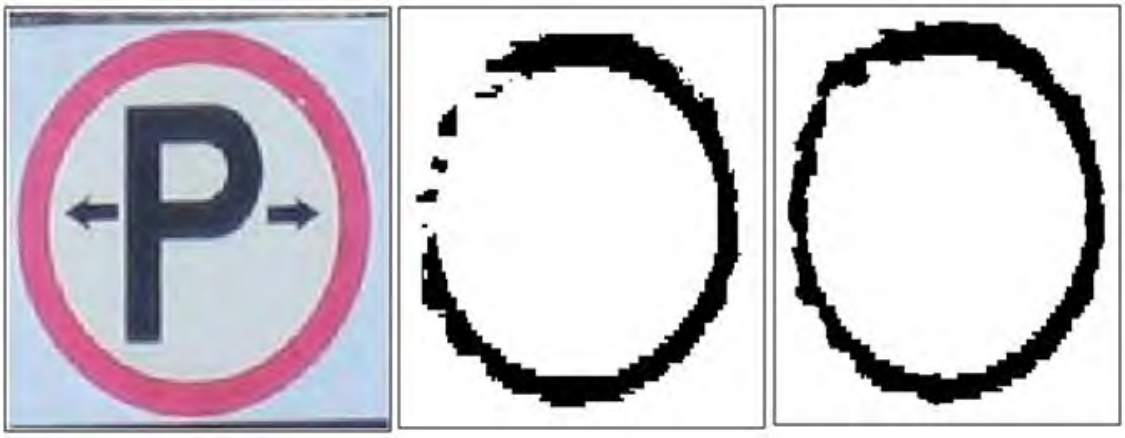


*Figure 4-3 Original image (left) and its two binary images generated by equation (6) from previous researches (middle) and the new equation (12) (right) respectively.*

The pseudocode of Ycrcb_color_thresholding() is:

```
Ycrcb_color_thresholding(img)
    for row in img:
        for column in img:
            pix = img [row][col]
            r = pix[red]
            g = pix[green]
            b = pix[blue]

            Cr = (0.5*r) + (-0.419*g) + (-0.081*b) +128
            Cb = (-0.169*r) + (-0.331*g) + (0.5*b) +128

            if (Cb >= 0.37*255) && (Cb <= 0.52*255)
                if (Cr>= 0.54*255) && (Cr<= 0.87*255)
                    img[row][col] = BLACK
                else
                    img[row][col] = WHITE
    return img
```

Hence, non-red regions are extracted from input image / frame and presented as white regions in result binary image. Before labelling regions, the binary image is smoothed by filling small black holes and then eliminating small white regions. This process is known as “Closing and Opening” [32]. Opening and closing are two important morphology operators derived from the fundamental operations of Image Erosion and Image Dilation. An Opening is defined as an erosion followed by a dilation using the same structuring element for both operations. It is less destructive than erosion in general. It removes smaller details (i.e. salt noise), narrow features (i.e. bridge structures) and smoothes the object boundaries. On the other hand, a Closing is Opening performed in reverse – simply as a dilation followed by an erosion. It joins objects which are close together and fills in holes within objects. The system applies a Closing operation followed by an Opening to clean up binary image (see figure 4-4).

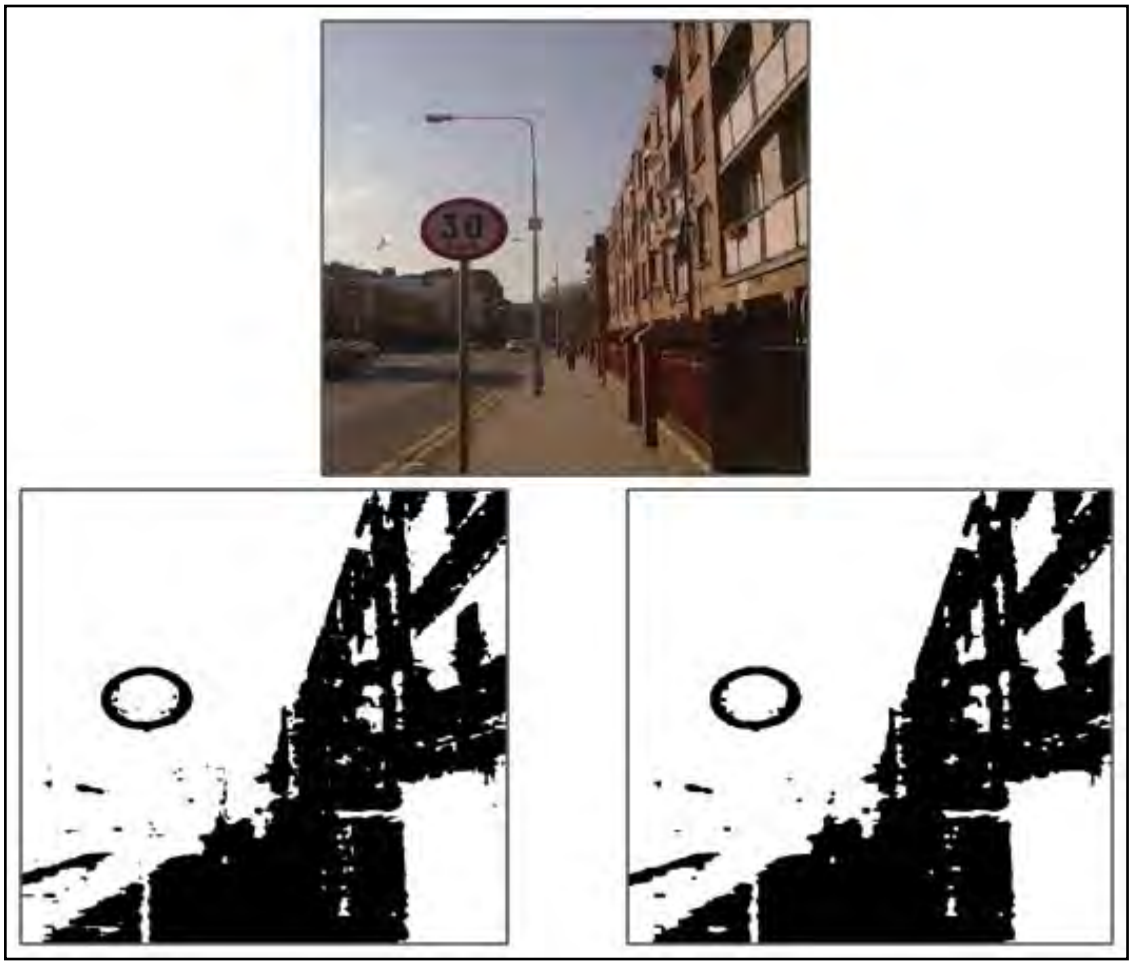



*Figure 4-4 An original image (top), its binary image (left) and the smoothed image (right)*

Scene in original image in figure 4-4 was dyed red by the sunset and causes many noises and small regions on its binary image. Closing and Opening operation removed most of those small details to smooth the subsequent Region labelling process. The implementation of Closing and Opening is achieved by using mathematical morphology library in OpenCV 2.1. As it is a subordination process, the pseudocode of `Image_smoothing()` is not presented in this thesis.

## Region Labelling

This process is used to detect connected regions in binary image and label every connected object regions with a unique colour label. Concept of this algorithm is introduced in previous chapter. The pseudocode of connected_components() is:

```
connected_components (binaryImg, nonRedSegImg)

    for row in binaryImg:
        for column in binaryImg:
            pix = img [row][col]
            if pix is a white pixel
                neighbour[4] = four previous adjacent pixels of pix.
                if all neighbour is unlabelled
                    pix.label = new label
                    label[count] = new label
                    nonRedSegImg [row][column] = label[count].color
                    count++;
                else
                    pix.label = neighbour's label
                    nonRedSegImg [row][column] = neighbour's label.color
                if any two neighbours have different label
                    mark neighbour label 1 equal to neighbour label 2
    for index in count:
        if label[count] is marked equal to label[count+1]
            label[count+1] = label [count]

    return label
```

Hence, the non-red regions are all labelled with different colour identifiers (see figure 4-5). The region segmented image and colour label array obtained from connected_components() are stored and will be used in subsequent processes.

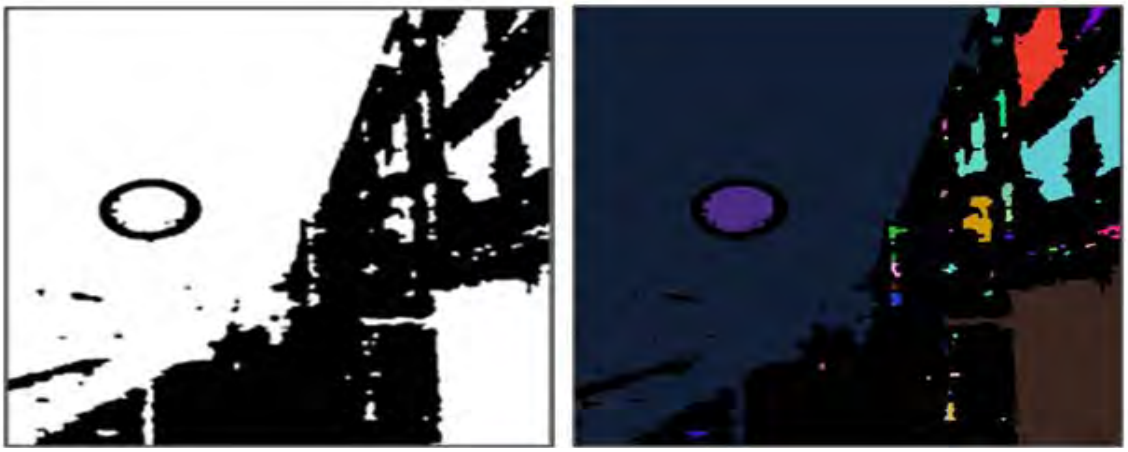


*Figure 4-5 A binary non-red image (left) and its labelled image (right)*

## Circle Detection

The LSVM algorithm is applied to the labelled image to extract non-red ellipses in the scene. Function svm_circle_detection() takes labelled image, colour label array, a vector container which will hold result circleRegions and a svm as function parameters. It generates the top DtB of every region and checks the contour to decide whether the top part of the region matches the feature of a circle. Then, does the same to test bottom DtB and makes the final decision. The pseudocode of the function is shown below:

```
svm_circle_detection(labelImg,label[],circleRegionVector, svm)
    for label in label[]:
        flag = false
        element_index = 0
        initialize(svm)
        // start to generate top DtB
        while NOT flag:
            flag = true
            for row in labelImg:
                pix = img [row][col]
                if element_index > MAX_NO_OF_ELEMENT   // 90
                    break
                else if pix = label
                    svm.element[element_index][0] = col
                    svm.element[element_index][1] = row
                    flag = false
                    element_index++
                    col+=2     //every second column
                    break
            if element_index is 0
                flag = false
                col+=2
        //start to check top DtB
        total_mark =100
        if svm.element[median][1] is NOT at the peak   // condition 1
            total_mark -= 5
                    .
                    .
                    .
                    .
                    .                                  // condition  8
```

```
//check the reading of total_mark
if total_mark >= MARK   // 80
    do the same to generate
        and test bottom DtB
//check the reading of total_mark again
if total_mark >= MARK   // 80
    circleRegion cRegion.label = label
                 cRegion.top = top
                 cRegion.height =   bottom – top
                 cRegion.width = element_index *2
                 cRegion.mark = total_mark
                 cRegion.label = label
    //mark the circle region on labelled image
    draw(red_rectangle, cRegion, labelImg)
    //store the circle region
    push(cRegion) to circleRegionVector
```

The other critical conditions which decide whether a region is circle are the symmetry of DtBs, if the differences between elements are evenly increasing (or decreasing), if the two feet of DtBs are the first and the last element etc. The result shows that this algorithm works well and faster than traditional Hough based circular detection (see figure 4-6).


*Figure 4-6 Two original images (left) and their labelled region images (right). Non-red ellipses are marked with red rectangles.*

As shown in Figure 4-6, although the ellipse in the bottom right labelled image is somewhat damaged due to the twilight, LSVM algorithm is still able to identify it as an ellipse correctly.

### Component Segmentation

After all non-red ellipses in the scene have been gathered into a vector, the next process is to segment and extract pattern blobs of them. Firstly, determine_optimal_threshold() function is applied to the histogram of ellipse's gray-scale image (i.e. single channel image) to generate the optimal threshold value. Pseudocode of this function is:

```
int determine_optimal_threshold (hist)
    int i = 0
    int last_threshold = 0
    for i to 255:
        total_sum += i* histogram value at i
        value_sum += histogram value at i
    new_threshold = total_sum / value_sum
    // start to calculate optimal threshold
    while new_threshold != last _threshold :
            last _threshold = new_threshold
            for j to last_threshold:
                background_sum +=j* histogram value at j
                backValue_sum += histogram value at j
            background_threshold = background_sum / backValue_sum

            for k = last_threshold to 255:
                object_sum +=k* histogram value at k
                objectValue_sum += histogram value at k
            object_threshold = object_sum / objectValue_sum
            // assign new_threshold
            new_threshold = object_threshold + background_threshold/2

return new_threshold
```

The ellipse is then binarized with this optimal threshold value to distinguish the pattern from background. Again, connected_components() in Region labelling class is applied to the binary image to label different blobs with unique colour. Ellipse is split into upper and lower sections and the distribution of blobs in these two sections are examined to check whether they match with the modality of a speed limit sign. If match, blobs in upper section are passed to recognition. Otherwise, discard the ellipse. Function blob_segmention() is the crucial method in Component segmentation class which implements segmentation process and calls

determine_optimal_threshold() and connected_components() functions. It takes the original image, labelled image and ellipse vector as its parameter, and returns a vector of segmentedImg which holds all essential data of upper section blobs and the segmented image. The pseudocode of blob_segmention() is shown below:

```
vector blob_segmentation (originalImg, labelImg,circleRegionVector)
    vector <segmentedImg> segmentedImgVector

    for circleRegion in circleRegionVector:
        corner_color = pixel at corner of circleRegion
        ellipseImg = get the ellipse from orinialImg and fill four corners
                     with corner_color
        convert ellipseImg to grayscaleImg
        histogram hist = createHist (grayscaleImg)
        //get optimal threshold
        threshold = determine_optimal_threshold(hist)
        binaryImg = Thresholded grayscaleImg
        for pix in binaryImg:
            pixelcounter++
            if pix is background pixel
                background pixel++
        if percentage of background pixel is in suitable range
            segmentedImg  segImg
            labelResult = binaryImg
            //get labelled image
            label = connected_components (binaryImg, labelResult)
            if label.size() >=2      // least 2 blobs in labelResult
                if blobs lay in the centre area of LOWER_SECTION
                and percentage of blobs is in suitable range
                    characterComponent  char
                    initialize (char)
                    segImg.character.push(char)
            if 1<=number of blobs in UPPER_SECTION <=3
                if all these blobs have roughly same height
                and NOT small blobs
                    //mark them on labelled segImgimage
                    draw(red_rectangle, char, segImg)
                    //push segImg to vector
                    segmentedImgVector.push(segImg)

return segmentedImgVector
```

The qualified segmented ellipses (i.e. considered as ellipse of speed limit signs), together with blobs in their upper section are stored into the vector. These candidate blobs will be passed to Recognition stage to generate the reading of sign. Figure 4-7 below demonstrates the process of blob_segmention() function.

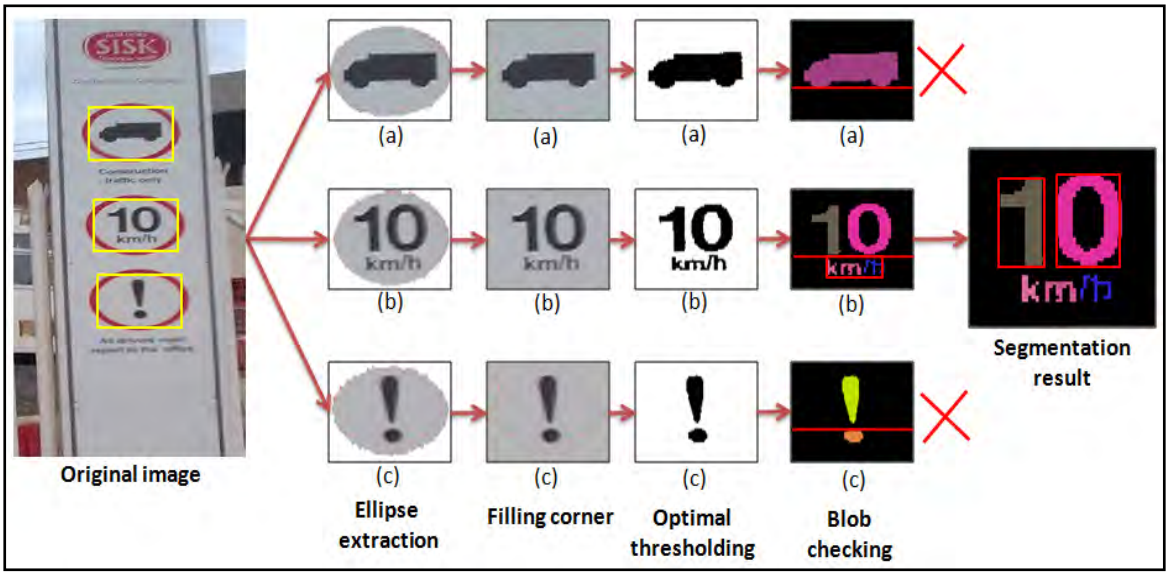



*Figure 4-7 An example of speed limit sign detection and fake ellipse elimination. (a) and (c) are discarded after Blob checking due to fail to detect the large pattern (implies a "Km/h") in the lower section.*

## Recognition

The Recognition class implements the proposed scan-line based digit recognition algorithm, which consist of digit_recognition (), inner_edge_check() and right_edge_check() functions. digit_recognition() function takes the vector of segmentated blobs as input. It scans each blob with scan-lines, calls inner_edge_check() and right_edge_check() functions to generate *features* of the blob. Finally, match blob *features* with *standard features* of digit character and generate the recognition result based on probability of matching. Pseducode of digit_recognition () is:

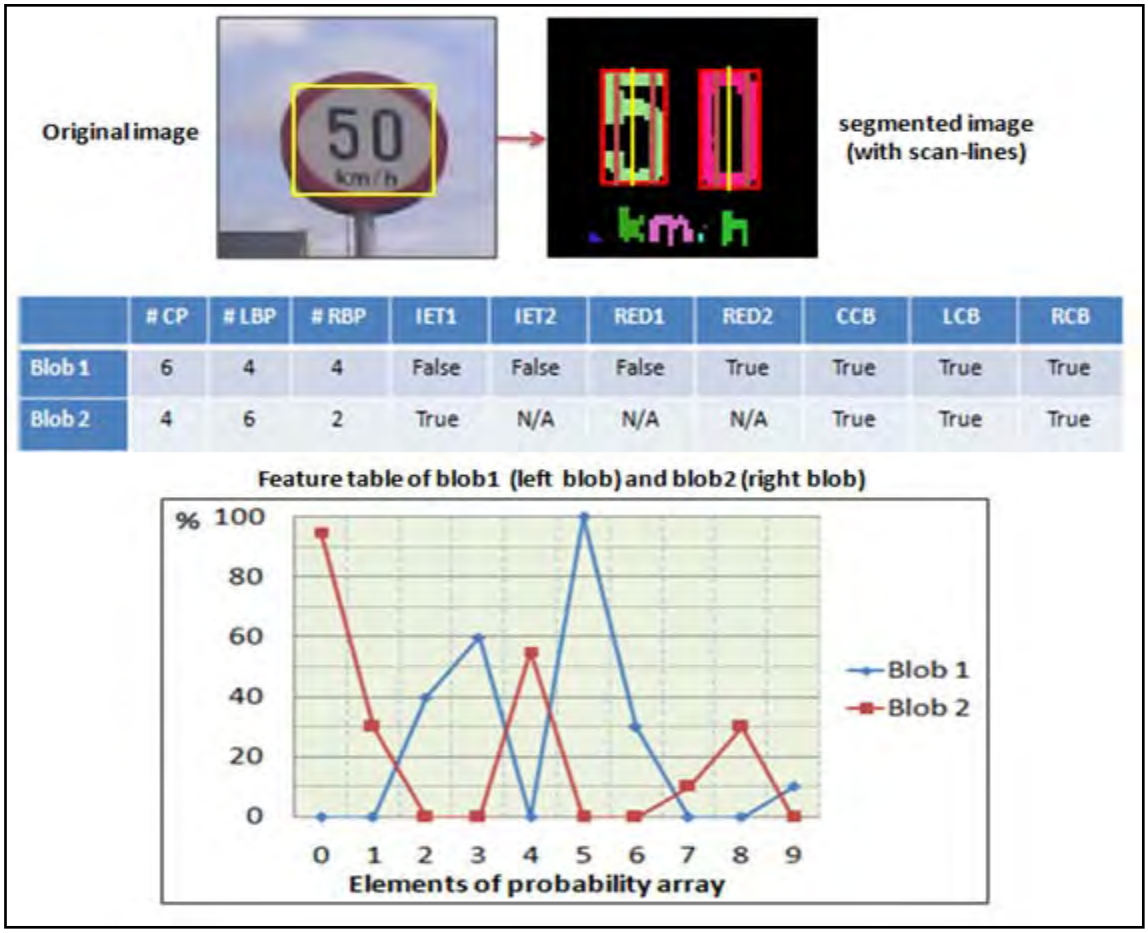
```
digit_recognition (segmentedImgVector)
    characterComponent   char_blob
    for   segImg in segmentedImgVector:
        for char_blob in segImg:
            //get column index for three vertival scan lines
            middleLine = (char_blob.left + char_blob.right)/2
            leftLine = (char_blob.left + middleLine)/2
            rightLine = (middleLine+ char_blob.right)/2
```

```
//go through segImg row by row
for pix in segImg row
    if pix is edge pixel on middleLine
        cp[i] = pix.row
        i++
    if pix is edge pixel on leftLine
        lbp[j] = pix.row
        j++
    if pix is edge pixel on rightLine
        rbp[k] = pix.row
        k++
//check number of cp
if 2<=cp.size() <=6
    char_blob.Reading = -1 //initialize recognition result
    //initialize element of probability array to 0%
    initialize(char_blob.probability [0~9])
    #cp = i
    #lbp = j
    #rbp = k
    if cp[i-1] close to bottom of the boundary
    ccb = true
    if lbp[j-1] close to bottom of the boundary
    lcb = true
    if rbp[k-1] close to bottom of the boundary
    rcb = true
    //
    if cp.size() is more than 4
        bool iet1 = inner_edge_check(cp[1,2], middleLine)
        if ! iet1
        bool red1 = right_edge_check(cp[1,2], middleLine)
    if cp.size() is 6
        bool iet2 = inner_edge_check(cp[3,4], middleLine)
        if !iet2
        bool red2 = right_edge_check(cp[3,4], middleLine)

    //start to match features with standard feature table
    if #cp is 2
        // increase digit probability if match,
        char_blob.probability [1] +=15
        //decrease digit probability of all not match
        char_blob.probability[0,2,3,4,5,6,7,8,9] -=15
        .
        .
```

```
        .       //match every feature with each field of the table
    findLargest (char_blob.probability[])
    if the largest is >= 80
        char_blob.Reading = largestAt()
        //assign char_blob back to the vector
        segmentedImgVector[segImg.at()]
                                .characters[char_blob.at()]
                                = char_blob
```

The new algorithm is quite straightforward and robust. After all features are captured and the probability array (10 elements, corresponds to 10 possible digit characters) is initialised, compare each feature with corresponding field in the standard feature table (see Table 2 in chapter 3), increase probability of every digit with matched field and decrease probability of every digit with not matched field. If the final largest probability is greater than 80, reading of the blob is the array index of the largest probability. If largest probability is less than 80, the blob is considered as a non-digit pattern. Figure 4-8 below shows a feature table of two blobs and the broken line graph of elements in their probability arrays. Two peaks of the graph indicate the reading of blob 1 is digit "5" and blob 2 is digit "0".




| | # CP | # LBP | # RBP | IET1 | IET2 | RED1 | RED2 | CCB | LCB | RCB |
|---|---|---|---|---|---|---|---|---|---|---|
| Blob 1 | 6 | 4 | 4 | False | False | False | True | True | True | True |
| Blob 2 | 4 | 6 | 2 | True | N/A | N/A | N/A | True | True | True |

*Figure 4-8 A feature table contains features of two blobs and the broken-line graph which demonstrates the distribution of probability of each possible reading after feature matching.*

## Result Checking

In order to eliminate illegal speed limit readings, recognition result needs to be checked before cueing driver. reading_check() function is designed to handle this task:

```
reading_check (segmentedImgVector)

    for segImg in segmentedImgVector with its characters is NOT empty:
        //discard segImg if the first digit is recognised as '0'
        if segImg.characters[0].possibleReading is '0'
            segmentedImgVector.remove(segImg)
        else
            //if only one blob on the ellipse
            if segImg.characters.size() = 1
                //discard segImg if the digit is '1' '2' '3' or '4'
                if segImg.characters[0].possibleReading is '1','2', '3' or'4'
                    segmentedImgVector.remove(segImg)
            //if there are 3 blob on the ellipse
            else if segImg.characters.size() = 3
                //discard segImg if the first digit is not '1'
                if segImg.characters[0].possibleReading is NOT '1'
                    segmentedImgVector.remove(segImg)
```

As mentioned in previous chapters, accumulated probability is introduced when processing video sequences. It means that a single successful recognition of one video frame will not trigger the speed limit cue. System will keep tracking that sign in subsequent several frames. If the probability is accumulated and reached the threshold, the driver will be notified and speed limit sign will be displayed. accum_probability() takes the accumulated probability vector (a global variable, which each cell holds a speed limit, its current accumulated probability and number of frames passed), the reading of the latest successful recognition and its probability as the parameter. It returns the accumulated probability if exceeding the threshold:

```
int accum_probability (accumProbabilityVector, latest_reading, latest_pro)
    accum_pro = 0
    if latest_reading is NOT in accumProbabilityVector
        new_accum.reading = latest_reading
                    .accumPro = latest_pro * 0.6
                    .framesPassed = 0
        accumProbabilityVector.push(new_accum)
    else
         accumProbabilityVector[].accumPro= 0.6*temp + 0.4*latest_pro
```

```
    for temp in accumProbabilityVector.
        //if accumPro exceeds the threshold
        if temp.accumPro >= 85
            accum_pro = temp.accumPro
            //remove from vector
            accuProbabilityVector.remove(temp)
        temp.framesPassed ++
        if temp.framesPassed >10
            //remove from vector once more than 10 frames passed
            accuProbabilityVector.remove(temp)

return  accum_pro
```

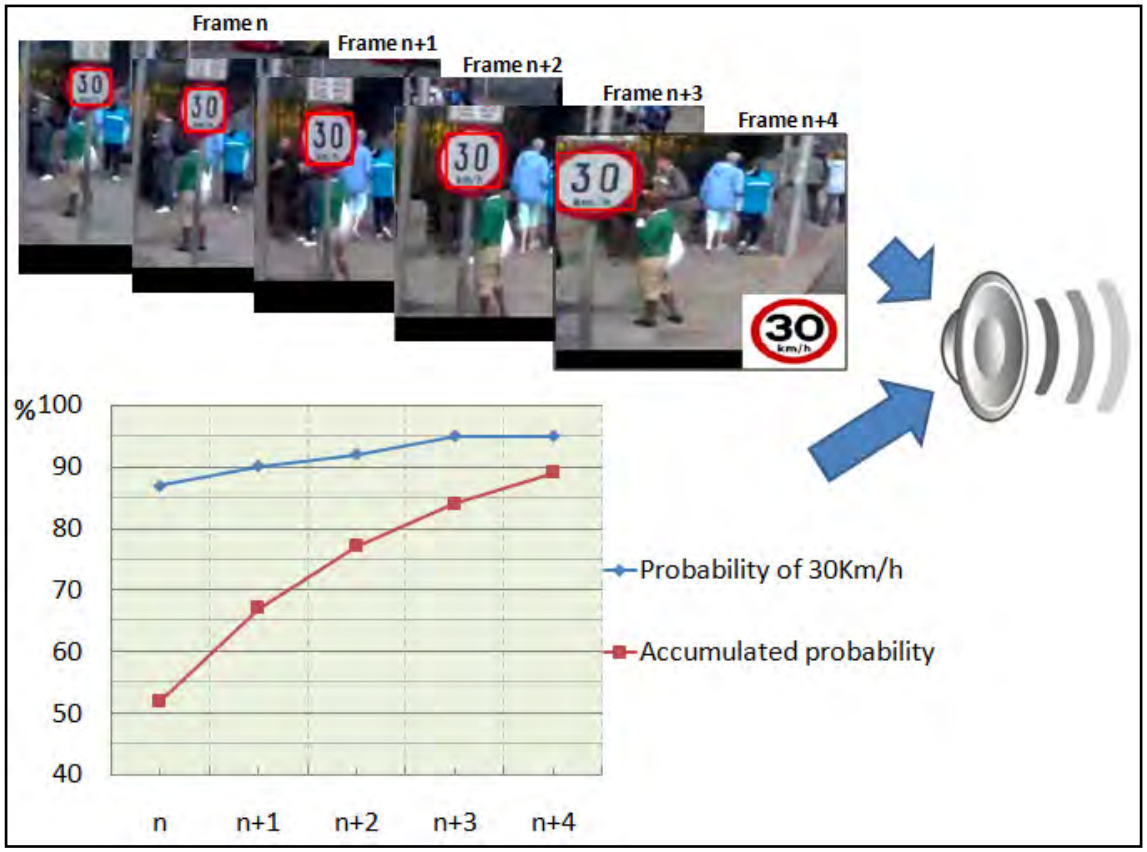



*Figure 4-9 A speed limit sign has been tracked since frame n. Driver is notified with cue tone at frame n+4 as the accumulated probability of "30 Km/h" exceed 85%.*

Introducing accumulated probability into the system can reduce the FPR (False Positive Rate) of recognition and also increases the robustness of the proposed system. Thus, after driver is notified the current speed limit sign, a recognition process cycle is finished.

# 5 Evaluation and Discussion

## 5.1 Sample database set up

In order to test and evaluate the performance of the product speed limit sign recognition system, a sample database is set up. Both static image and video segment are collected and stored as test samples. The equipment used to capture samples is a 5 Megapixel camera embedded in HTC Desire smart phone, which provides moderate image quality (see figure 5-1). Samples cover wide range of weather conditions including sunny, cloudy, rainy etc. different times of a day (at dawn, daytime, dusk etc.) and different environments (urban area, countryside and motorway). Several damaged speed limit sign samples are also included as static image samples. Database consists of 157 static images covering 9 different speed limit signs (5, 8, 10, 30, 50, 60, 80, 100 and 120 Km/h) and various negative samples (i.e. other regulatory traffic signs) with 800 * 600 image resolution, and 17 video segments (total length 38 minutes) covering multiple speed limit and other road signs with 640 * 480 resolution. Table 5-1 and 5-2 below shows the structure of static image and video database.


*Figure 5-1 The 5 megapixel camera of a HTC Desire smart phone.*

| | 5km/h | 8km/h | 10km/h | 30km/h | 50km/h | 60km/h | 80km/h | 100km/h | 120/km/h | Traffic sign |
|---|---|---|---|---|---|---|---|---|---|---|
| Sunny | 1 | 2 | 4 | 11 | 8 | 5 | 12 | 6 | 3 | 5 |
| Cloudy | 1 | 2 | 2 | 9 | 8 | 8 | 7 | 5 | 4 | 6 |
| Rainy | 0 | 0 | 2 | 4 | 2 | 5 | 4 | 3 | 1 | 7 |
| Dawn/Dusk | 0 | 1 | 2 | 2 | 3 | 0 | 1 | 1 | 0 | 2 |
| Damaged | 0 | 0 | 1 | 3 | 2 | 0 | 1 | 0 | 1 | 0 |
| Sum | 2 | 5 | 11 | 29 | 23 | 18 | 25 | 15 | 9 | 20 |
| **Total** | **Speed limit sign = 137** | | | | | | **Traffic sign = 20** | | | |

*Table 5-1 sample content of the static image database*

| | Urban area | | | Motorway | | | |
|---|---|---|---|---|---|---|---|
| | Sunny | Cloudy | Rainy | Sunny | Cloudy | Rainy | **Total** |
| Speed limit signs | 10 | 12 | 7 | 5 | 6 | 12 | **52** |
| Traffic signs | 7 | 14 | 9 | 1 | 3 | 6 | **40** |

*Table 5-2 totals of speed limit and traffic sign in video segments*

The evaluation is divided into two parts, which are static image test and video test respectively. In static test, every sample in static image database is processed by the system. Then, Real-time recognition ability of the system is evaluated in video test with video segments in sample database. All experiments were carried out on a laptop computer with Intel Core 2.00GHz CPU and 2 GB RAM.

## 5.2 Experiments and evaluation

### 5.2.1 Static image test

Static test concentrated on evaluating the performance of speed limit sign detection and recognition processes with sample images. Statistical results of detection and recognition stages are shown in Table 5-3:

| | | | **SLS Detection** | | **Recognition** | |
|---|---|---|---|---|---|---|
| | SLS | Traffic signs | True Positive | True Negative | Correct | Misclassified |
| Sunny | 52 | 5 | 50 | 5 | 48 | 2 |
| Cloudy | 46 | 6 | 42 | 6 | 40 | 2 |
| Rainy | 21 | 7 | 18 | 7 | 17 | 1 |
| Dawn/Dusk | 10 | 2 | 7 | 2 | 6 | 1 |
| Damaged | 8 | 0 | 7 | 0 | 6 | 1 |
| **Sum** | **137** | **20** | **124** | **20** | **117** | **7** |
| | | | **TPR = 90.51%** | **TNR= 100.00%** | **CR = 94.35%** | **MR = 5.65%** |

*Table 5-3 Results of speed limit sign detection and recognition*

TPR denotes the True Positive Rate, which shows the percentage of successful speed limit sign detection. TNR (True Negative Rate) shows the percentage of successful negative sample (i.e. other traffic sign) elimination. CR (Correct Rate) shows the accuracy of classification and MR (Misclassification Rate) shows the percentage that system misclassifies the input sign. The statistics indicate that the overall performance for both detection and recognition stage is very good. TPR of the detection stage is 90.51%, and none of negative is regarded as a speed limit sign. Meanwhile, 94.35% CR proves that the new scan-line based algorithm is working well in practice. The average processing speed of static image test is 0.209 second per image.

By inspecting the obtained results, the situations of confused speed limit sign detection can be attributed to poor lighting. Colour segmentation is not able to detect the inner ellipse of a speed limit sign correctly, since red colour becomes much darker under poor lighting conditions. Twilight also causes the same problem as it dyes everything in the scene red. Figure 5-2 below illustrates some of these situations:

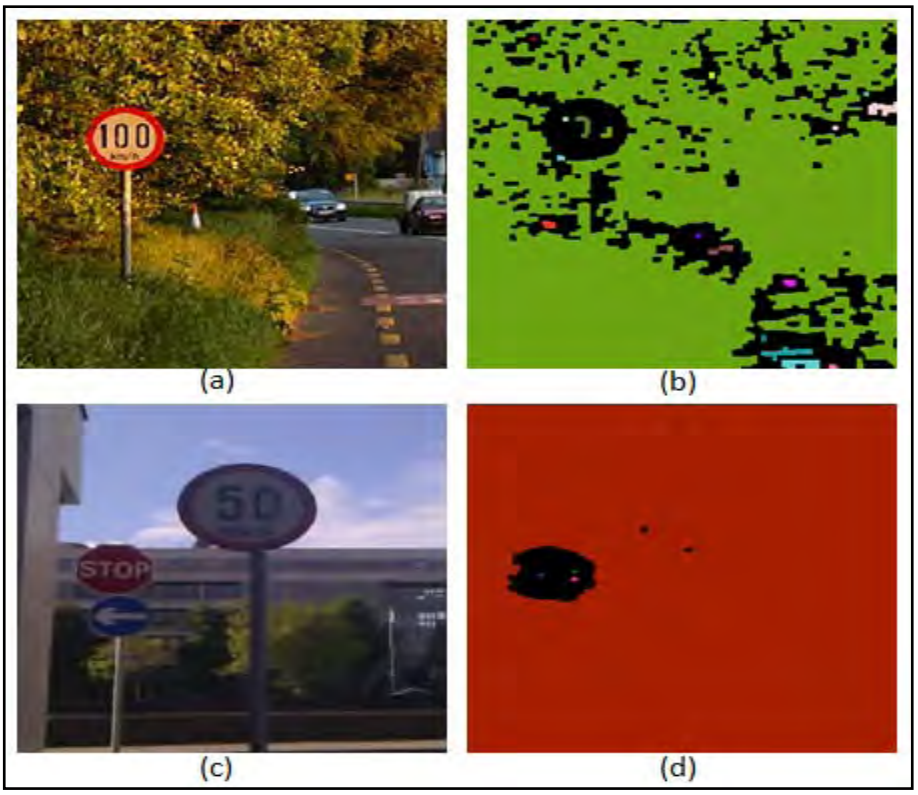



*Figure 5-2 An example of unsuccessful detection caused by twilight and poor lighting condition*

Figure 5-2(a) is a sample image captured at dawn. (b) is the labelled image after colour segmentation. As we can see, the inner ellipse of the sign is dyed red by the twilight and becomes a “red” region. Hence, the sign is discarded by the system mistakenly. Similarly, the speed limit sign in (c) was covered under the shadow of the building on the left. This causes that the outer ring of the sign is so dark to be red. Therefore, the system marks it as a non-red region and merges it with other regions. Thus, detection process is aborted due to no non-red ellipse detected.

The main situation which causes misclassification in static test is the serious slope of speed limit sign. Although the new digit recognition algorithm is resistant to small slope of character, a few serious sloped samples are still confusing the system (see figure 5-3).


*Figure 5-3 A serious sloped speed limit sign causes system misclassified it as a negative sample.*

We can see that the two character blobs of the sample in Figure 5-3 are extracted correctly. However, due to mismatched features, the left blob could not be classified as digit "1" and the sign is discarded, even though the right blob is classified as digit '0' correctly.

### 5.2.2 Video test

After achieving a good performance in static image test, video test is carried out to examine the system accuracy of real–time recognition. Table 5-4 below contains the statics of this experiment:

| | SLS | Traffic signs | NOA | NCA | FP | FN | SA | FPR | FNR |
|---|---|---|---|---|---|---|---|---|---|
| Urban area | 29 | 30 | 28 | 27 | 0 | 1 | 93.10% | 0.00% | 3.45% |
| Motorway | 23 | 10 | 22 | 21 | 0 | 1 | 91.30% | 0.00% | 4.35% |
| **TOTAL** | **52** | **40** | **50** | **48** | **0** | **2** | **92.20%** | **0.00%** | **3.85%** |

*Table 5-4 Statics of system performance in video test*

NOA represents the Number of Alarm triggered. NCA (Number of Correct Alarm) represents the number of correct speed limit sign recognition. FP (False Positive) denotes how many negative samples are classified as speed limit sign. FN (False Negative) denotes number of speed limit signs which were NOT detected by the system. SA is the System overall Accuracy. FPR and FNR are the "False Positive Rate" and "False Negative Rate" of the system respectively. Result indicates that high overall system accuracy 92.20% is achieved. It proves that the algorithms of the system are robust since none of negative sample misclassified as a speed limit sign. A low FNR also proves that a good reliability is attained. Meanwhile, it has been proved that the system is able to perform real-time recognition. The average frame frequency in the test is 13 fps (Frames per second), which means that it takes around 0.077 second to process one video frame. By investigating the statistics, we can see that higher system accuracy can be attained if the FNR reduced. The main reason causes False Negative result in video test is the poor weather condition, especially, the rainy weather (see figure 5-4).

*Figure 5-4 A False Negative result was caused by the rain drops on the windshield.*

In the example, the “60 Km/h” sign was blurred by the rain drops on the windshield which resulted in unclosed red outer ring. Thus, the speed limit sign detection was unsuccessful. The similar situation exists in the next several frames, and the accumulated probability of this 60 Km/h sign could not exceed 85% throughout.

However, this does not simply mean that the system cannot work properly on rainy day. In fact, the system works well on rainy days most of the time. In most cases, rain drops could only affect several serial frames and speed limit signs in the scene will be recognised once no longer disturbed by those rain drops (see figure 5-5).

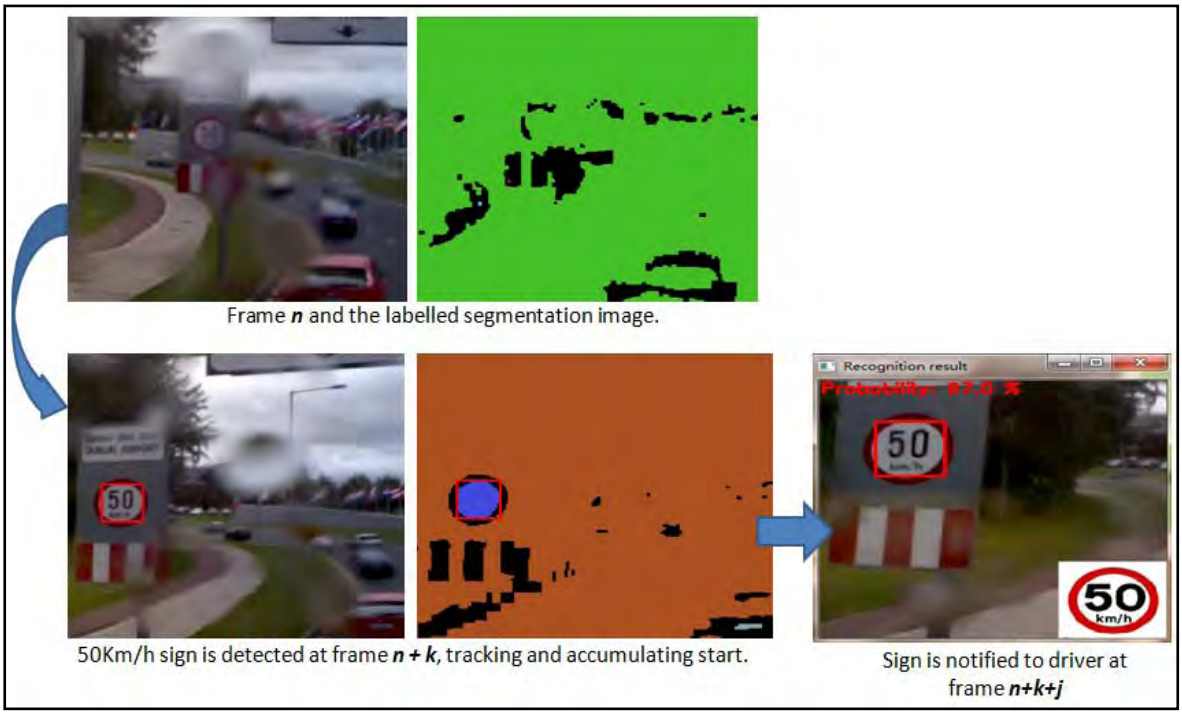



*Figure 5-5 An example demonstrates the process how a 50 Km/h sign is detected and notified to the driver under rainy weather.*

# 6 Conclusion

## 6.1 Conclusion

In this thesis, a robust approach for reliable real-time speed limit sign recognition has been designed and presented. A new improved scan-line based numerical digit recognition algorithm has been introduced. Experimental results indicate that the proposed system of this research achieved real-time speed limit sign recognition with high overall accuracy.

The proposed system, following the classical structure of recognition system, was divided into three main process stages, which are colour segmentation, sign detection and digit recognition. Colour segmentation discards all uninterested regions in YCrCb colour space to smooth speed limit sign detection. Possible speed limit signs in the scene are detected and extracted from regions of interest. Instead of applying Circular Hough Transformation to detect a circular object, LSVM was used to identify circular region of interest and speed up the process. In the last stage, candidate signs are classified via proposed scan-line based digit recognition algorithm, which provide reliable performance with less system complexity. Finally, diver is notified with evaluated speed limit readings via both visual and acoustic signals. The system attained a 92.20% recognition accuracy and an average processing speed 0.077 second per frame, which means the main research goal of this thesis has been achieved. Meanwhile, several possible improvements have been discovered during evaluation. Due to the time constraints, these improvements are considered as future works of this thesis.

## 6.2 Future works

First of all, in order to overcome speed limit sign misclassification caused by sloped signs (see figure 5-3 for an example), geometric transformation algorithms could be introduced. These algorithms are frequently used to introduce distortion into a scene. In this case, for any sloped sign, we can calculate the angle between the bottom of the digit character (or the bottom of the pattern “Km/h”) and the lower border of the image first, and rotate every pixel according to the rotation operators[2] (13) below:

$$x_2 = \cos\theta * (x_1 - x_0) - \sin\theta * (y_1 - y_0) + x_0$$
$$y_2 = \sin\theta * (x_1 - x_0) + \cos\theta * (y_1 - y_0) + y_0 \qquad (13)$$

Where $x_0, y_0$ are two coordinates of the centre of rotation in the original image, and $(x_1, y_1)$

[2] Geometric Operation – Rotate URL: http://homepages.inf.ed.ac.uk/rbf/HIPR2/rotate.htm (Last visit: 25/08/2011)

and $(x_2, y_2)$ are coordinates of a pixel in the original image and its corresponding pixel after rotation. $\theta$ is the included angle we calculated. By applying (13), the original sloped speed limit sign is rotated back to vertical, and will be classified by the system correctly.

Another possible improvement is introducing unclosed red region fixing scheme to speed limit sign detection stage. Sometimes, part of the outer red ring is covered by other objects (see figure 5-4 for an example) which causing unclosed red regions and misdetection. If we can fix those unclosed red regions by evaluating the shape and size of the gap (i.e. whether the gap is small enough; whether the gap is a curve), the detection accuracy will be increased.

Last but not least, due to shortage of the time, the portability of the proposed system has not been tested, although it has been proved that the ability of embedded camera of Android smart phone is good enough as the input device of speed limit sign recognition system. In the future, the proposed system will be implemented on Android platform. Thus, the maximum portability of a real-time speed limit sign recognition system is achieved.

# Appendices

*Contents of the CD attached:*

1. PDF version of the dissertation document.
2. PDF version of abstract of the thesis.
3. Source code (.cpp) and Microsoft Visual Studio Solution (.sln) of the proposed system.
4. Sample database for demonstration (./SLS recognition sys Ver 1_0/ samples)

*System Installation & Demonstration Instruction:*

5. Install OpenCV2.1. URL: http://sourceforge.net/projects/opencvlibrary/
6. Set up OpenCV2.1 in Microsoft Visual Studio 2008.
7. Copy “SLS recognition sys Ver 1_0” folder from the CD into the OpenCV2.1 installation directory.
8. Copy the sound file “sound.wav” in “SLS recognition sys Ver 1_0” folder to the root directory of E disk drive.
9. Click “road_sign.sln” in “SLS recognition sys Ver 1_0” folder to open Visual Studio 2008.
10. Press “F5” to run SLS recognition sys Ver 1.0.
11. In the application, press “0~9” on the keyboard to demonstrate recognition from static images. Press “Q” to start recognition from video sample. In video demonstration, press Space key to pause / resume the recognition. Press “Alt + Tab” to switch between result images of different processing stages.

# Bibliography

[1] W. C. Xu Han Wei, "A New Algorithm for Numeral Recognition," *Surveying and Mapping of Geology and Mineral Resources*, vol. 2, p. 31, 2002.

[2] K. A. Ishak, *et al.*, "A Speed limit Sign Recognition System Using Artificial Neural Network," in *Research and Development, 2006. SCOReD 2006. 4th Student Conference on*, 2006, pp. 127-131.

[3] C. Hsin-Han, *et al.*, "Road speed sign recognition using edge-voting principle and learning vector quantization network," in *Computer Symposium (ICS), 2010 International*, 2010, pp. 246-251.

[4] L. Wei, *et al.*, "Real-Time Speed Limit Sign Detection and Recognition from Image Sequences," in *Artificial Intelligence and Computational Intelligence (AICI), 2010 International Conference on*, 2010, pp. 262-267.

[5] S. Maldonado-Bascon, *et al.*, "Road-Sign Detection and Recognition Based on Support Vector Machines," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 8, pp. 264, 2007.

[6] H. Yea-Shuan and L. Yun-Shin, "Detection and recognition of speed limit signs," in *Computer Symposium (ICS), 2010 International*, 2010, pp. 107-112.

[7] J. Srinonchat, "Efficient detection of speed limit signs within obscure environment," in *Wireless, Mobile and Multimedia Networks (ICWMNN 2010), IET 3rd International Conference on*, 2010, pp. 311-314.

[8] A. Broggi, *et al.*, "Real Time Road Signs Recognition," in *Intelligent Vehicles Symposium, 2007 IEEE*, 2007, pp. 981-986.

[9] N. Barnes and A. Zelinsky, "Real-time radial symmetry for speed sign detection," in *Intelligent Vehicles Symposium, 2004 IEEE*, 2004, pp. 566-571.

[10] W. Jianping, *et al.*, "Real-time Robust Algorithm for Circle Object Detection," in *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for*, 2008, pp. 1722-1727.

[11] S. Muller-Schneiders, *et al.*, "Performance evaluation of a real time traffic sign recognition system," in *Intelligent Vehicles Symposium, 2008 IEEE*, 2008, pp. 79-84.

[12] J. Miura, *et al.*, "An active vision system for real-time traffic sign recognition," in *Intelligent Transportation Systems, 2000. Proceedings. 2000 IEEE*, 2000, pp. 52-57.

[13] L. Han, *et al.*, "Real-time recognition of road traffic sign in motion image based on genetic algorithm," in *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, 2002, pp. 83-86 vol.1.

[14] M. L. Eichner and T. P. Breckon, "Integrated speed limit detection and recognition from real-time video," in *Intelligent Vehicles Symposium, 2008 IEEE*, 2008, pp. 626-631.

[15] W. Yongping, *et al.*, "A Method of Fast and Robust for Traffic Sign Recognition," in *Image and Graphics, 2009. ICIG '09. Fifth International Conference on*, 2009, pp. 891-895.

[16] J. Torresen, *et al.*, "Efficient recognition of speed limit signs," in *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, 2004, pp. 652-656.

[17] A. de la Escalera, *et al.*, "Visual sign information extraction and identification by deformable models for intelligent vehicles," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 5, pp. 57, 2004.

[18] S. Xu, "Robust traffic sign shape recognition using geometric matching," *Intelligent Transport Systems, IET*, vol. 3, pp. 10, 2009.

[19] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381, 1981.

[20] N. C. Stache and H. Zimmer, "Robust Circle Fitting in Industrial Vision for Process Control of Laser Welding," in *Proceedings of the 11th International Student Conference on Electrical Engineering POSTER 2007.*, 2007, p. pp.1.

[21] D. P. Huttenlocher, *et al.*, "Comparing images using the Hausdorff distance," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 15, pp. 850, 1993.

[22] G. K. Siogkas and E. S. Dermatas, "Detection, Tracking and Classification of Road Signs in Adverse Conditions," in *Electrotechnical Conference, 2006. MELECON 2006. IEEE Mediterranean*, 2006, pp. 537-540.

[23] D. J. Kerbyson and T. J. Atherton, "Circle detection using Hough transform filters," in *Image Processing and its Applications, 1995., Fifth International Conference on*, 1995, pp. 370-374.

[24] X. Baro and J. Vitria, "Fast traffic sign detection on greyscale images," *Recent Advances in Artificial Intelligence Research and Development*, pp. 69, 2004.

[25] V. N. Vapnik, "An overview of statistical learning theory," *Neural Networks, IEEE Transactions on*, vol. 10, pp. 988, 1999.

[26] P. Gil-Jimenez, *et al.*, "Traffic sign shape classification based on Support Vector Machines and the FFT of the signature of blobs," in *Intelligent Vehicles Symposium, 2007 IEEE*, 2007, pp. 375-380.

[27] D. Castells-Rufas and J. Carrabina, "Camera-based Digit Recognition System," in *Electronics, Circuits and Systems, 2006. ICECS '06. 13th IEEE International Conference on*, 2006, pp. 756-759.

[28] L. Yibo and Q. Hongjuan, "Automatic Recognition System for Numeric Characters on Ammeter Dial Plate," in *Young Computer Scientists, 2008. ICYCS 2008. The 9th International Conference for*, 2008, pp. 913-918.

[29] C. M. Bishop, *"Neural Networks for Pattern Recognition"*: Oxford University Press, 1996.

[30] H. Samet and M. Tamminen, "Efficient component labeling of images of arbitrary dimension represented by linear bintrees," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 10, pp. 579, 1988.

[31] T. W. Ridler and S. Calvard, "Picture thresholding using an iterative selection method," *IEEE Trans. System*, vol. SMC-8, pp. 630, 1978.

[32] R. Haralick and L. Shapiro, "Computer and Robot Vision," *Addison-Wesley Publishing Company*, vol. 1, p. 174, 1992.