A Colour Segmentation Method for Detection of New Zealand Speed Signs

Abhishek Bedi

A thesis submitted to Auckland University of Technology in fulfilment of the requirements for the degree of Master of Engineering

AUT

UNIVERSITY

TE WĀNANGA ARONUI O TAMAKI MAKAU RAU

August 2011

School of Engineering

Primary Supervisor: Dr John Collins

Acknowledgements

This dissertation is a part of the Masters of Engineering at Auckland University of Technology New Zealand.

This piece of work is a result of hard work, patience, sacrifice and unconditional support of many people. I wish to thank everyone who has supported and helped me in completing this research.

Firstly, I wish express my gratitude to Geosmart NZ Limited for inspiring me to carry out this work. I am thankful to my supervisor Dr. John Collins for keeping patience, being a motivating force and taking care of me during the process. I would like to thank them for the guidance they have given me throughout the period.

I express my honest thanks to the AUT library for their overall support for the entire period of the study at AUT.

At the end, I am grateful to my parents and family members, especially my wife for her extreme sacrifices and providing moral support.

Statement of Originality

‘I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for qualification of any other degree or diploma of a university or other institution of higher learning, except where due acknowledgement is made in the acknowledgements.’

Abhishek Bedi

Abstract

New Zealand Speed signs provide safe travelling speed limit or guidance information to drivers on roads. The speed signposts are provided alongside the national roads and highways as a part of guidance system for drivers and are entrenched by the New Zealand Transport Agency (NZTA). The speed limit in New Zealand is decided by the speed limits policy of NZTA.

The objective of speed limits policy is to balance the interests of mobility and safety by ensuring speed limits are safe, appropriate and credible for the level of roadside development and the category of road for which they are set. In New Zealand, speed limits may be temporary, changing kilometre by kilometre or hour to hour.

The research undertaken is to develop a commercially effective NZ Speed sign recognition system to be used at Geosmart NZ Ltd. Geosmart (NZ) Ltd is New Zealand's geospatial solutions provider. The current system being used to determine the speed signs is a manual system, where a human input is required to observe terabytes of data. To speed up the process and get accurate information, the company requires an automated system to detect changes in the speed limits in short interval of time at different areas on the country’s roads.

This research will aim at finding an efficient solution for recognition of NZ based road signs in software, especially when the speed signs are located in dark or gloomy images.

This research proposes a new method to treat images with low level lighting conditions during the process of colour segmentation, a method used in digital image processing.

Table of Contents

Table of Figures

List of Tables

List of Code Snippets

Companion CD

The Companion CD provided comes with all the white papers, websites and other electronic sources referred while compiling this thesis.

The CD also comes with the results of experiments done while doing the research, including a working software setup file.

Definitions

Chromaticity: It is an objective specification of the quality of a colour regardless of its luminance that is, as determined by its hue and colourfulness.

Colorimetry: Colorimetry is the science of colour measurement.

Colour Histogram: Colour histogram is a representation of the distribution of colour in an image

Colour Meter: Also known as a tristimulus colorimeter is used in digital imaging, to profile and calibrate output devices.

Colour Segmentation: It refers to the process of partitioning a digital image into segments based on colour.

Colour Threshold: It is a method of image segmentation. During this process, individual pixels in an image are selected if their value is greater than some threshold value.

Contour Line: A contour line of a function of two variables is a curve along which the function has a constant value.

Illumination: Illumination is a deliberate application of light to achieve some aesthetic or practical effect.

Image Moment: An image moment is a certain particular weighted average (moment) of the image pixels' intensities

Luma: A luma also called as luminance, represents the brightness in an image.

Normalisation: It is a process in image processing that changes the range of pixel intensity values.

Reflection: Reflection is the change in direction of a wave of light at an interface between two mediums.

Abbreviations

CAD: Chromatic/ Achromatic Decomposition

CCD: Charged Couple Device

CIE: Commission Internationale de I’Eclairage

CIECAM: Commission Internationale de I’Eclairage Colour Appearance Model

CRT: Cathode Ray Tube

CST: Colour Space Thresholding

CV: Computer Vision

DFT: Discrete Fourier Transform

DIP: Digital Image Processing

FLD: Fisher’s Linear Discriminant

GUI: Graphical User Interface

HST: Hue and Saturation Thresholding

LCD: Liquid Crystal Display

LED: Light Emitting Diode

LUT: Look Up Table

NTSC- National Television System Committee

NZTA: New Zealand Transport Agency

PCA: Principle Component Analysis

RBFNN: Radial Basis Function Neural Network

sRGB: Standard RGB Colour Space

SVM: Support Vector Machine

TSR: Traffic Sign Recognition

TV: Tele Vision

UCS: Uniform Chromaticity Scale

1 Introduction

Road signs are installed to regulate traffic, provide traffic designations and other useful information. They are installed at specific locations and appear with colours that contrast against road environment[1]. They are designed to be easily recognized by human drivers mainly because their colour and shapes differ from those present in natural environments[2]. Recognition of traffic sign has been a challenging problem for many years and is an important task for the intelligent vehicles and related technologies. Although the first work in this area started in 1960s, but significant advances were made later in 1980s and 1990s [3]. To recognise various road signs, there have been a few systems developed so far ,for example , Traffic Sign Recognition (TSR) [2], Automatic target recognition system [4], TSR using neural networks [5], Driver Assistance System (DAS) [6] and TSR using discriminative local features [7] , to name a few.

Digital Image Processing (DIP) is the use of computer programs to analyse and process digitally captured images (using digital or video camera). As a subfield of digital signal processing, digital image processing has many advantages over analogue image processing; it allows a much wider range of algorithms to be applied to the input data. It can also avoid problems such as the build-up of noise and signal distortion during processing.

To develop a sign recognition system, the field of digital image processing is used in conjunction with template matching techniques[2] or some researchers prefer to use artificial intelligence techniques like neural networks [8] and genetic algorithm [1] for processing the images. The images are analysed and processed using various DIP techniques like shape detection, edge detection, colour matching.

The procedure of road sign recognition is usually a two-stage sequential approach, firstly it aims at locating regions of interest and verifying the hypotheses on the sign’s presence and subsequently determining the type of detected sign [3].

Geosmart (NZ) Ltd is New Zealand's geospatial solutions provider. It provides the New Zealand market with hardcopy aerial photography, digital imagery for GIS land bases, full photogrammetric mapping, 3D modelling and remote sensing. To develop their dataset, they drive through every public road in New Zealand and collect a wide database of attributes which are combined with their data collected through aerial photography and other means[9]. The stored data is searched manually for any special situations like variation of speed zones, which is then used to update their database with accurate information. Currently this process of analysing terabytes of image data is cumbersome and time consuming. This process can lead to incorrect or out of date information which can be harmful for people using this data in systems like GPS or in car navigation systems.

Due to this problem this research aims at automating the task of speed sign recognition using cameras, so as to provide an automated system to update the current information. The research taken is to develop a commercially effective NZ Speed sign recognition system to be used at Geosmart NZ Ltd. As mentioned, the current system being used to determine the speed signs is a manual system, where a human input is required to observe terabytes of data. To speed up the process and get accurate information, the company required an automated system to detect speed signs in the images held in their database.

This research aims at designing efficient software for automatic recognition of NZ based road signs. Open CV, an open source image processing library by Intel®, was used to develop this software along with its C# wrapper, Emgu CV. It was a requirement of Geosmart that open source software be used for this project.

In computer vision, colours are represented using mathematical models, known as colour models. Today there are different types of colour models that are being used in the computer vision field. Usage of a particular colour model depends on the application it is being used for. Here the author has analysed and used two of the most common type of colour models, the RGB colour model and the HSV colour model for detection of traffic signs.

The basic procedure for detecting of objects in computer vision involves segmenting of images based on colour, followed by detection of edges and then finally matching of shapes.

One of the major challenges while using computer vision during the procedure of colour segmentation is detecting of objects under different illumination conditions. It specially becomes more challenging when the image has been shot out in the open, as a change in lighting condition can turn bright objects into dark coloured objects, thus making it difficult to detect an object based on its colour. The most difficult images are the ones shot during cloudy weather or sudden change of illumination.

Different available methods for colour segmentation were reviewed for the purpose of traffic sign detection, mentioned in the literature review. These methods, though worked well with most of the images, behaved inconsistently while recognising signs in gloomy images and bright images. A similar kind of problem arises with poorly shot images, using low quality cameras or some other kind of added noise. Hence there was a need to develop a better method for speed sign detection.

This research is aimed at finding a robust solution to counter such a problem, especially when the speed signs are located in a dark or gloomy images, or when the illumination changes drastically, hence making it difficult to recognise a sign.

This research proposes a new method to treat images with low level lighting conditions during the process of colour segmentation.

The testing of the new software was performed individually for colour segmentation technique proposed and then it was tested together with all other computer vision methods involved in developing this software, like edge detection and shape matching techniques. Tests were performed on a data set of images provided by Geosmart, also it was tested on some random speed sign images downloaded from the internet.

A literature review including an explanation of colour, its properties and types of colour models is given in chapter 2. Chapter 3 reviews the currently available colour segmentation techniques for detection of speed signs. Chapter 4 discusses the image processing techniques used in the final and the sample program for sign detection provided with Emgu CV. Chapter 5 explains the newly proposed method and chapter 6 and chapter 7 analyse the results and discuss the enhancements. The final chapter is chapter 8 where the conclusions of this work and future work have been described.

2 Background: Colour Processing Techniques used in Program

This chapter will describe various image processing terminologies and techniques used in this research to develop the computer vision program (such as colour, colour model, colour spaces).

2.1 Colour

Digital image processing is over 45 years old and colour has been a part of that for at least last 35 years [10]. With the advent of low cost colour CCD cameras and fast computer systems, colour image processing has become increasing popular [11], hence making colour one of the most important features of modern day image processing .A colour is a perceived phenomenon, not a physical property of light [10]. Colour is the perception of relative different- wavelength light intensities by the human eye [12].

In the early 1670s, Isaac Newton demonstrated that white light is a combination of all other colours of light. Colour perception relates to the variable absorption of different components of white light. Objects absorb colours rather than emitting them and they reflect different colours in different amounts, allowing objects to be perceived as coloured, based upon what colours they do not absorb[12]. As per Sangwine [10], "The perceived colour of a surface is due to the distribution of power across the wavelengths visible to the human eye".

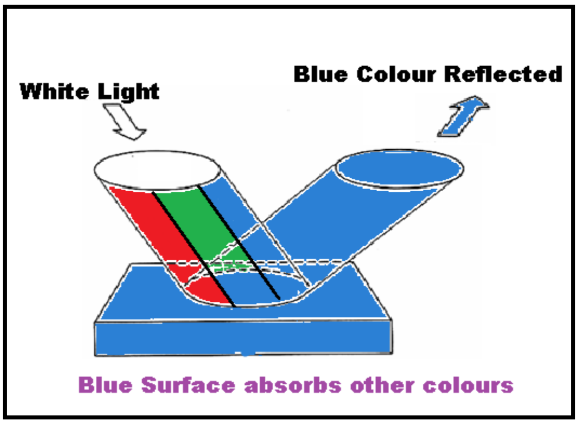


Figure 1 Surface absorbing all the components of light except blue.

As shown above, we can define colour as a variation in visible light viewed under different lighting conditions, caused due to the absorption of various components of a white light, by a surface. Colour is usually defined as a value or an attribute of visual perception that can consist of any combination of chromatic (yellow, orange, red, blue) and achromatic (white, gray, black) content. Hue, Chroma and Value are three attributes that are frequently used to define colour [13]. Brightness, lightness, saturation are some of the other properties that define a colour.

To understand a colour, we will refer to CIECAM97s, the colour appearance model recommended by the CIE (Commission Internationale de I'Eclairage). The CIECAM97s colour model defines various elements of an illumination environment and a colour stimulus in following way[14] :

I. Colour stimulus is the colour being observed. It is defined to cover about 2 degrees of visual angle. It is also defined as the colour element for which a measure of colour appearance is described [15].

II. Background is the area immediately surrounding the stimulus.

III. Surround field is the field that extends to the limit of vision from the background.

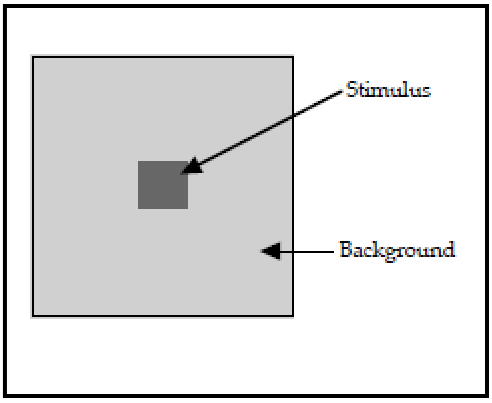


Figure 2 Showing the Stimulus and a Background[16]

CIECAM97 defines a surround field as average, dark, dim, etc.

In colorimetry, any colour can be represented by three numbers, which correspond to the amount of each of three primary colours used to generate the colour. These three numbers are called the tristimulus values for the colour [17].

2.1.1 Colour properties

2.1.1.1 Hue

Hue is a gradation of a colour. It is one of the main properties of a colour. It is the degree to which a stimulus can be described as similar to or different from stimuli that are described as red, green, blue and yellow [16]. The term hue, describes colour as the combination of different wavelengths used to produce colour [13].

Figure 3 Hue

2.1.1.2 Value/ Lightness

Value is a property of colour that usually defines its perceived lightness or darkness. It is different from the measured brightness (or luminance) of colour [13] and it reflects the subjective brightness perception of a colour to human eye. It is also, sometimes referred to as lightness and corresponds to the amount of energy present in all wavelengths of a colour. As described by Fairchild [16] Lightness is also "The brightness of a stimulus relative to the brightness of a stimulus that appears white under similar viewing situations".

Figure 4 Lightness / Value

2.1.1.3 Chroma

Chroma is colourfulness of a stimulus relative to the brightness of a stimulus that appears white under similar viewing conditions [16]. Chroma describes the saturation or purity of a colour. This indicates the range of wavelengths in which most energy of a colour is present [13].

Figure 5 Chroma

2.1.1.4 Saturation

Saturation is the colourfulness (perceived quantity of hue content) that is relative to its own brightness [15].

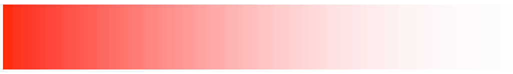


Figure 6 *Saturation of a colour*

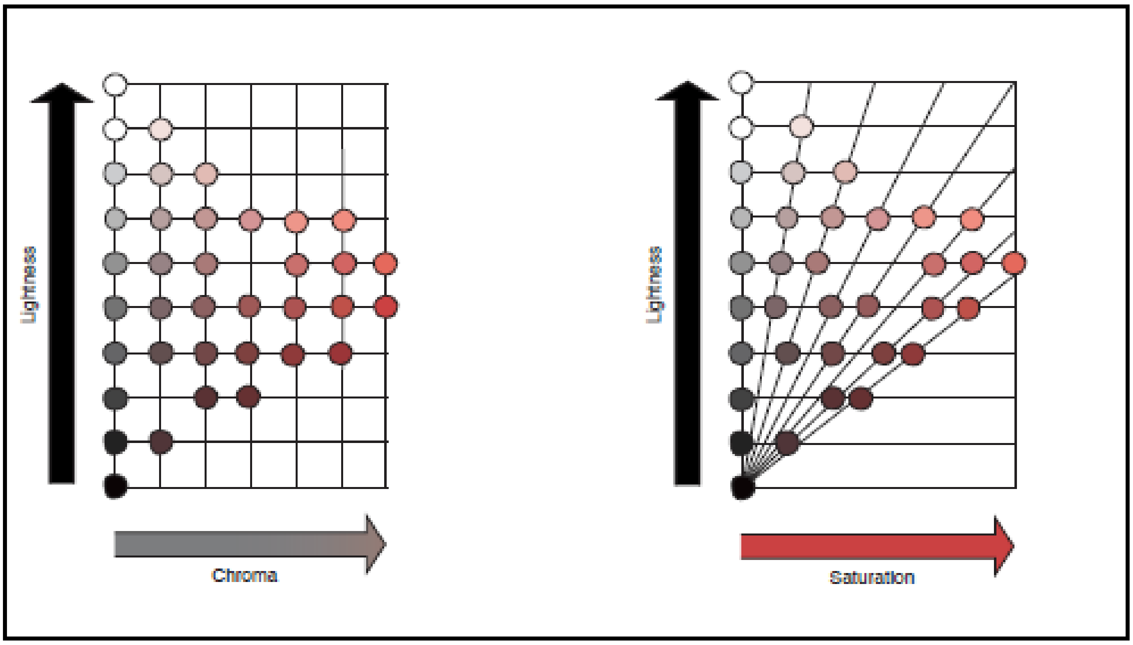


Figure 7 *Saturation and Chroma [16]*

2.1.1.5 Brightness

Brightness is a visually perceived quantity of light from a source that appears to be reflecting light. It is a subjective property of an object being observed.

Figure 8 *Brightness*

After looking at the properties associated with colour, a colour can also be defined as a combination of various attributes of visual stimulus, that are independent of spatial and temporal variations [15].

2.1.2 Types of Colour

Colours can be represented in two ways,

- The natural perception, also called Spectral Colours.
- The electronic representation of colours generated using artificial light, used in modern day technologies like CRT, TV, printing etc.

2.1.3 Spectral Colours

The colours that can be defined by a single wavelength or frequency of light are called Spectral Colours. They are also called monochromatic and are also referred to as pure colours. These colours are observed in nature and hence are also sometimes known as natural. They are separated into six or seven main regions as in a rainbow: red, orange, yellow, green, blue and violet.

Figure 9 A representation of spectral colours [12].

2.1.3.1.1 Human Eye

Our eye's colour vision is made up of light sensitive cells called *cones* that are located in the retina [17]. The retina is where an image is formed of the object being viewed. Hence cones act as sensors for the human imaging system that sense objects in different environments and lighting conditions. Cones are sensitive to three types of colours: red, blue and green, hence we perceive every colour as a combination of these three [18].

Figure 10 below shows the cone sensitivities of an eye. Here the response of each cone has been encoded as a wavelength called the spectral response curve for the cone. The cone response curves overlap significantly in the medium (green) and long (red) regions.

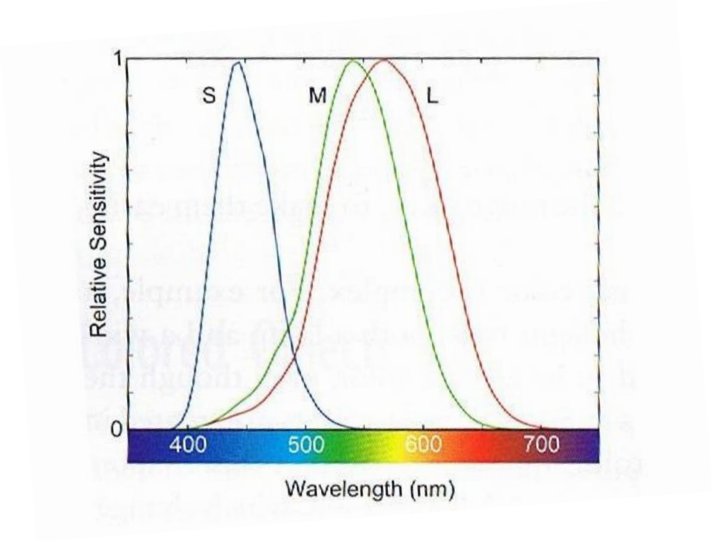


Figure 10 Eye's Cone Sensitivity[17]

2.1.3.2 Electronic Representation of Colour

Electronic colour representation was established by engineers developing colour television in 1950s [19]. The three primary colours of red, blue and green, were chosen to match the structure of the human eye. In addition to the natural occurrence of absorption-reflection, a transmitted light can be a combination of primary colours to make other colours. Different combinations of transmitted primary colours of various intensities yield different perceived colours. This combination of light is characterized as being additive and the natural absorption-reflection is subtractive.

2.1.3.2.1 Additive Colours

In electronic media, several colours are combined to produce a result that is perceived as a different colour that is lighter than its components. The colours generated due to this process are called additive colours. The colours that are used to generate additive colours are known as primary colours. The choice of primary colour varies, based on different colour models. For example, in the screen of a television or CRT phosphoric dots are clustered in groups, with each group containing red, blue, and green as the primary colours. Different amounts of red, blue and green are combined to generate different colours.

Shown below is an illustration of additive colours being generated:

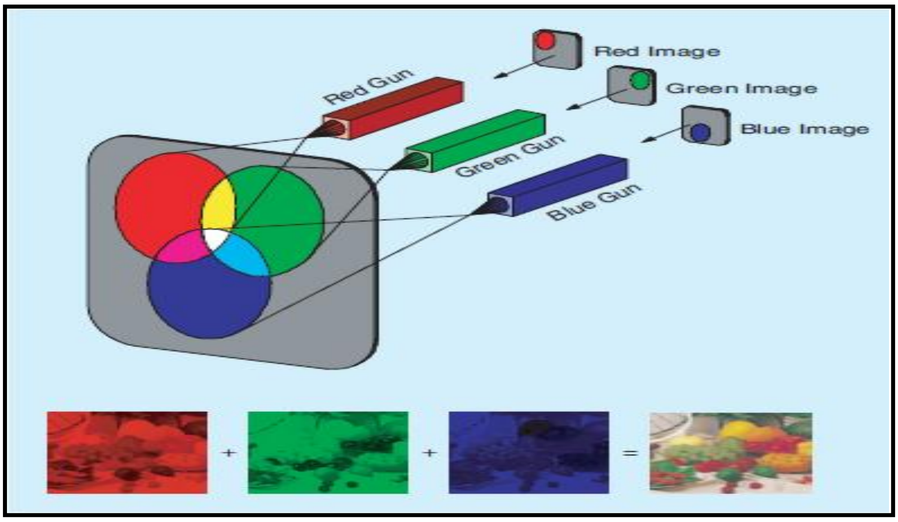


Figure 11 Additive Colour System[20]

In this example of additive colour formation, red (R), green (G) and blue (B) are the three stimuli.

2.1.3.2.2 Subtractive Colour

As one adds colours to transmitted light, the image gets brighter, but as more colours of paint are added on to a reflecting surface, the image gets darker. This is because painted surface absorbs light of particular frequencies, hence subtracting colours from the reflection of incident white light [18]. Thus, subtractive colours are obtained by the fundamental property of removing a selected portion from a source spectrum [20].

Each of the additive colours that are reflected from a monitor has an opposite or complementary subtractive colour, as shown below. The complemented colours for a RGB model are as follows:

- Red: Cyan
- Blue: Yellow
- Green: Magenta

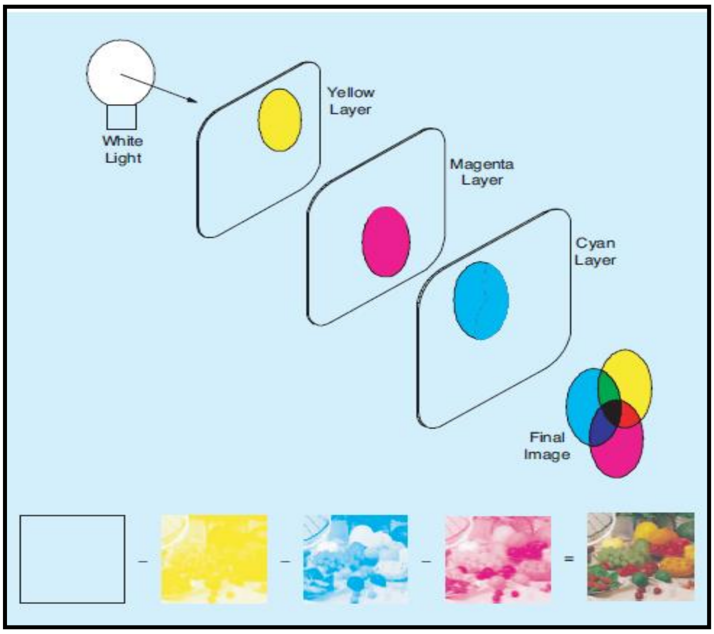


Figure 12 Subtractive Colour[20]

A digital image represented by either of the colour models consists of three components per pixel: one each for red or cyan, blue or yellow, green or magenta intensities.

To be able to represent these colours in digital world, various mathematical models have been created. The next section will describe the different types of models being used in traffic sign recognition.

2.2 Colour Model and Colour Space

In computer vision colours are represented using mathematical models, describing colours as ordered sets of number, typically as three or four values or colour components. These mathematical representations of colours are known as colour models. A colour model is also known as a colour appearance model, and tries to model how the human visual system perceives the colour of an object under different lighting conditions and with different backgrounds[14].

In technical terms, a colour model converts measured values such as the tristimulus values of the sample and its surrounding colour, correlating it to the perceptual attributes of a colour[17]. It is a mathematical model that is used to predict colour perceptions under different viewing conditions [14] like outdoors, indoors, on a CRT or a printed document. There are different types of colour model [21, 22] such as RGB Colour Model [23, 24], which is a more generic model, and also the specialised models like the Munsell Colour space [25, 26] and the CIECAM02 [14, 16, 27] colour model.

A colour model is often helpful in adjusting the colours of an image to look similar, when viewed in different lighting conditions. This ability of a colour model makes it useful for developing a device independent method for storing images[14], thus making it one of the prominent image processing elements while identifying digitally stored images.

Colourful images provide more information than grey scale images and can improve performance of existing systems; therefore colour models can be used to convert low level colour information into useful primitives for higher level processing [11, 28].

Colour systems can be of two kinds. Some colour systems are based on technical characteristics such as wavelength, and are formulated to help humans select colours, e.g. the Munsell system. Other colour systems are formulated to ease data processing in machines, for example The RGB model [13, 29]. The RGB model is a colour model without any associated mapping function to a technically defined colour system, and is considered as an arbitrary colour system.

When a colour model is associated with a precise description of how the components are to be interpreted, the resulting set of colours is called a colour space[30]. The technical association of a colour model and a reference colour space results in a definite footprint within the reference colour space and is known as a gamut. In digital colour displays, varying the brightness of the RGB primary colours varies the colour produced, the complete set of colours that can be produced on a display is called the colour gamut for that display[17]. The gamut in combination with the colour model, defines a new colour space e.g. Adobe RGB and sRGB are two different colour spaces based on the generic RGB model.

In colour science, there are colour spaces in which the perceptual difference between two colours is linearly related to the Euclidean, or linear, distance between them. They are known as uniform colour spaces or uniform chromaticity scale (UCS) colour spaces [13, 31] e.g. the L*a*b* and L*u*v* colour spaces. A UCS model is a mathematical model that matches the sensitivity of human eye with computer processing. The colour spaces in which the colour difference of human perception cannot be directly expressed by a Euclidean distance are known as non-UCS colour spaces, such as the RGB and XYZ colour spaces [31]. Colours selected from uniform colour spaces are optimised for visual discrimination by computer processing.

Currently there are a number of different types of colour spaces that are being used in the computer vision field. There have been around twenty specific defined colour systems as identified by Niblack [32], in the literature on colour and colorimetry. Usage of a particular colour space depends on the application it is being used for. Two of the most commonly used colour models in traffic sign detection are the RGB colour model and the CIECAM colour model. Their derivative colour spaces are discussed in the next sections.

2.2.1 RGB Colour Model

The RGB colour model is one of the main colour formats and simplest systems used to model colours, and is based on additive colours: red, green, and blue [23]. In this model, all the three colours are added together, and their light spectra add in various ways to reproduce a broad array of colours[33, 34].

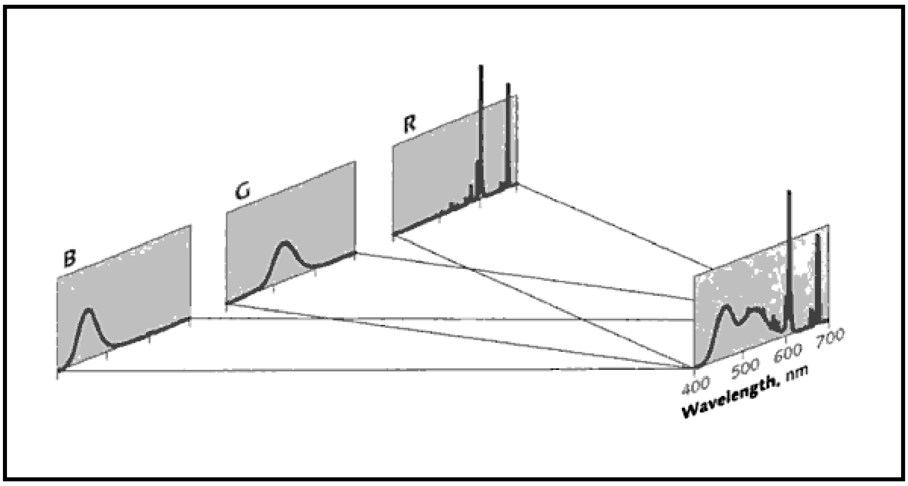


Figure 13 Additive reproduction in video signals[34]

The name of the model comes from the initials of the three additive colours that also act as primary colours for this model, namely, red (R), green (G), and blue (B).

The main purpose of the development of RGB colour model was for the representation and display of images in electronic systems, such as televisions, though it has also been used in conventional photography. The RGB model is mostly used in hardware-oriented applications such as colour monitors [35].Before the electronic age, the RGB colour model already had a solid theory behind it, based in human perception of colours. The choice of primary colours is related to the physiology of the human eye. Good primaries are stimuli that maximize the difference between the responses of the cone cells of the human retina to light of different wavelengths, and that thereby make a large colour triangle [36].

Devices that typically use RGB as input or output are colour TV sets (CRT, LCD, and plasma), personal computers and mobile phone displays and multicolour LED displays.

2.2.1.1 Representation of the RGB Model

A colour in the RGB model is described by indicating the amount of red, green and blue included in it. An arbitrary colour can be obtained from this model, by a weighted sum of the three primaries:

C= r R+g G+b B *Where r, g and b are known as the tristimulus values of R,G and B components, and C is the stimulus [23, 37].*

The RGB colour model can also be treated as a three-dimensional unit cube in which the three primaries form the coordinate axes. The RGB values are positive and lie in the range [0,C$_{max}$]; for most digital images, C$_{max}$ = 255. RGB values are often normalized to the interval [0,1] to form a colour space that can be represented by a unit cube [38].

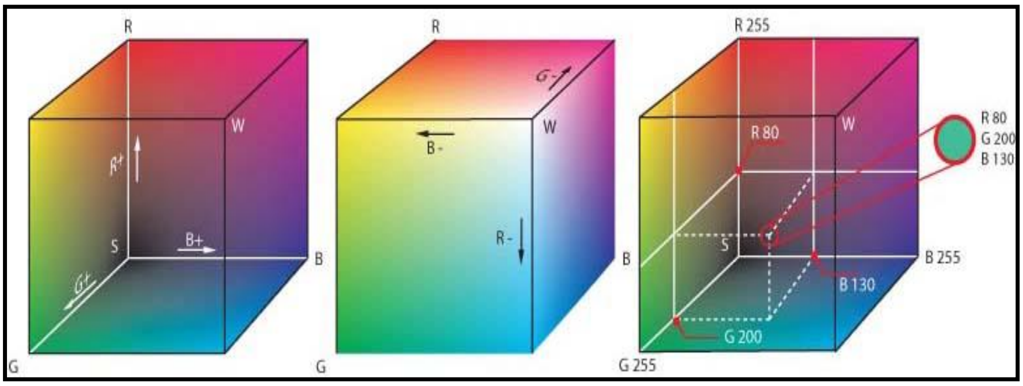


Figure 14 ***The RGB cube[39]***

The primary colours red (R), blue (B) and green (G) form the coordinate system. The “pure” red colour (R), green (G), blue (B), cyan (C), magenta (M) and yellow (Y) lie on the vertices of the cube.

In numeric terms, a colour is expressed as an RGB triplet (r, g, and b), each component of which can vary from zero to a defined maximum value. If all the components are at zero the result is a black colour; if all are at maximum, the result is the brightest representable white.

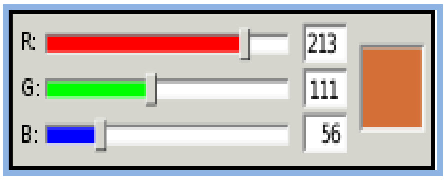
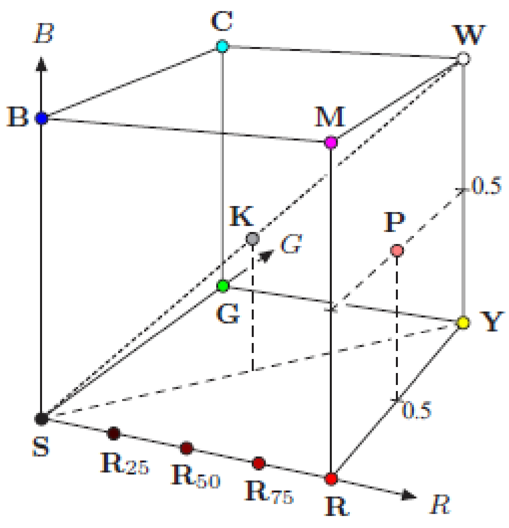


Figure 15 ***Numeric Representation of a colour using RGB model[33]***

2.2.2 RGB Colour Spaces

The RGB colour space is a non UCS colour space that uses properties of the three primary colours of red, green and blue. This colour space is based on the RGB colour model. In this colour space, these primary colours are combined using the principles of additive mixing to form new colours. The RGB colour space is the most common colour representation and it has been adopted in a large number of input/output devices for colour information. The two most common RGB colour representations are NTSC-(RGB) and CIE-(RGB) [40].

Creating a colour space is a compromise between the availability of good primaries, the signal noise, and the number of digital levels supported by the file type [29].As explained in the RGB model, in mathematical terms , the R, G and B form the axis of a three-dimensional colour space represented by a cube (shown below).



		RGB Value		
Point	Color	*R*	*G*	*B*
S	Black	0.00	0.00	0.00
R	Red	1.00	0.00	0.00
Y	Yellow	1.00	1.00	0.00
G	Green	0.00	1.00	0.00
C	Cyan	0.00	1.00	1.00
B	Blue	0.00	0.00	1.00
M	Magenta	1.00	0.00	1.00
W	White	1.00	1.00	1.00
K	50% Gray	0.50	0.50	0.50
R$_{75}$	75% Red	0.75	0.00	0.00
R$_{50}$	50% Red	0.50	0.00	0.00
R$_{25}$	25% Red	0.25	0.00	0.00
P	Pink	1.00	0.50	0.50

Figure 16 Representation of the RGB colour space as a three-dimensional unit cube [38]

In this generic RGB colour space cube, black (S) is located at Cartesian coordinates (0, 0, 0) and white (W) at (1, 1, 1). The other corners of the cube have the three primary colours of Red, Blue and Green at (1, 0, 0), (0, 1, 0), and (0, 0, 1) respectively. The pair wise sums of these primaries are known as secondary colours (cyan, magenta and yellow) and are located at the other three corners of the cube.

Most applications using the RGB colour spaces restrict the values to the range (0, 1). The unit cube is therefore the gamut of colours defined for that colour space [17]. In the centre of the cube, where the values of the primary colours are similar, shades of gray are produced.

RGB colour spaces have evolved over time, sometimes for technological reasons (NTSC to SMPTE-C) , sometimes due to professional requirements (ColourMatch, Adobe RGB) and sometimes because that was the initial space used in colour displays (Apple RGB) [29].

Descriptions of RGB spaces that are closely related to this research are presented in the following sections.

2.2.2.1 NTSC RGB

The NTSC colour space was developed by the National Television System Committee of North America. This was the colour space introduced in one of the first North American TV Sets and was probably the only colour space to realise complete separation between luminance and the chrominance information [41, 42]. This colour space converted the RGB coordinate system to the YIQ [42] colour space that is closer to human perception, where ‘Y’ is the yluminance (y is silent) , ‘I’ is the in-phase and ‘Q’ stands for quadrature . The Y component represents the luma information; I and Q represent the chrominance information. The transformation applied to convert the RGB to YIQ colour space was:

$$\begin{aligned} \mathbf{Y} &= \mathbf{0.29YR + 0.587G + 0.114B} \\ \mathbf{I} &= \mathbf{0.596R - 0.275G - 0.321\ B} \\ \mathbf{Q} &= \mathbf{0.212R - 0.523G + 0.31\ 1\ B} \end{aligned}$$

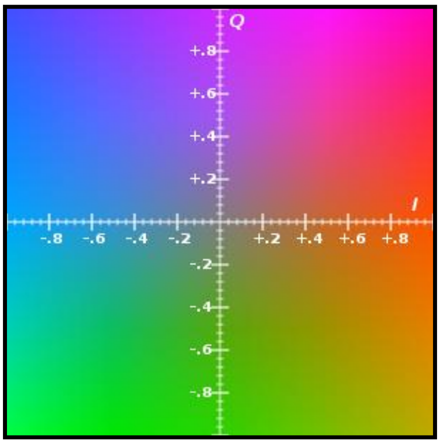


Figure 17 The YIQ colour space at Y=0.5 [43]

This colour space is now obsolete and has been replaced by SMPTE-C colour space [29].

2.2.2.2 ‘sRGB’ Colour Space

Standard RGB Colour Space (sRGB) is a standardised specification for RGB colour images in terms of a virtual display. It is a non-linear colour space that has been standardised by the International Electrotechnical Commission (IEC) as IEC 61966-2-1 [17].

The colour space was initially created cooperatively by HP and Microsoft in 1996 [44] for use on monitors , printers and the internet . It is being adopted as a general-purpose colour space for consumer use, where embedding the space profile may not be convenient for file size or due to compatibility issues [29].

It is widely being used in the World Wide Web and the sRGB specification considers the images being viewed under office environment.

Shown below are the chromaticity coordinates for the sRGB space with respect to CIE XYZ (explained later) and the chromaticity diagram of sRGB.

Chromaticity	**Red**	**Green**	**Blue**	**White (D65)**
x	**0.6400**	**0.3000**	**0.1500**	**0.3127**
y	**0.3300**	**0.6000**	**0.0600**	**0.3290**
z	**0.0300**	**0.1000**	**0.7900**	**0.3583**

Table I The sRGB colour specification[44]

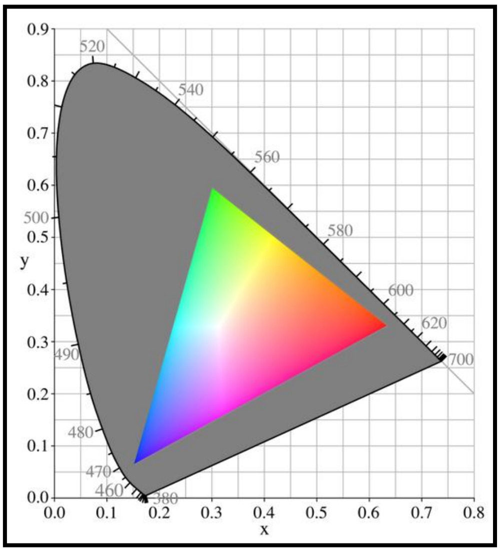


Figure 18 The sRGB chromaticity coordinates[45]

2.2.2.3 CIE1931/ CIE RGB/ CIE XYZ Colour Space

In colour perception , the CIE 1931 XYZ colour space is one of the first mathematically defined colour spaces [46] . It was derived as a result of various experiments that were conducted by W.David Wright [47] and John Guild [48].Their experimental results were combined into one specification, the CIE RGB colour space. It was from CIE RGB that the CIE XYZ colour space was derived. In this colour space, a set of measured colour matching functions are transformed to create a set of curves that are more convenient to use. These curves are named as $\bar{x}$, $\bar{y}$ and $\bar{z}$ as shown in fig below:

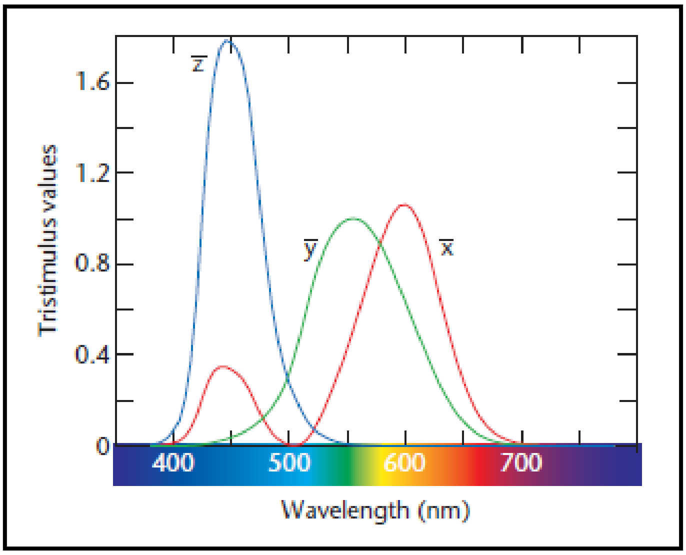


Figure 19 CIE recommended colour matching functions (1931)[17]

Throughout the entire visible spectrum, this set of curves is positive. The curve $\bar{y}$ can be used to compare the perceived brightness of the measured colour, called its luminance. The function $\bar{z}$ is zero for most wavelengths.

In this colour space instead of the three tristimulus values (S, M, and L) (ref to figure 10) of the eye we have a set of tristimulus values called x, y and z, which were computed from the CIE colour matching functions of $\bar{x}$, $\bar{y}$ and $\bar{z}$ .These tristimulus are additive and they can also be scaled. Therefore, the colour matching values can be transformed using linear algebra. As tristimulus values are vectors in a linear 3-D, it is relatively simple to transform to a new set of primaries by changing of basis (it requires values of new primaries with respect to old ones). This is the link that ties XYZ to the physical RGB values of a computer display[49].

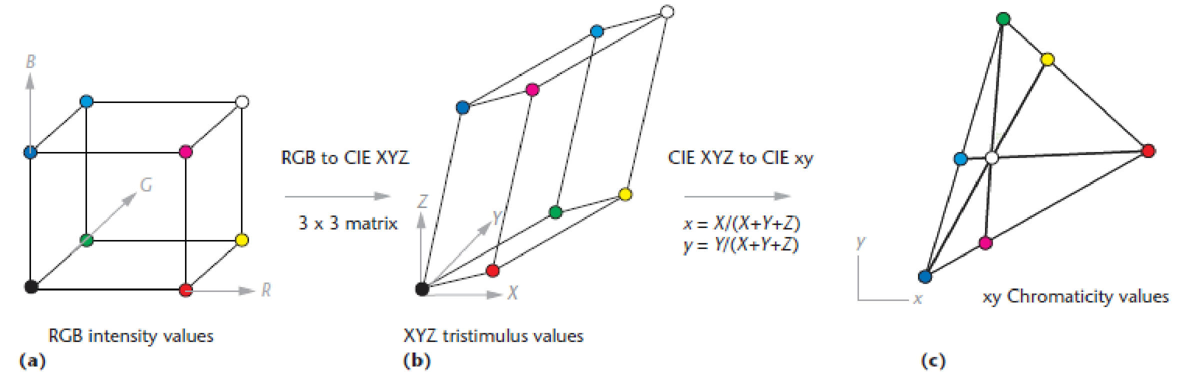
Characterisation of an RGB space with respect to XYZ is done in two easy steps:

- Measure the R, G, B primaries.
- Setup the corresponding linear transformation (represented by a 3x3 matrix) as shown below:

$$\begin{bmatrix} R & G & B \end{bmatrix} M = \begin{bmatrix} X & Y & Z \end{bmatrix}$$

$$M = \begin{bmatrix} X_R & Y_R & Z_R \\ X_G & Y_G & Z_G \\ X_B & Y_B & Z_B \end{bmatrix}$$

Linear transformation from RGB to XYZ [49]



(a) Converting the RGB color cube to XYZ. (b) This transformation scales, rotates, and skews the cube. The XYZ values can be mapped to the chromaticity diagram via the usual formulas. (c) The result is a triangle, whose vertices are red, green, and blue. White and black lie on the same point near the center.

Figure 20 RGB to CIEXYZ linear transformation and CIEXYZ to RGB non-linear transformation[49]

2.2.2.4 HSV and HLS Colour Spaces

HSV (‘Hue’, ‘Saturation’, ‘Value’) also called HSB (B being ‘Brightness’), and HLS (‘Hue’, ‘Lightness’, ‘Saturation’) also referred to as HSL , are more perceptual organisations of the RGB colour cube [17].Both these models were first defined by Joblove and Greenberg in 1978 [50] .They are two of the most common cylindrical-coordinate representations of points in an RGB colour model and are considered to be closely related to human perception of colour [51] .The transformation of the RGB colour space to the HSV colour space is a conversion of a set of rectangular coordinates to a set of cylindrical coordinates [52].

Hue is considered as a circular or modular quantity, while other characteristics of a colour imply existence of minimum and maximum values. Hence points in HSV colour space related to these qualities are specified by cylindrical coordinates[50].

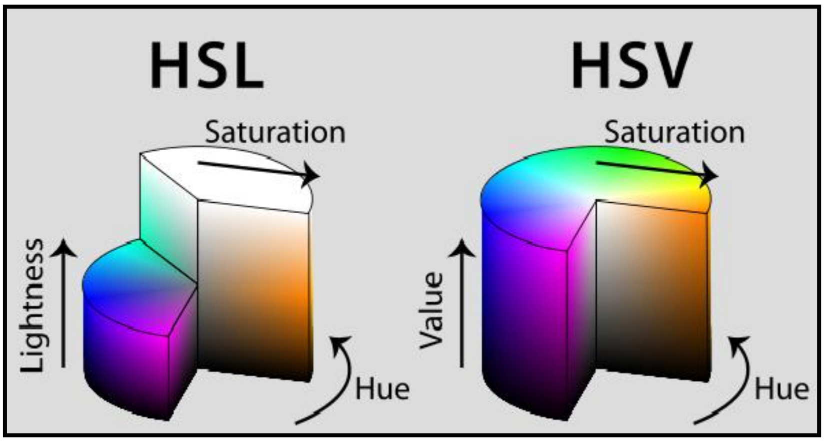


Figure 21 The cut-away 3d models of HSL and HSV [53]

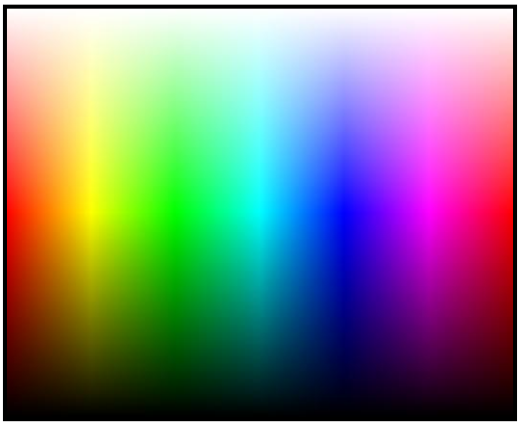


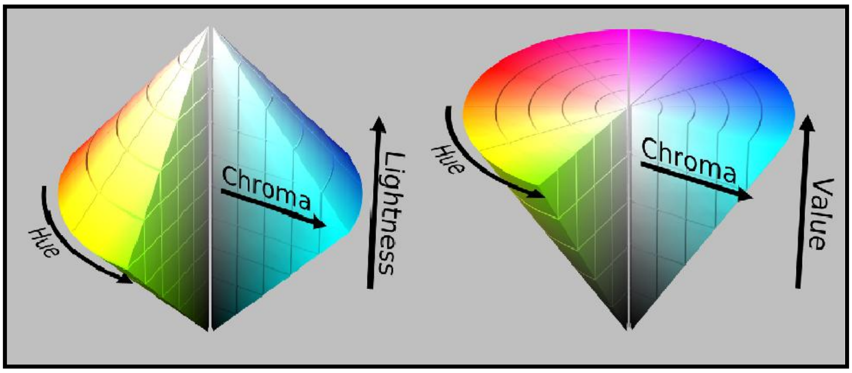
Figure 22 HSV colour space (mapped to 2D, not including greyscale) [54]

However some authors represent HSV and HLS by a hexacone model [51, 55], this is mainly due to the definition of saturation – in which very dark or very light neutral colours are considered fully saturated. In these models saturation is replaced by Chroma, thus resulting in a cone model.

A colour in the HSV colour space is considered as a vector having H, S and V as its components. Colours with maximum density, considered as 'Pure colours' are placed on the perimeter of the hexagonal base of HSV hexacone, where S= V= 1.

The HSV colour space describes a hexagonal cone with black at its tip and white in the centre of the flat face. All the other colours including white have V=1, even though their perceived brightness is quite different.

HLS is structured as a double ended cone, which has the same cross section as HSV but the primary and secondary colours are organised at the middle of the cone [17].



When HSL/HSV hue and HSL lightness/HSV value are plotted against Chroma C;

Where C = max(R, G, B) − min(R, G, B) – the resulting colour solid is a bi-cone or cone.

Figure 23 The cut-away bi cone model for HLS and HSV [53]

2.2.2.5 Conversion of RGB space to HSV colour space

The HSV colour space representation is considered to be more closely related to human perception of colours [56]. In other words the role of each component in HSV space is more understandable by human beings[56]. Due to this quality, it is easier to perform colour analysis on the HSV space than any other space. In the human vision system, viewing of a colour object is characterised by its brightness and chromaticity, whereas in the colour vision model it is defined by hue and saturation. Brightness is a subjective measure of luminous intensity that embodies the achromatic notion of intensity[35]. In the HSV hexacone colour model, the attributes are converted from RGB space as follows [35, 57]:

$$H = \begin{cases} 60\,((G-B) / \partial) & \text{if } MAX = R \\ 60\,((B-R) / \partial + 2) & \text{if } MAX = G \\ 60((R-G) / \partial + 4) & \text{if } MAX = B \\ \text{Not defined} & \text{if } \partial = 0 \end{cases}$$

$$S = \begin{cases} \partial / MAX & \text{if } MAX \neq 0 \\ 0 & \text{if } MAX = 0 \end{cases}$$

$$V = MAX$$

Where ∂ = (MAX-MIN),

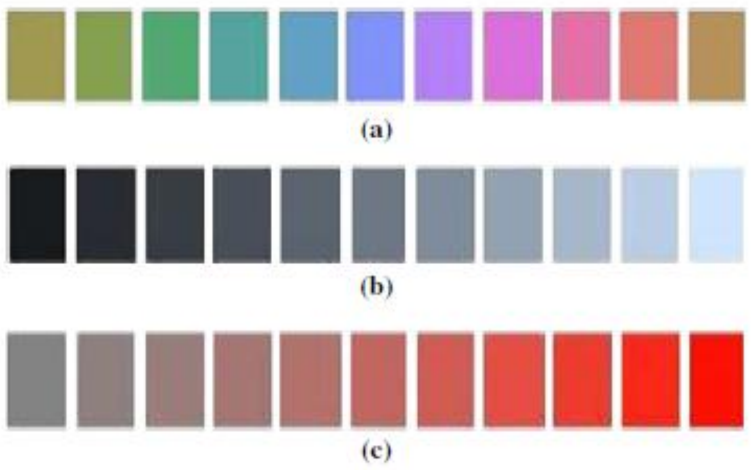
MAX = max (R, G, B) and

MIN = min (R, G, B). R, G, B values are scaled to [0, 1].

In order to confine H within range of [0,360], H= H+360, if H<0.

2.2.3 HSI Colour Model

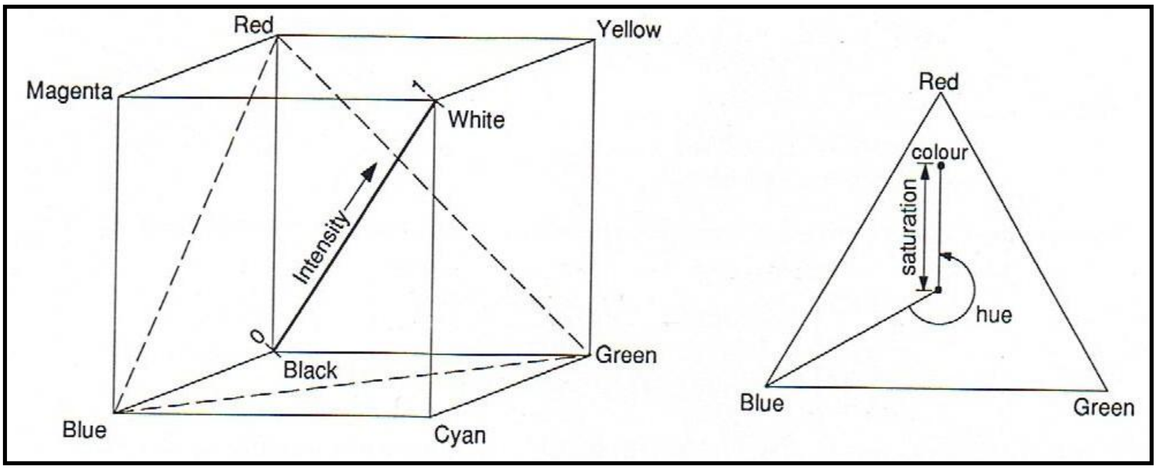
The HSI colour model is a perceptual colour space that represents colours based on hue (H), saturation (S) and intensity (I). It is also sometimes referred to as the IHS model [58]. In a mixture of light waves, hue is associated with the dominant wavelength. Saturation is the amount of white light mixed with the dominant colour, and intensity represents the perceived illumination [59]. The HSI colour model is quite similar to the HSV and HSL models, hue and saturation being the common attributes in all three models. Intensity, being a different attribute from HSV colour space, represents the extent to which light is reflected from an object. Intensity is an achromatic component of colour that is considered as a measure of total reflectance in the visible region of spectrum [31].The HSI model can also be defined as a model that defines colours as co-ordinates of hue and saturation, on equal intensity planes [13]. Figure below, shows different attributes of the HSI model:



(a) Different hues(dominant colour); (b) different intensities represented by this greyscale; (c) A variety of saturation of red colour [59]

Figure 24 Hue, saturation and intensity

The hue is measured as an angle around the plane and saturation as the radial distance from the centre of the plane, shown below:



(a) (b)

(a) Intensity measured along the diagonal axis between black and white.
(b) Saturation is the linear distance from the intensity axis to the selected colour on an equal intensity plane.

Figure 25 HSI Coordinates [13].

2.2.3.1 HSI attributes in relation with RGB components

Considering R, G, B as real numbers in the range 0-1, the following formulas are used to transform an RGB space to HSI space [60]:

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

$$\mathbf{S = 1- [3/(R + G + B)\ \min(R, G, B)]}$$

$$\mathbf{H = \arccos\left(\frac{1/2 \times [(R - G) + (R - B)]}{\sqrt{(R- G)^{2} + (R- B)\ (G - B)}} \right)}$$

If B > G, then H = 2π – H.

The conversion process of RGB to HSI is similar to a transformation from rectangular to polar coordinates. A new axis is placed in the RGB space between (0, 0, 0) and (1, 1, 1). This axis passes through all the achromatic points and is therefore called achromatic axis.

2.2.4 CIECAM Colour Model

As described earlier (Section 2.2) CIECAM97 is one of many colour appearance models that were recommended by the CIE Technical committee. The main objective of such colour appearance models is to ensure that various digital devices can produce reliable colours, closely related to the human visual system[61] .

CIECAM97 currently is only developed for a single input colour and a certain illumination environment. In a specified viewing condition it uses the colour stimulus, the colour of white, the background, and the surround field to calculate its representation of the stimulus. In CIECAM97 the stimulus colour is represented by lightness, chroma and hue. It can be used to transform a colour to a different viewing condition, after the lightness, hue, chroma and viewing information for the colour stimulus has been computed. The ultimate goal of the CIECAM97 model is for the two colours to appear exactly the same [14]. Shown below is, the diagram of input/ output of a CIECAM:

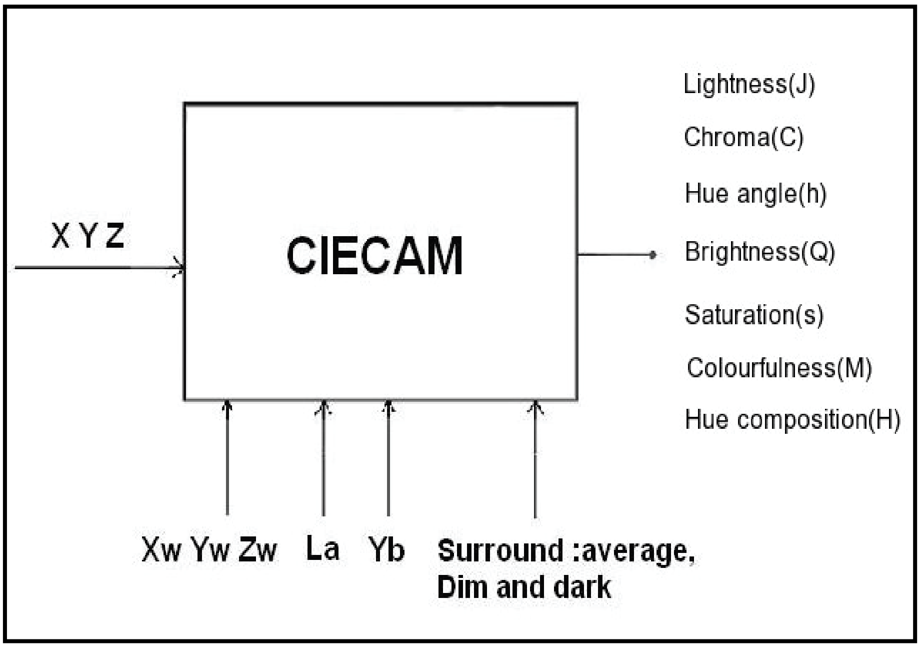


Figure 26 Diagram of CIECAM [62]

As per the diagram shown above, CIECAM97 parameters are as follows:

Input Data

- XYZ: Relative tristimulus values of colour stimulus
- L_a: Luminance of the adapting field (cd/m^2)
- $X_wY_wZ_w$: Relative tristimulus values of white
- Y_b: Relative luminance of the background

Output Data

- J: Lightness: Overall illumination of the colour
- C: Chroma: Related to colourfulness
- h: Hue Angle: Classifies the colour as an angle, in degrees
- Q: Brightness :
- S: Saturation
- M: Colourfulness
- H:Hue

The CIECAM02 is the latest colour appearance model by the CIE and is based upon the basic structure of CIECAM97s [61].

It is one of the important colour models that have been used for traffic sign detection by some researchers, but due to the limitation of Emgu CV we could not use this model for our experiments. We have still reviewed this model here and in the next chapter, as this model has been proposed by some as suitable for traffic sign detection.

The next chapter will discuss the methods reviewed by the author for detection of speed signs using different colour spaces including RGB, HSV and CIECAM.

3 Literature Review: Papers Referred and Method Justification

There are two main stages in identifying a road sign [63, 64], namely detection and recognition. In the detection stage, road sign images are pre-processed, enhanced and segmented by using properties like colour or shape. The resulting images contain potential regions which are further tested in the recognition stage, against certain parameters to be able to determine the existence of signs in the images.

Here we look at the current and past work that has been conducted in the field of road sign recognition.

3.1 2006: Shadow and Highlight Invariant Colour [63]

This paper was published in 2006 and discussed the challenges faced by computer vision researcher while dealing with shadows and highlights in an outdoor environment [63]. The main problem with the images is having shadows on the objects (in this case a traffic sign). This causes the objects to be exposed to different illumination levels and hence makes it difficult for the computer vision programs to be able to distinguish them from other objects in the image. In case of highlighted objects, the light of the illuminant (the sun) is directly reflected to the viewer (in this case the camera) by the object. Therefore it has an increased brightness, thus making it difficult for a computer program to recognise the object.

This research performed the following steps to deal with the problem of shadow and highlight in images:

- The RGB images were converted to HSV colour space

- The shadow- highlight invariant method was applied to these converted images to extract the colours from the sample images of road signs.

3.1.1 Colour Variation in outdoor images

According to the author one of the most difficult problems related to using colours in outdoor images is the chromatic variation of daylight [63]. The colour information is very sensitive to variations in lighting conditions caused by changes in weather [64]. Due to this chromatic variation, the apparent colour of the object varies as daylight changes. The author describes how this irradiance of an object present in a colour image, depends on three parameters:

i. ***The colour of the incident light:***

The variation of daylight's colour (y) along a CIE curve is given by

$y = 2.87x - 3.0x^2 - 0.754$ *for* $0.25 \leq x \leq 0.38$, x is just a variable

The variation of daylight's colour is a single variable which is independent of the intensity.

ii. ***The reflectance properties of the object:***

The reflectance of an object $s(\lambda)$ is a function of the wavelength λ of the incident light, given by:

$s(\lambda) = e(\lambda)\,\phi(\lambda)$

Where, $e(\lambda)$ *is the intensity of light at wavelength* λ *and* $\phi(\lambda)$ *is the object's albedo (reflection coefficient) at each wavelength.*

iii. ***The camera properties,*** are defined by the lens diameter ***d***, its focal length ***f*** and the image position of the object measured as an angle ***a*** off the optical axis. The observed intensity ($E(\lambda)$) is given by :

$$E(\lambda) = L(\lambda).(\pi / 4)(d / f)^2 \cos(4a) \qquad \text{.................... (i)}$$

Where, $L(\lambda)$ *is the radiance which is multiplied by a constant.*

(λ)*, is the a lights wavelength*

Cancelling the camera's lens chromatic aberration, only the density of the observed light is affected, resulting in the colour of the light reflected by an outdoor object as a function of object's albedo , the temperature of the daylight and the observed irradiance, which is the reflected light surface scaled by the irradiance equation [65, 66].

3.1.2 Hue Properties and illumination changes

In this section, the author describes hue as one of the important colour space attributes that is invariant to the variation in light conditions. The properties which make 'Hue' a special attribute, for being invariant to illumination changes, are:

- Hue is multiplicative/scale invariant
- Hue is additive/shift invariant
- Hue is invariant under saturation changes.

Also described in this section is the instability of hue coordinate which can be caused by a small change in RGB, this is because 'Hue' belongs to a colour space that is a derivative of RGB colour model and hence is dependent on RGB. The main problems that hue suffers from are:

- It becomes meaningless when the intensity (I) is very low or very high
- It also is meaningless when the saturation (S) is very low
- 'Hue' becomes unstable when saturation goes below a certain threshold

Hence to overcome these problems of hue and to be able to use it as a meaningful parameter in detecting illumination changes, Vitabile et al [67] have defined three different areas in a HSV colour space and the author recommends to use those areas to achieve robustness in any colour segmentation system [63].

The three defined areas in HSV colour space are as follows:

- ***The achromatic area**: characterised by $s \leq 0.25$ or $v \leq 0.2$ or $v \geq 0.9$*
- ***The unstable chromatic area**: characterised by $0.25 \leq s \leq 0.5$ and $0.2 \leq v \leq 0.9$*
- ***The chromatic area**: characterised by $s \geq 0.5$ and $0.2 \leq v \leq 0.9$.*

3.1.3 Reflectance with white illumination

In this section the author has described the reflection model of a colour image taken by a camera, the effect of shadows on colour invariance (tested for different colour spaces) and the effect of highlights on colour invariance.

To describe *the reflection model*, the author has derived a mathematical relation for a sensor response of a camera using the figure below:

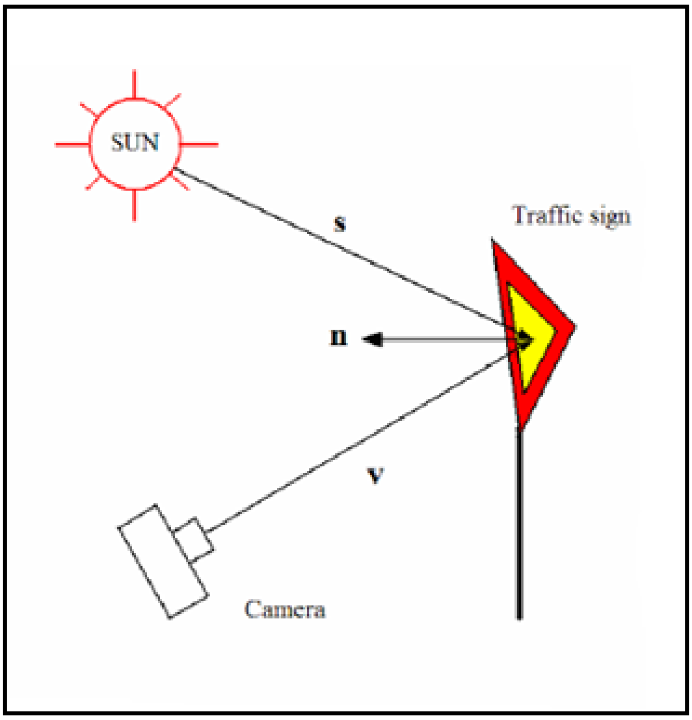


Figure 27 Traffic Sign Scene [63]

n,s,v are unit vectors representing the direction of normal vector to the surface patch, direction of source of illumination, and direction of the viewer, respectively.

The reflection of a surface, as derived is given by:

$C_w = e\, m_b\, (\mathbf{n}, \mathbf{s})\, k_C + e\, m_s\, (\mathbf{n}, \mathbf{s}, \mathbf{v})\, c_s f$................. (ii)

Where,

- C_w *is the sensors response*
- m_b *and* m_s *denote geometric dependencies of the body and surface reflections component respectively.*
- $k_C = \int_\lambda f_C(\lambda)\, c_b\, d\lambda$
- c_s *and* e *are constants*
- $f = \int_\lambda f_R(\lambda) d\lambda = \int_\lambda f_G(\lambda) d\lambda = \int_\lambda f_B(\lambda) d\lambda$ *in case of white illumination*

Considering body reflection term; $C_b = e\, m_b\, (\mathbf{n}, \mathbf{s})\, k_C$ for $C_b = \{R_b, G_b, B_b\}$ in equation (ii), the colour perceived by a sensor depends on

- Sensor response and surface albedo (k_C),
- Illumination intensity e and
- Object geometry m_b (**n**, **s**).

The *effect of shadows* (surface illuminated by different levels of brightness) on colour invariance was studied on four of the colour spaces including RGB, NRGB, HSI and HSV. It was concluded that hue and saturation are only two attributes that are affected by sensor response and surface albedo, i.e. they are invariant to shadows, hence suitable for images with shadow in them.

The *effect of highlights on colour invariance* has been described as follows:

While viewing an image, when a viewer is situated in a position where the angle between vectors ***nv*** and ***ns*** (as shown in figure above) are approximately equal, then the viewer receives two components of light:

i. A light from the source reflected on the surface of the object to the viewer called the highlight

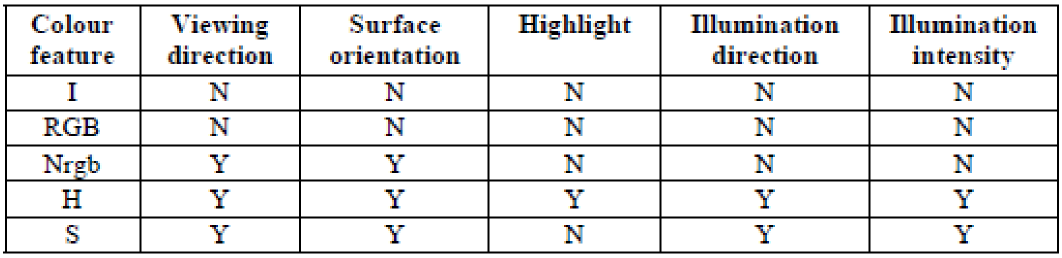
ii. The light reflected by the object itself.

The highlight component is generated due to the surface acting as a mirror, thus reflecting the light from the source to the viewer.

As per the above equation (ii), adding the surface reflection component $C_s = e\, m_s (\mathbf{n}, \mathbf{s}, \mathbf{v})\, c_s f$ with the body reflection component $C_b = e\, m_b (\mathbf{n}, \mathbf{s})\, k_C$ gives a collective measure of colour.

Testing different colour spaces according to the aforementioned argument showed that only the hue feature was invariant for surface reflection component and hence invariant for the highlight component of the perceived colour. It also showed that hue depended only on sensor response and surface albedo.

Whereas the saturation varied with the surface reflection component, because of which it could not be used for developing any robust algorithms for highlight problem. Shown below is the table, given by the author that shows comparison of different colour models and their abilities to be invariant to different imaging conditions:



Colour feature	Viewing direction	Surface orientation	Highlight	Illumination direction	Illumination intensity
I	N	N	N	N	N
RGB	N	N	N	N	N
Nrgb	Y	Y	N	N	N
H	Y	Y	Y	Y	Y
S	Y	Y	N	Y	Y

Table II ***Invariance of colour models to imaging conditions***

Here 'Y 'denotes an invariant colour model and 'N' denotes sensitivity of the colour model to that imaging condition.

3.1.4 Colour Segmentation Algorithm

In this section of the paper the algorithm that was used has been mentioned.
The main points of the algorithm, as given in the paper, are as follow:

i. Convert RGB images to HSV space.
ii. Normalise the Brightness (I), Hue (H) and Saturation (S) into [0,255].
iii. Convert pixels to white (255) or black (0) depending on the required H value ranges.
iv. Generate a sub image of 16x16 pixels from the main image.
v. For every sub image,
    a. Calculate number of white pixels
    b. If number of white pixels >= 60, then put a white pixel in the corresponding position in the seed image.
vi. Seed image and H image are used for a region growing algorithm to find proper regions with sign.

3.1.5 Results

It proves that hue and saturation are invariant to the effects of illumination variations and recommends using them in developing shadow-invariant algorithms for colour segmentation. It also uses saturation and value for defining chromatic sub space, in which hue could be used. The experiments conducted achieve a 95% success rate but fail when pixel is not in the chromatic subspace of HSV colour space.

3.2 2008: New Hybrid Technique for Traffic Signs [68]

Road signs contain very important and valuable information, which can be used to assist drivers in avoiding any kind of danger of collision on road and to control the speed of vehicle. These road signs are specially designed in shape and colour [68] to regulate traffic laws like speed limits and assist drivers while driving in different conditions. However, drivers still make mistakes due to carelessness and ignorance of traffic signs. Hence, an automatic computer aided vision- based traffic sign recognition (TSR) system has been proposed here to assist drivers. In general, TSR systems are composed of sign detection and classification stages, where these systems extract traffic signs from the complicated moving road scene and verify segmented results with various image processing techniques like edge detection, colour detection and pictogram.

Various detection techniques have been proposed for road sign detection and have been implemented using neural networks [69, 70]. The classification with purely neural network is not sufficient to recognise a large number traffic signs; hence authors propose a new TSR system that will be less complex as compared to old systems and it uses knowledge based approach and neural classifier approach.

This paper proposes a hybrid traffic sign recognition scheme combining of knowledge based analysis and radial basis function neural classifier (RBFNN) [68]. Here the authors use colour segmentation as an initial method to detect the signs from the road scenes and have reported a three staged recognition technique in this paper:

- Stage I: colour histogram classification
- Stage II: Shape Classification
- Stage III: RBF neural classification

We will discuss the first two stages in detail and the third stage will be only briefly described (as this is out of the scope of this research). Also the results of this paper will be analysed in this section.

3.2.1 Overview of the system proposed

The proposed system was designed to detect and recognise traffic signs in Malaysia. In this system, classification is first performed based on the colour and shape information acquired from the images. From this knowledge based analysis some signs are easily detected due to their unique appearance, hence reducing the traffic sign database into smaller groups. Then, within each subclass, the traffic sign features are extracted using PCA (Principle Component Analysis) and FLD(Fisher's Linear Discriminant). After this, they are passed to the RBF neural classifier for further recognition. This leads to better performance as compared to previous systems.

3.2.2 Traffic Sign Detection

In this section, the authors describe the colour segmentation technique used to obtain a road sign from its background. Colour segmentation is a popular method in traffic signs extraction, due to its low complexity and low computational cost [69, 71].

Here the authors have used the RGB colour space for colour segmentation. According to the authors of this paper, this colour space is quite susceptible to illumination, thus affecting an image's recognition capability. Hence, the authors propose a normalisation method for the colour channels in the RGB colour space.The normalisation of the colour space is performed by using the Intensity, I, which is given by:

$$I = (R' + G' + B') / 3$$
$$r = R' / I$$
$$g = G' / I$$
$$b = B' / I$$

Where,

- R', G', B' are colour channels of an image
- **r, g and b** are the normalised red, green and blue channels of the input image.

Four colour masks (red, blue, green and yellow) are constructed based on equations given in [72],shown below:

$$R = r - (g + b) / 2$$
$$G = g - (r + b) / 2$$
$$B = b - (r + g) / 2$$
$$Y = (r + g) / 2 - |r - g| / 2 - b$$

These newly constructed colour channels help in discriminating the dominant colours from the background in the sample image.

The dominant colours that are discriminated due to normalisation in the image are shown below:

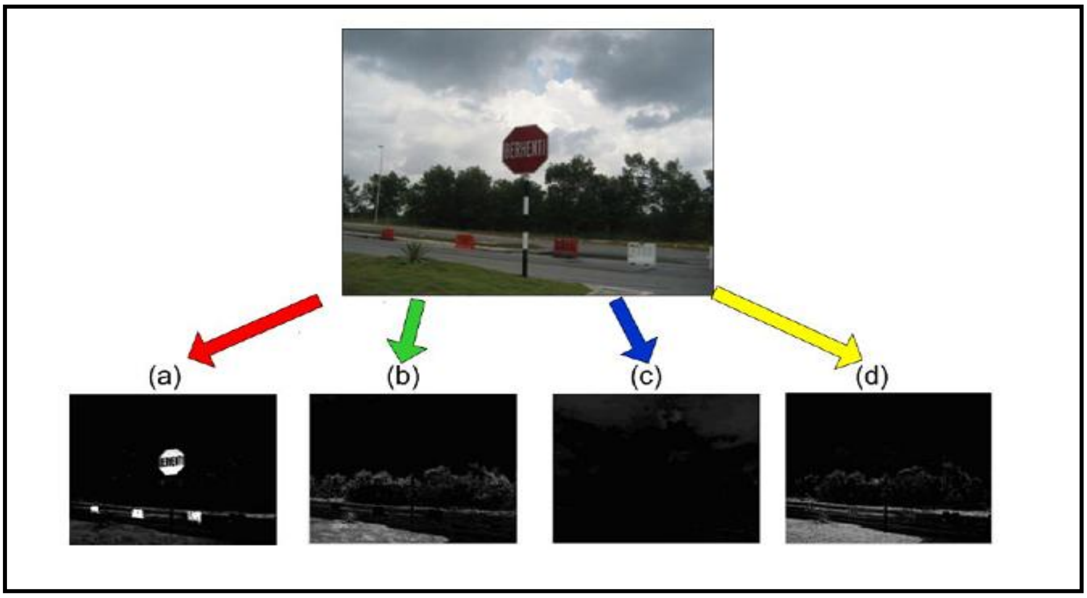


Figure 28 Traffic sign segmentation on road scene (a) red, (b) green (c) blue and (d) yellow colour maps [68].

Once the colour maps are generated, they are transformed into binary images using colour thresholding. Then morphological operations are performed on binary images in order to erode unwanted pixels and to dilate the Region of interest (ROI). Then using pictogram, in an ROI, the extracted region is verified as traffic sign depending on the amount of white or black pixel content of the pictogram.

3.2.3 Proposed System

The figure below shows the proposed system [68] with a new hybrid recognition technique combining knowledge based method and neural classifier

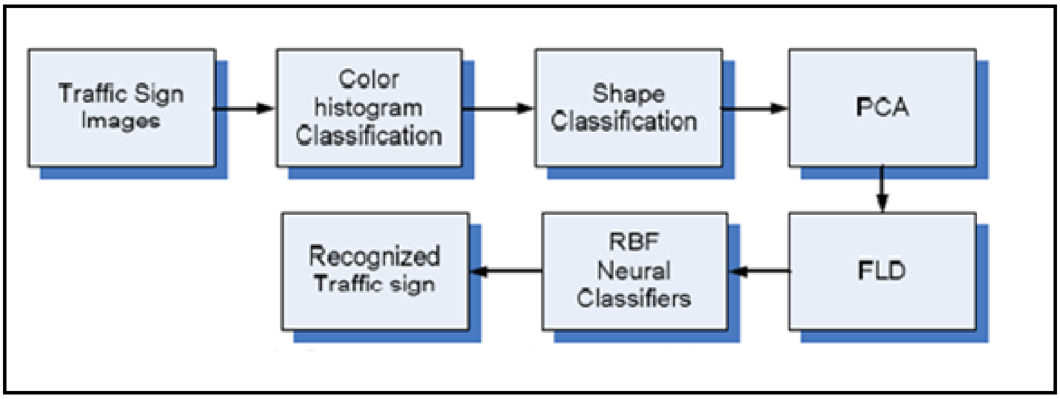


Figure 29 The proposed hybrid recognition system [68]

3.2.3.1 Colour Histogram and Shape classification

Colour is an important element for traffic sign classification and it is important to identify colour correctly. Some road signs can be easily recognised depending on their colour uniqueness. In [68] divide road signs are divided into subclasses, based on their colour histogram classification. Four colour bins were created, yellow/orange, red, blue, and green; the traffic signs are assigned to these bins based on the maximum value of a type of colour pixel in the sign. This has been shown below:

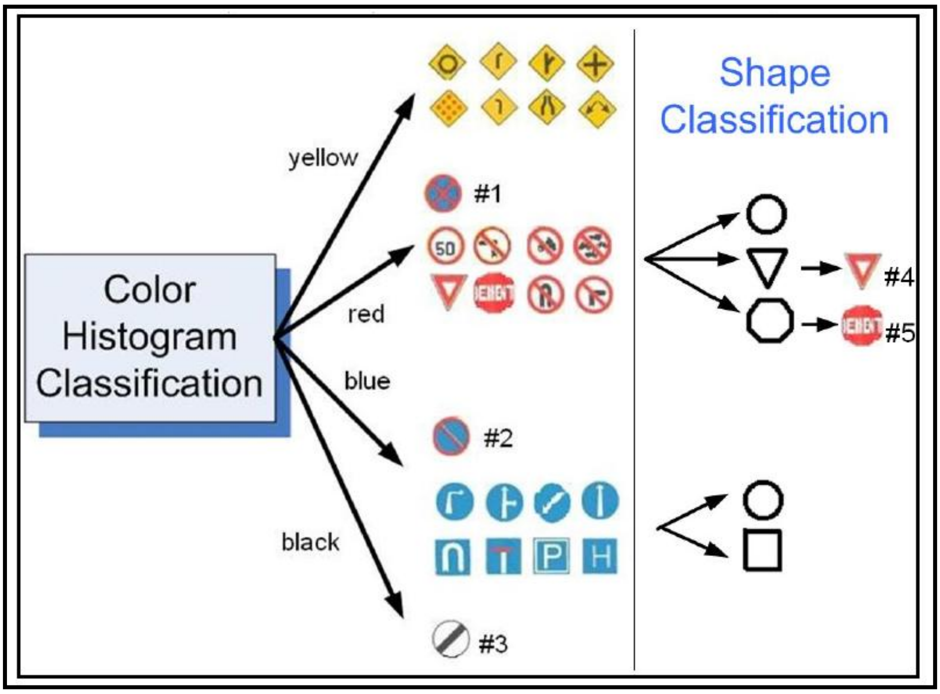


Figure 30 Colour histogram and shape classification [68]

As shown above traffic sign #1, #2 and #3 could be recognised by the colour histogram classification. Traffic sign 1 had maximum red component in it hence was placed there; subsequently #2 was high in blue pixels and some red hence was recognised by blue histogram. #3 was a black and white sign.

Besides colour, shape recognition was applied in this paper to further split the large traffic sign classes into smaller groups. Shape classification was performed on red and blue colour bins only. Due to shape classification sign #4 and #5 were easily recognised.

After the colour and shape classification the traffic signs that could not be recognised, were processed through the following steps:

1. Principal Component Analysis (PCA) was performed to reduce the high dimensional traffic sign data and generate the most significant features.

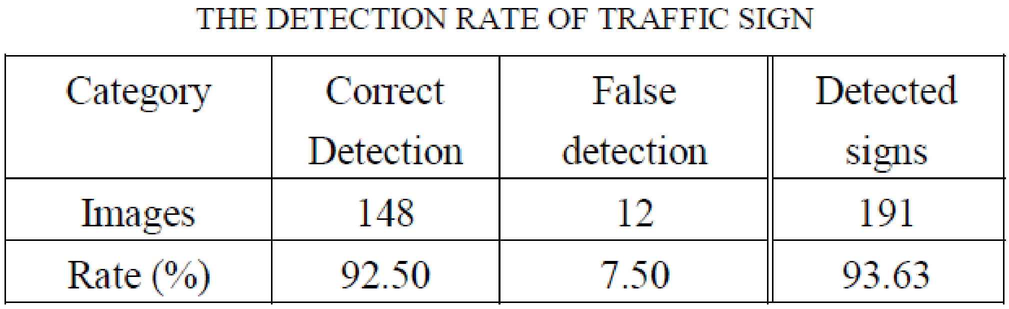
2. Fisher’s Linear Discriminant (FLD) was applied to obtain the most discriminant feature of traffic signs. It was used to find a linear projection of the set of training samples in the Eigen space to find an optimal low- dimensional subspace.

3. The FLD features were passed to the RBF Neural Network which was used to classify the subclasses generated after colour and shape classifications.

3.2.4 Results and Analysis

The colour segmentation was performed in simulation on 160 images with 204 traffic signs in them. It could correctly detect signs with a rate of 92.5 % and also had a false detection rate of 7.5 %.

The results are shown below:



THE DETECTION RATE OF TRAFFIC SIGN

Category	Correct Detection	False detection	Detected signs
Images	148	12	191
Rate (%)	92.50	7.50	93.63

Table III Results from the simulation[68]

Over all it could detect 93.63% signs using colour segmentation and 95.29% for whole system using RBF Neural classifier.

This paper shows that colour segmentation using RGB normalisation can reduce complexity of a program and can also increase detection speed with a good accuracy.

3.3 2008: Colour vision Model based approach Segmentation for Traffic Signs [73]

It is very important to recognise traffic signs correctly and at the right time to make a journey safe [73]. Occasionally, depending on the viewing conditions, traffic signs can hardly be spotted or recognised by the drivers. This gives rise to a need for an automatic system to assist car drivers for recognition of traffic signs.

As described earlier, colour information being a dominant visual feature, is widely used for segmentation of images in traffic sign recognition systems[74]. Colour is not only regulated for traffic sign category but also for the tint of the paint covering the sign that corresponds to a specific wavelength in the visible spectrum [75]. Red, blue, green, yellow, orange, violet, brown and achromatic colours are the most discriminating colours for traffic signs [73, 76].

The authors here propose a colour vision model based approach for segmentation of traffic signs. Here we will discuss the approach taken and analyse the results.

3.3.1 Traffic Segmentation based on Colour

Researchers have developed different techniques to utilise colour information cointed in a traffic sign [68, 77]. The colour spaces used in traffic sign detection are usually HSI, HSV and L* a* b*. These colour spaces are mainly based on one lighting condition of D65 (white light). Hence the range of each colour attribute, such as hue is limited. This is due to the changes in weather conditions which change colour temperatures from 5000K to 7000K [73]. Change of weather condition, such as sunny, cloudy, and evening times when all sorts of artificial lights are present, causes the colour of the traffic signs to appear different [75], resulting in most colour based techniques for traffic signs segmentation to fail. Hence, the authors propose to use a segmentation system based on CIECAM97colour appearance model that is recommended by CIE.

3.3.2 CIECAM colour appearance model

The CIECAM models can predict colour appearance as accurately as an average observer and is expected to extend traditional colorimetry (CIE XYZ and CIELAB) to the prediction of the observed appearance of coloured stimuli under different viewing conditions. As explained earlier, the model takes into account the tristimulus values (X, Y, Z) of the stimulus , the luminance levels and other factors such as cognitive discounting of the illuminant [73]. The output of the model includes mathematical correlates for perceptual attributes that are lightness, brightness, colourfulness, chroma, saturation and hue. In this paper the colour attributes of lightness (J), chroma (C) and hue angle (h) are applied as follows:

$$J = 100\left(\frac{A}{A_w}\right)^{CZ},$$

$$C = 2.44s^{0.69}\left(\frac{J}{100}\right)^{0.67n}(1.64 - 0.29^n),$$

$$h = \tan^{-1}\left(\frac{b}{a}\right),$$

where

$$A = \left[2R'_a + G'_a + \left(\frac{1}{20}\right)B'_a - 2.05\right]N_{bb},$$

$$s = \frac{50(a^2 + b^2)^{1/2}100e(10/13)N_cN_{cb}}{R'_a + G'_a + (21/20)B'_a},$$

$$a = R'_a - \frac{12G'_a}{11} + \frac{B'_a}{11},$$

$$b = \left(\frac{1}{9}\right)(R'_a + G'_a - 2B'_a),$$

Where, R'_a, G'_a, B'_a are the post adaptation cone responses with detailed calculations in [78]. A_w is the A valued for reference white. N_{bb}, N_{cb} are constants and calculated as:

$$N_{bb} = N_{cb} = 0.725\left(\frac{1}{n}\right)^{0.2}$$

Where $n=Y_b/Y_w$, the Y values for the stimulus and reference white respectively.

As per the authors CIECAM model has not been applied to a practical application since its standardisation, hence they investigated it on segmentation of traffic signs and compared it with other models such as CIELUV, HSI and RGB.

3.3.3 Methods Used

3.3.3.1 Data collection

Image data collection, was carried out by using a high –quality Olympus digital camera. The images collected as data included sign images reflecting the variety of viewing conditions and the variations in sizes of traffic signs caused by the changing distances between traffic signs and the position to take pictures.

The viewing conditions consisted of:

- Weather conditions including sunny, cloudy and rainy
- The viewing angles with complex traffic sign positions as well as multiple signs at junctions; distorting the shape of signs

As per the UK Highway Code, the photos taken were between the distances of 10,20,30,40, and 50 meters.

3.3.3.2 Initial estimation of viewing conditions

Authors here classify images into three viewing conditions of sunny, cloudy and rainy to apply the CIECAM model. Considering all images been taken under similar driving conditions, one image consists of three parts from top to the bottom:

1. *Sky*
2. *Signs/scenes*
3. *The road surface*

If any image missed one of these parts it was considered to be a sunny, that could be easily corrected using the recognition stage. The degree of saturation of sky, which was determined using a sign database, was used to determine the viewing conditions of image as being sunny, cloudy or rainy. In addition, the texture of road was used to confirm the viewing conditions. The road's texture was measured using fast Fourier transforms with the average magnitude (AM) as threshold, as shown below:

$$\mathrm{AM} = \frac{\sum_{j,k} |F(j,k)|}{N}$$

Where

| F (j, k) | are the amplitudes of the spectrum, calculated by:

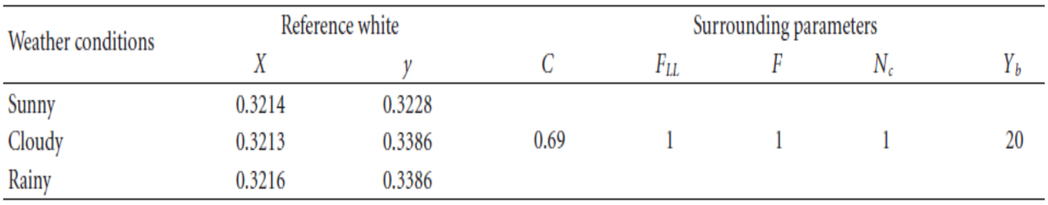
$$F(u,v) = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)\exp\left[-2\pi i\left(\frac{mu}{M}+\frac{nv}{N}\right)\right]$$

Where,

- ***f (m, n) is the image***
- ***n ,m are the pixel coordinates***
- ***N, M are the numbers of image row and column,***
- ***u,v are frequency components.***

3.3.3.3 Traffic Sign Segmentation

The reference white was obtained by measuring a piece of white paper many times during a period of two weeks using a colour meter, CS-100A, under each viewing condition. The average values are shown below:



Weather conditions	Reference white		Surrounding parameters				
	X	y	C	F_{LL}	F	N_c	Y_b
Sunny	0.3214	0.3228	0.69	1	1	1	20
Cloudy	0.3213	0.3386					
Rainy	0.3216	0.3386					

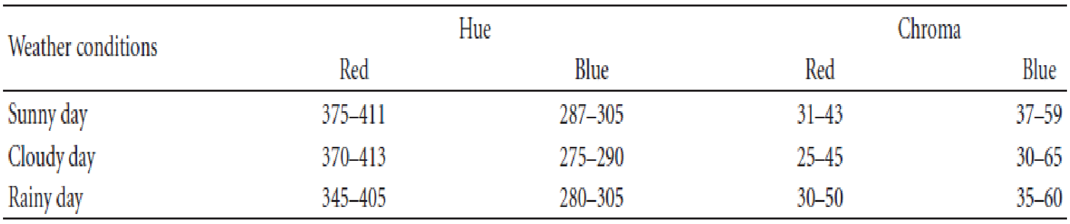
Table IV Reference parameters used in the paper

Also, the images were transformed from RGB to CIECAM using:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.2169 & 0.1068 & 0.048 \\ 0.1671 & 0.2068 & 0.0183 \\ 0.1319 & -0.0249 & 0.3209 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

Then these values were converted to LCH (lightness, chroma, hue).

The range of LCH was calculated for each weather condition as shown in the table below:



Weather conditions	Hue		Chroma	
	Red	Blue	Red	Blue
Sunny day	375–411	287–305	31–43	37–59
Cloudy day	370–413	275–290	25–45	30–65
Rainy day	345–405	280–305	30–50	35–60

Table V Range of colour attributes used for segmentation of traffic signs.

These values were the mean values ± standard deviations. Considering lightness as unchangeable with the change of viewing condition, only hue and chroma were employed in the segmentation. Those pixels that were within the range , were clustered together using algorithm of quad-tree histogram method [79], a method that recursively divides the image in to quadrants until all elements are homogeneous or until a predefined ‘grain’ size is reached.

3.3.4 Results

To evaluate the results of segmentation, two measures were used:

1. *Probability of correct detection, P_c:*

$$P_c = \frac{\text{numbers of segmented regions with signs}}{\text{numbers of total signs}}$$

2. *Probability of false detection, P_f:*

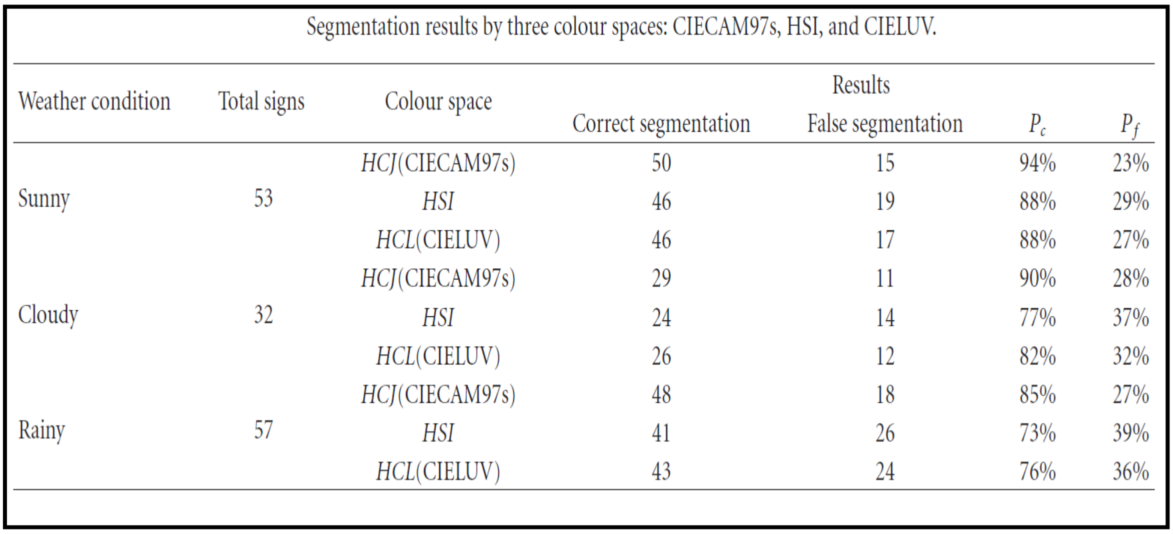
$$P_f = \frac{\text{numbers of segmented regions with no signs}}{\text{total number of segmented regions}}$$

The result for CIECAM model show:

a) *94% as accuracy for sunny days with 23% false detection.*

b) *Also shows that system works better on sunny days than other days*

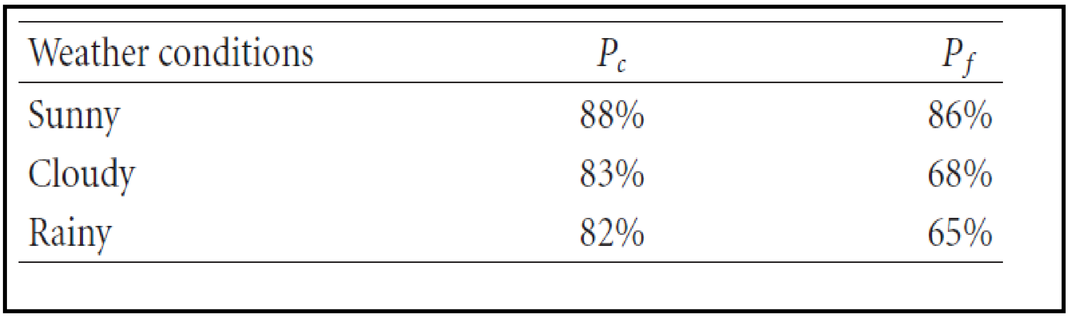
When CIECAM was compared with other models using same segmentation procedure the results were:



Segmentation results by three colour spaces: CIECAM97s, HSI, and CIELUV.

Weather condition	Total signs	Colour space	Results			
			Correct segmentation	False segmentation	P_c	P_f
Sunny	53	*HCJ*(CIECAM97s)	50	15	94%	23%
		HSI	46	19	88%	29%
		HCL(CIELUV)	46	17	88%	27%
Cloudy	32	*HCJ*(CIECAM97s)	29	11	90%	28%
		HSI	24	14	77%	37%
		HCL(CIELUV)	26	12	82%	32%
Rainy	57	*HCJ*(CIECAM97s)	48	18	85%	27%
		HSI	41	26	73%	39%
		HCL(CIELUV)	43	24	76%	36%

Table VI ***Segmentation of images using CIECAM model as compared to HSI and CIELUV***



Weather conditions	P_c	P_f
Sunny	88%	86%
Cloudy	83%	68%
Rainy	82%	65%

Table VII Segmentation of images using RGB

3.3.5 Conclusion

The paper utilised the application of CIE colour appearance model successfully demonstrating an accuracy of 94% for sunny days. Also when compared to other colour models, it outperformed them. However, as per the authors CIECAM model when applied in calculations was more cumbersome and complex. It requires more than 20 steps to perform simple calculations. Also, the researchers had reduced the segmentation processing time to 1.8 seconds and recognition time to 0.19 seconds when images were arranged in a database.

The authors advised that the cause of detection rate being less than 100% was due to small signs in some images.

3.4 2010: Goal Evaluation of Segmentation Algorithms for Traffic Sign Recognition [80]

This paper represented a quantitative comparison of several segmentation methods that had been successfully been used in traffic sign recognition prior to the paper. Here the authors have tested various segmentation methods including their own, for efficiently recognising traffic signs.

The authors advised that the best described algorithm usually consists of following three stages:

- Segmentation
- Detection
- Recognition.

In this paper the authors have applied an evaluation method to a set of images, which consists of an entire recognition system, with the segmentation method being altered to test the efficiency.

Traffic Signs are normally recognised using their colour and shape. However, problems like outdoor lighting, camera settings or the camera itself affect the segmentation step, as this is the first step in high-level detection. In this paper the goal of segmentation was to extract the traffic sign from the background, but there are more than 150 references in literature on colour segmentation itself [80]. Hence, the authors here discuss only the techniques that have been previously used in traffic sign recognition along with some methods that display good characteristics for this task.

To be able to identify the best segmentation method the authors choose the ones which give the best recognition results. The criteria for good recognition results include high recognition rate, low number of lost signs, high speed, and low number of false alarms.

The further discussion includes the traffic recognition system used to measure the performance, algorithms implemented for comparison purpose, the results and final discussion.

Here we will focus mainly on the algorithms implemented, the results and discussion.

3.4.1 The System Overview

The traffic sign recognition system that the authors implemented in an earlier paper [81] has been used to evaluate various segmentation algorithms presented in this paper.

The system implemented consisted of four stages:

1. Segmentation: This stage extracts traffic signs (the objects) from the background and is the main focus of this paper.

2. Detection: Shape detection uses the output masks obtained from the segmentation stage and gives the position and shape of a possible traffic sign. The shapes considered are triangle, circle, rectangle and semicircle and were used as reference shapes to detect the signs from the signatures of each blob obtained by thresholding, colour classification or edge marking. The detection stage is based on work described in [82], where blobs were normalised using minimum inertia axis angle as a reference to reduce projection distortions. Eccentricity was reduced through a method based on second order moments; occlusion was overcome through interpolating the signature after opening the blob. The scaling and rotation problems were reduced by using the absolute values of discrete Fourier transforms (DFT) and the total energy of the signature was normalized.

3. Recognition: After classifying the candidate blobs according to their shapes, the recognition process was initiated. This recognition task was divided into different colours and shapes. Training and testing were carried out according to the colour and shape of each candidate region, thus to reduce the complexity of the problem, each candidate blob was only compared with those signs that had the same colour and shape.

4. Tracking: This stage is used to identify correspondence between recognised signs to give a single output for each traffic sign in the sequence. For each newly detected sign (one that does not have any correspondence to any other sign) a new track process is initiated. At least two detections of the same object are required to consider it as a speed sign. Other information such as position coordinates, size, colour, type category and the mean grey level of the region occupied are evaluated to establish correspondence between two objects in different images.

3.4.2 Segmentation Algorithm

This is the most important section of this paper where all the segmentation algorithms (such as CAD, RGB, RGBNT, HST, Canny, LUT, etc.) are evaluated. These implemented algorithms generate binary masks, enabling objects to be extracted from the background. A mask is obtained for each colour of interest, i.e. red, blue, yellow and white.

In case of edge detection techniques, only one mask is obtained, which is then used with all colours of interest. Detection of white regions using segmentation can be a major problem, as it is an achromatic region which cannot be used with other colours of interest. In addition some colour spaces like HSI are unstable near achromatic colours and cannot be directly used over these achromatic pixels. To improve white segmentation, only pixels considered chromatic are classified into different colours, whereas achromatic pixels over certain threshold are considered white. This idea based on saturation and intensity was used in [83]. Using this idea, here, the authors distinguish between colour-specific segmentation algorithms and chromatic/achromatic decomposition algorithms.

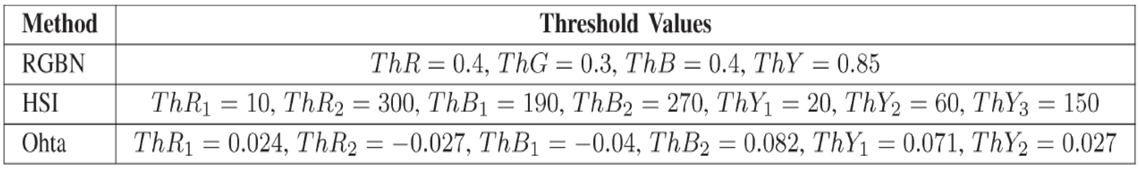
This section is further divided into

A. Colour Space Thresholding Techniques (CST)
B. Chromatic/Achromatic Techniques
C. Edge Detection Techniques
D. Other Techniques

Due to the relation of this paper to our method used (described later) we will only focus on first two techniques mentioned above, while briefly describing the other two techniques used here.

3.4.2.1 Colour Space Thresholding (CST)

One of the extended CST consists of thresholding components in a colour space, variants of this technique have been used for different spaces [84]. An automatic threshold can be obtained using histogram of the image, but it only identifies colour regions instead of colour. Thus the thresholds used in this paper have been selected based on exhaustive search around empirical thresholds. The selected threshold values for various CST techniques have been shown below:



Method	Threshold Values
RGBN	$ThR = 0.4$, $ThG = 0.3$, $ThB = 0.4$, $ThY = 0.85$
HSI	$ThR_1 = 10$, $ThR_2 = 300$, $ThB_1 = 190$, $ThB_2 = 270$, $ThY_1 = 20$, $ThY_2 = 60$, $ThY_3 = 150$
Ohta	$ThR_1 = 0.024$, $ThR_2 = -0.027$, $ThB_1 = -0.04$, $ThB_2 = 0.082$, $ThY_1 = 0.071$, $ThY_2 = 0.027$

Table VIII Threshold values for RGBNT, HST and OST Method.[80]

3.4.2.1.1 RGB Normalised Thresholding

RGB space is one of the basic colour spaces. If a simple segmentation is done, RGB can be used as it is, i.e. without any transformation applied to the RGB components. Finding correct thresholds using empirical methods is difficult in this space due to the effect of illumination changes on colour. Normalisation of RGB with respect to R+G+B as in [85, 86] has been here considered as a solution. Given that normalised components (r, g, b) are being used, illumination changes have less effect on colour. Also, if sum of new components is r+g+b=1, then only two components are required to carry out the classification, as other components can be obtained directly from these two. However, with low illumination, the transformation is unstable and at near-zero values, noise is amplified. Masks are obtained using the following expressions:

$$\text{Red}(i,j) = \begin{cases} \text{True}, & \text{if } r(i,j) \geq ThR \\ & \text{and } g(i,j) \leq ThG \\ \text{False}, & \text{otherwise} \end{cases}$$

$$\text{Blue}(i,j) = \begin{cases} \text{True}, & \text{if } b(i,j) \geq ThB \\ \text{False}, & \text{otherwise} \end{cases}$$

$$\text{Yellow}(i,j) = \begin{cases} \text{True}, & \text{if } (r(i,j) + g(i,j)) \geq ThY \\ \text{False}, & \text{otherwise.} \end{cases}$$

Figure 31 Mask for each colour in RGB colour space [80]

3.4.2.1.2 Hue and Saturation Thresholding (HST)

Here HSI colour space has been used for thresholding, with only two components, Hue and Saturation, being used. The HSI components are obtained from RGB [83],with hue H being in the interval [0,360] and saturation S within [0,255]. It has been observed that, hue is undefined where saturation is null at greyscale R=G=B and saturation is undefined with intensity at null. Output mask used is as follows:

$$\text{Red}(i,j) = \begin{cases} \text{True}, & \text{if } H(i,j) \le ThR_1 \\ & \text{or } H(i,j) \ge ThR_2 \\ \text{False}, & \text{otherwise} \end{cases}$$

$$\text{Blue}(i,j) = \begin{cases} \text{True}, & \text{if } H(i,j) \ge ThB_1 \\ & \text{and } H(i,j) \le ThB_2 \\ \text{False}, & \text{otherwise} \end{cases}$$

$$\text{Yellow}(i,j) = \begin{cases} \text{True}, & \text{if } H(i,j) \ge ThY_1 \\ & \text{and } H(i,j) \le ThY_2 \\ & \text{and } S(i,j) \ge ThY_3 \\ \text{False}, & \text{otherwise.} \end{cases}$$

Figure 32 Output mask of HSI space for HST [80]

The figure above shows threshold values used, saturation was only used for yellow.

This method is simple and is almost immune to illumination changes due to the use of hue, but the main drawbacks include the instability of hue and increase in processing time due to RGB-HSI conversion.

3.4.2.1.3 Hue and Saturation Colour Enhancement Thresholding

A different thresholding method for hue and saturation was proposed in [87] and has been applied here. Here the distribution of hue and saturation in red and blue hand-segmented signs was studied to identify values associated with each colour. An LUT was used as a soft threshold to prevent the problems of rigid threshold, by applying different values using linear functions to hue and saturation. Both the hue and saturation were normalised (check normalisation in definitions) in the interval [0,255]. Initially in [87] yellow was not included, but the authors incorporated his colour by using enhancement techniques for blue and applying different parameters as shown below:

The LUTS used are shown below:

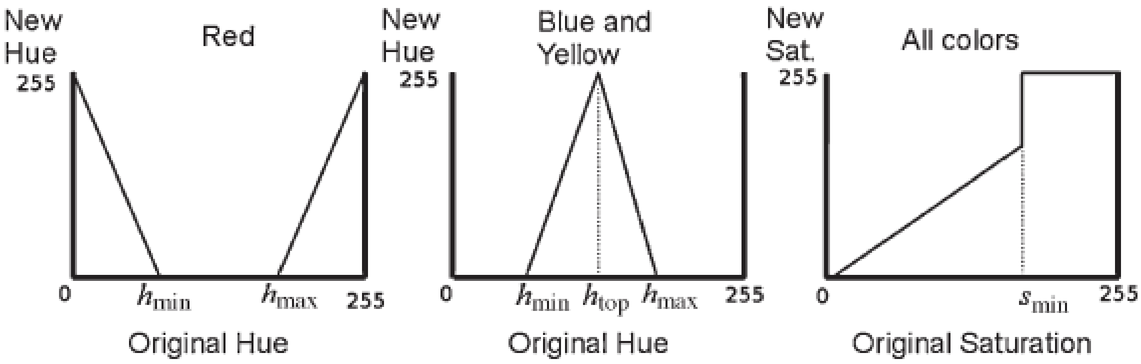


Figure 33 Colour LUTs for the HSET method. Hue and Saturation normalized in the interval [0,255].

Summary of the method applied:

I. Obtain hue and saturation from RGB

II. Transform hue and saturation with two LUTs

III. Normalise the product of the transformed hue and saturation

IV. Threshold the normalised product

Once the new hue and saturation were obtained, the product of these values could be calculated and the result was the threshold without normalisation. The values used were modification from the initial method as the product was not normalised and the aim was to obtain the best empirical results.

3.4.2.1.4 Ohta Space Thresholding

In this method, Ohta space was used for thresholding as it displayed some desired characteristics, including simplicity and low computational cost. Based on extensive experiments, the colour features that were derived from RGB for colour segmentation were:

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 1 & 0 & -1 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

The normalised components used were:

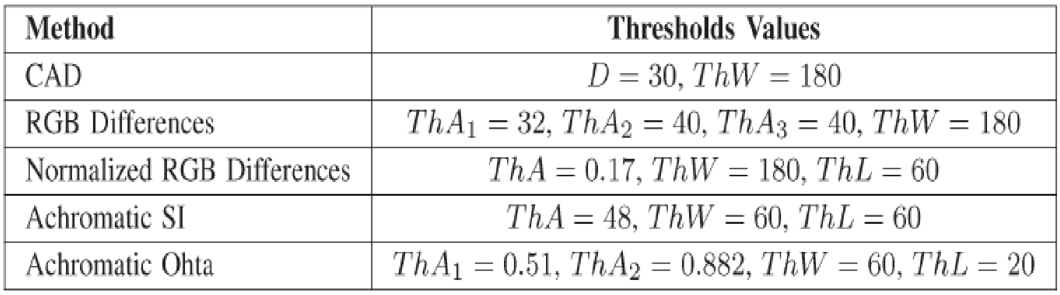
$$P_1 = \frac{1}{\sqrt{2}} \frac{R - B}{R + G + B} = \frac{1}{3\sqrt{2}} \frac{I_2}{I_1}$$

$$P_2 = \frac{1}{\sqrt{6}} \frac{2G - R - B}{R + G + B} = \frac{2}{3\sqrt{6}} \frac{I_3}{I_1}$$

'I' is the component related to illumination and 'P' is the normalised component.

3.4.2.2 Chromatic /Achromatic Decomposition

Chromatic/Achromatic decomposition tries to find the image with grey pixels or no colour. Usually, the grey pixels occur when R=G=B or this is approximately true. The authors here discuss the methods to extract grey pixels, and the brighter pixels are treated as white ones. In this section, authors just define the various conditions selected for choosing achromatic pixel and a white pixel according to threshold. Thresholds selected for different methods have been shown below:



Method	Thresholds Values
CAD	$D = 30$, $ThW = 180$
RGB Differences	$ThA_1 = 32$, $ThA_2 = 40$, $ThA_3 = 40$, $ThW = 180$
Normalized RGB Differences	$ThA = 0.17$, $ThW = 180$, $ThL = 60$
Achromatic SI	$ThA = 48$, $ThW = 60$, $ThL = 60$
Achromatic Ohta	$ThA_1 = 0.51$, $ThA_2 = 0.882$, $ThW = 60$, $ThL = 20$

Table IX Threshold values for achromatic methods [80].

Following sub sections will show the formulae for different methods.

3.4.2.2.1 Chromatic/Achromatic Index

A chromatic/achromatic index is defined as

$$\mathrm{CAD}(R,G,B) = \frac{(|R-G| + |G-B| + |B-R|)}{3D}$$

Where R, G, and B represent colour components and D is the degree of the extraction of an achromatic colour. A pixel is considered achromatic when:

$$\mathrm{Achr}(i,j) = \begin{cases} \text{True}, & \text{if } \mathrm{CAD}(i,j) \leq 1 \\ \text{False}, & \text{otherwise} \end{cases}$$

and white when

$$\mathrm{White}(i,j) = \begin{cases} \text{True}, & \text{if } \mathrm{Achr}(i,j) = \text{True} \\ & \text{and } (R+G+B) \geq ThW \\ \text{False}, & \text{otherwise.} \end{cases}$$

3.4.2.2.2 RGB Differences

According to the authors, use of threshold to measure difference between every pair of components is more realistic, thus they have marked the colours as being achromatic when the three differences between components are below a fixed threshold. As shown below, a pixel is considered achromatic when

$$\text{Achr}(i,j) = \begin{cases} \text{True}, & |R(i,j) - G(i,j)| \leq ThA_1 \text{ and} \\ & |G(i,j) - B(i,j)| \leq ThA_2 \text{ and} \\ & |B(i,j) - R(i,j)| \leq ThA_3 \\ \text{False}, & \text{otherwise} \end{cases}$$

and white when

$$\text{White}(i,j) = \begin{cases} \text{True}, & \text{if Achr}(i,j) = \text{True} \\ & \text{and } (R+G+B) \geq ThW \\ \text{False}, & \text{otherwise.} \end{cases}$$

3.4.2.2.3 Normalised RGB Differences

In the Normalised RGB space, achromatic pixels are found in similar way as shown above, except that only two differences are required instead of three, since the third component is obtained from the other two as shown previously (section 3.3.7.1.1).

$$\text{Achr}(i,j) = \begin{cases} \text{True}, & \text{if } |r(i,j) - g(i,j)| \leq ThA \\ & \text{and } |r(i,j) - b(i,j)| \leq ThA \\ \text{False}, & \text{otherwise} \end{cases}$$

and white is obtained with

$$\text{White}(i,j) = \begin{cases} \text{True}, & \text{if Achr}(i,j) = \text{True} \\ & \text{and } (R+G+B) \geq ThW \\ \text{False}, & \text{otherwise.} \end{cases}$$

3.4.2.3 Edge-Detection Techniques

The edge detection is an extended segmentation technique that is used to locate the different objects within an image. The edge points are marked as ‘no object’ so that the points inside a closed edge are automatically marked as “object”. This method does not require colour information and hence problems of colour spaces can be avoided. Problems like hue being affected with illumination changes, larger computational cost for converting RGB to HSI and changes in lighting conditions can be overcome by these edge detection techniques. This is a shape based method and more robust as it is not affected by light changes. One common problem with these methods is that they produce quite a lot of candidate objects.

The methods used were

3.4.2.3.1 Greyscale Edge Removal (GER)

This method comprises the classical two-step second order derivative (Laplacian) method:

Step I Smooth the image

Step II Apply laplacian filter that performs second derivative over the image to extract the edges.

This produces a result image L (i,j) and the edge is obtained by:

$$O(i,j) = \begin{cases} \text{Edge,} & L(i,j) \geq T \\ \text{No-edge,} & L(i,j) < T \end{cases}$$

3.4.2.3.2 Canny Edge Removal (Canny)

Canny edge detection is used and recognised as the “standard method” for edge detection [88]. It uses linear filtering with a Gaussian kernel to smooth noise and then computes the edge strength and direction for each pixel in the smoothed image. Finally the edges are marked after differentiation and non-maximal suppression and thresholding. This gives minimal isolated points and hence gives connected shapes.

3.4.2.3.3 Colour Edge Removal

Colour edge removal method uses RGB colour space to provide edge detection. This method measures the distance between one pixel and its 3x3 neighbours in the RGB colour space.

The edge detector is applied over a colour image using the following equation:

$$D_{ij} = \sum_{k=1}^{8} (R_{ij} - R_{ijk})^2 + (G_{ij} - G_{ijk})^2 + (B_{ij} - B_{ijk})^2$$

Where R_{ij}, G_{ij} and B_{ij} are the red, green and blue values of a pixel *i,j*.R_{ijk}, G_{ijk} nad B_{ijk} are the k^{th} value of a neighbour, respectively. Pixels with values below a threshold are considered as belonging to the foreground based on the value of D_{ij} for each pixel, whereas those above the threshold are considered as belonging to the edges separating the objects from the foreground.

Authors also describe another two methods namely

1. SVM Colour segmentation (SVMC) and
2. Speed Enhancement using a LUT

Description of these two methods is out of scope of this research.

3.4.3 Experiments

This section mainly describes the different parameters that the authors used to evaluate signs for analysis of colour segmentation.

i. **Number of Signs recognised:** Each individual sign was counted the number of times it was correctly recognised. The sign was given a normalised version related to the maximum number of times that the sign could be detected; hence the maximum was ‘1’.

ii. **Global rate of correct recognition:** Sum of all the correctly recognised signs was related to the number of signs that could be recognised. A value of 100 indicated that all possible signs in all sequences were correctly recognised.

iii. **Number of lost signs:** This referred to the number of signs that were not recognised in any way.

iv. **Number of maximum:** this parameter gave the number of times that a method achieved the maximum score.

v. **False recognition rate:** this represented the percentage of signs detected falsely

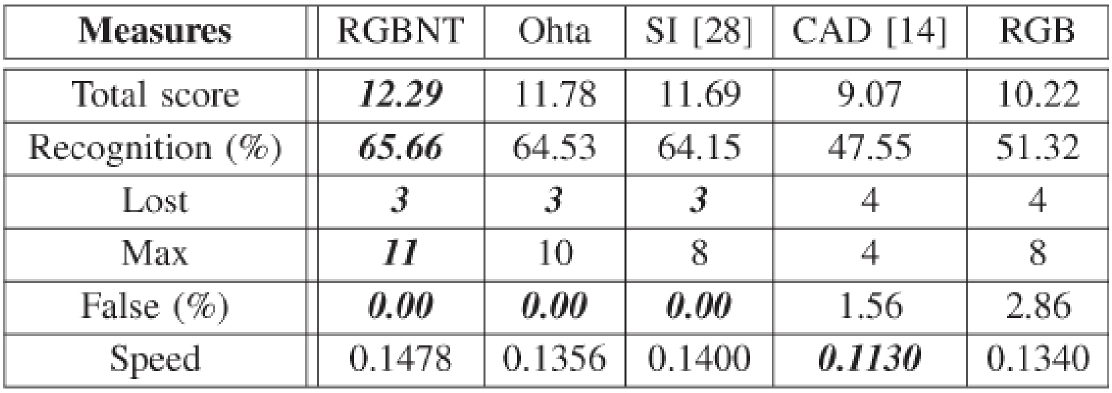
vi. **Speed:** It was the measure of execution time per frame.

3.4.4 Results

The findings of different methods used for colour segmentation were noted as follows

I. CST Methods: From the results obtained, it was clear that the best methods were the CST methods for image segmentation and the edge detection methods were the worst. Amongst CST methods, the RGB normalised method obtained the best score for total score and recognition percentage. The best result for lost and the number of maximum were obtained by LUT SVM method. Also in the execution speed, CST methods were the best with RGB normalised showing good speeds.

II. Achromatic Decomposition Methods: The result of achromatic decomposition was only based on white signs segmentation only. The method that got the best score for total score, recognition percentage and number of lost signs was the RGB normalised method. As well the RGB normalised had no false positives, as shown below:



Measures	RGBNT	Ohta	SI [28]	CAD [14]	RGB
Total score	***12.29***	11.78	11.69	9.07	10.22
Recognition (%)	***65.66***	64.53	64.15	47.55	51.32
Lost	***3***	***3***	***3***	4	4
Max	***11***	10	8	4	8
False (%)	***0.00***	***0.00***	***0.00***	1.56	2.86
Speed	0.1478	0.1356	0.1400	***0.1130***	0.1340

Table X Analysis of the results for various segmentation methods [80]

Other results are out of scope of this research, hence not discussed.

3.4.5 Conclusion

The author's analysis of data has led to six conclusions:

i. The recognition percentage results for the best method are 69.49% for the test set and 78.29% for the validation sets. These results were obtained by taking into account all the times that a traffic sign could be recognised in a sequence.
ii. For test and validation sequences RGB Normalised method performed the best.
iii. Edge detection methods may be used as a complement to other colour – segmentation methods.
iv. No method performed well in all the chosen criteria.
v. The use of LUT method improves speed
vi. Normalisation, as in the RGB normalised method, improved performance and represents a low cost operation.

Although, no method was best for all the cases, but the authors recommend colour space thresholding methods incorporating illumination normalisation as a good choice. In addition, they also recommended using of LUT to increase speed at the cost of loss of information. Achromatic decomposition has been advised as a good choice for treating achromatic pixels as identifying white colour in signs has been improved by this decomposition.

3.5 Review of current methods

After analysing the above methods recommended for detection of traffic signs, the following conclusions were made:

1) Hue and Saturation are invariant to illumination.

2) Saturation and Value can be used for defining chromatic subspace.

3) Hue is unstable when pixel is not in the chromatic subspace.

4) RGB Normalisation can reduce complexity of a program and hence increase speed.

5) CIECAM has been proved as one of the best models to use for colour segmentation but it is slow for processing.

6) RGB Normalisation has been recommended as one of the most effective models and can be used for achromatic decomposition.

The observations mentioned above, were used to design a robust model for colour segmentation in this research and hence they became the underlying principle of the method proposed for detection of traffic signs in dark and gloomy images.

The next chapter explains the software and the sample program used to conduct the research and briefly explains the sample program which formed the basis of the final software.

4 Sample Program: Open CV, Emgu CV, and API

After going through current work being conducted, in the field of computer vision for detection of speed signs, this chapter will describe the computer vision tools used for successfully designing the software for detection of speed signs.

The chapter has been divided into three sections, section one describes the Open CV, section two the Emgu CV and the third section describes the sample program used to develop the software for detection of speed signs.

4.1 Open CV

Open Source Computer Vision (Open CV) is a library of programming functions mainly useful for real time computer vision applications [89]. It has a BSD license (free for commercial or research use), hence it was chosen for the development of our software. OpenCV was originally written in C but now has been developed so that it has a full C++ interface and any new development that is carried out is developed in C++.

The OpenCV library has more than 2000 optimized algorithms *(see figure below)*. It is used around the world for different purposes, including but not limited to interactive art, to mine inspection, stitching maps on the web, on through advanced robotics.

Main advantages of OpenCV except for being open source and freely available that have been outlined in [90], are:

- It has been optimized and is intended for real-time applications.
- It is independent of operating system, hardware or any window manager.
- It has got generic image and/or video loading, saving, and acquisition functions present in it.
- Both low and high level API is present for OpenCV.
- It also provides interface to Intel's Integrated Performance Primitives (IPP) library, with processor specific optimization (Intel processors).

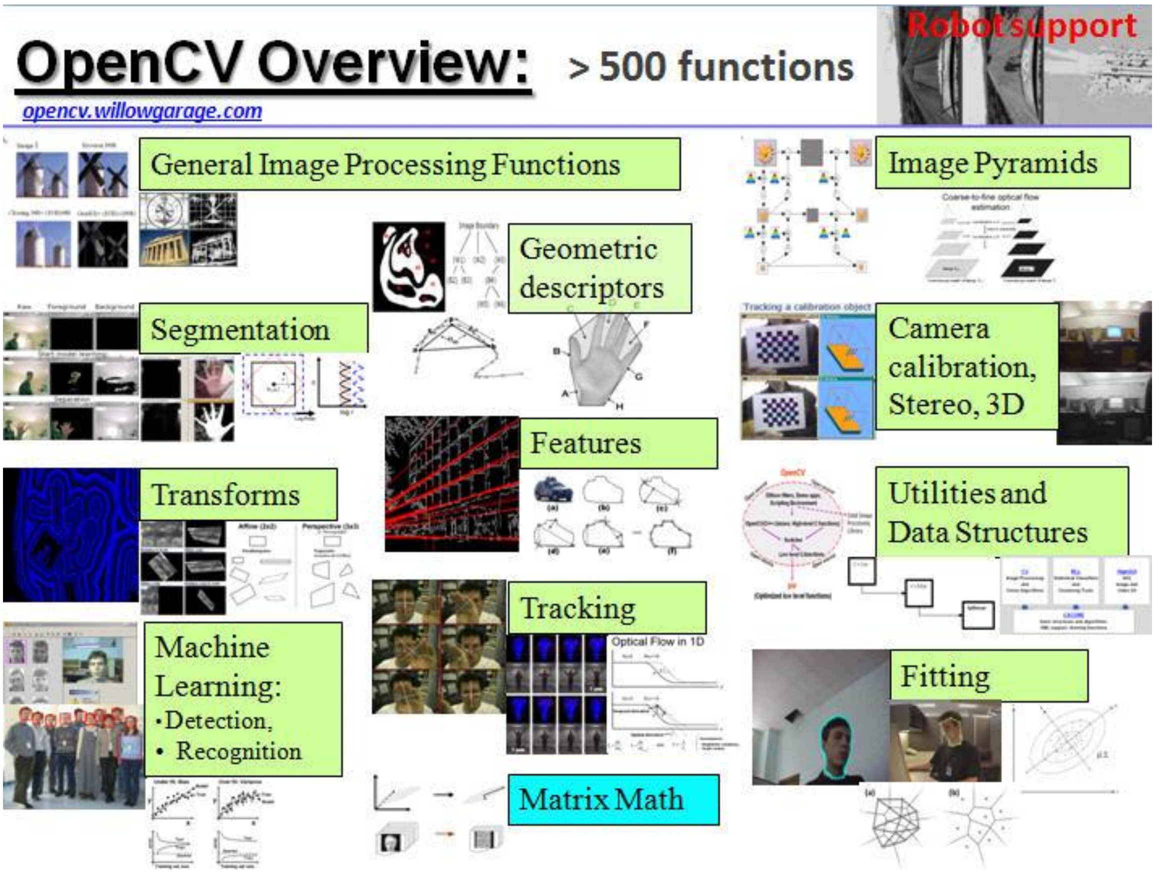
4.1.1 Features of OpenCV:

Basic function of image data manipulation like allocation, release, copying, setting and conversion are all implemented in OpenCV. It also provides Input / Output functions for image and video files, in addition camera based input and image/video file output is also available for advance functions. OpenCV makes it simpler for end user to implement matrix and vector manipulation and also provides linear algebra routines like products, solvers and Eigen values.

Using OpenCV for development also means that implementation of various dynamic data structures like lists, queues, sets, trees and graphs is done in the background, thus lets a programmer focus on the main tasks. Basic image processing functions like filtering, edge detection, corner detection, sampling and interpolation, colour conversion, morphological operations, histograms, image pyramids are all provided in OpenCV and can be modified if required due to its open nature. Structural analysis like connected components, contour processing, distance transform, template matching, Hough transform, polygonal approximation, line fitting, ellipse fitting etc. are also present in the library. Other advanced features provided in the library are Camera calibration (finding and tracking calibration patterns, calibration), Motion analysis (optical flow, motion segmentation, tracking), Object recognition (Eigen-methods, HMM), Basic GUI (display image/video, keyboard and mouse handling, scroll-bars) and Image labelling (line, conic, polygon, text drawing).

The main OpenCV modules that act as building block of any program are:

- *Cv* - Main OpenCV functions.
- *Cvaux* - Auxiliary (experimental) OpenCV functions.
- *Cxcore* - Data structures and linear algebra support.
- *Highg*ui- GUI functions.



OpenCV Overview: > 500 functions

opencv.willowgarage.com

Robot support

General Image Processing Functions

Image Pyramids

Geometric descriptors

Segmentation

Camera calibration, Stereo, 3D

Features

Transforms

Utilities and Data Structures

Tracking

Machine Learning:
- Detection,
- Recognition

Fitting

Matrix Math

Tracking a calibration object
Optical Flow in 1D
Coarse-to-fine optical flow estimation
Affine (2x2)
Perspective (3x3)
OpenCV

Figure 34 Image showing an overview of OpenCV library [89]

To be able to use OpenCV in .Net platform, a .Net wrapper called Emgu CV was used. It has been discussed in the next section.

4.2 Emgu CV

Emgu CV is a cross platform .Net wrapper to the Intel OpenCV image processing library. It has been written in C# and it can be compiled in Mono. The main benefit is that it is able to run on any platform Mono supports, including Linux, Solaris and Mac OS X.

4.2.1 Architecture Overview

Emgu CV has two layers of wrapper as shown below:

- The basic layer (layer 1) contains function, structure and enumeration mappings which directly reflect those in OpenCV.
- The second layer (layer 2) contains classes that mix in advantages from the .NET world.

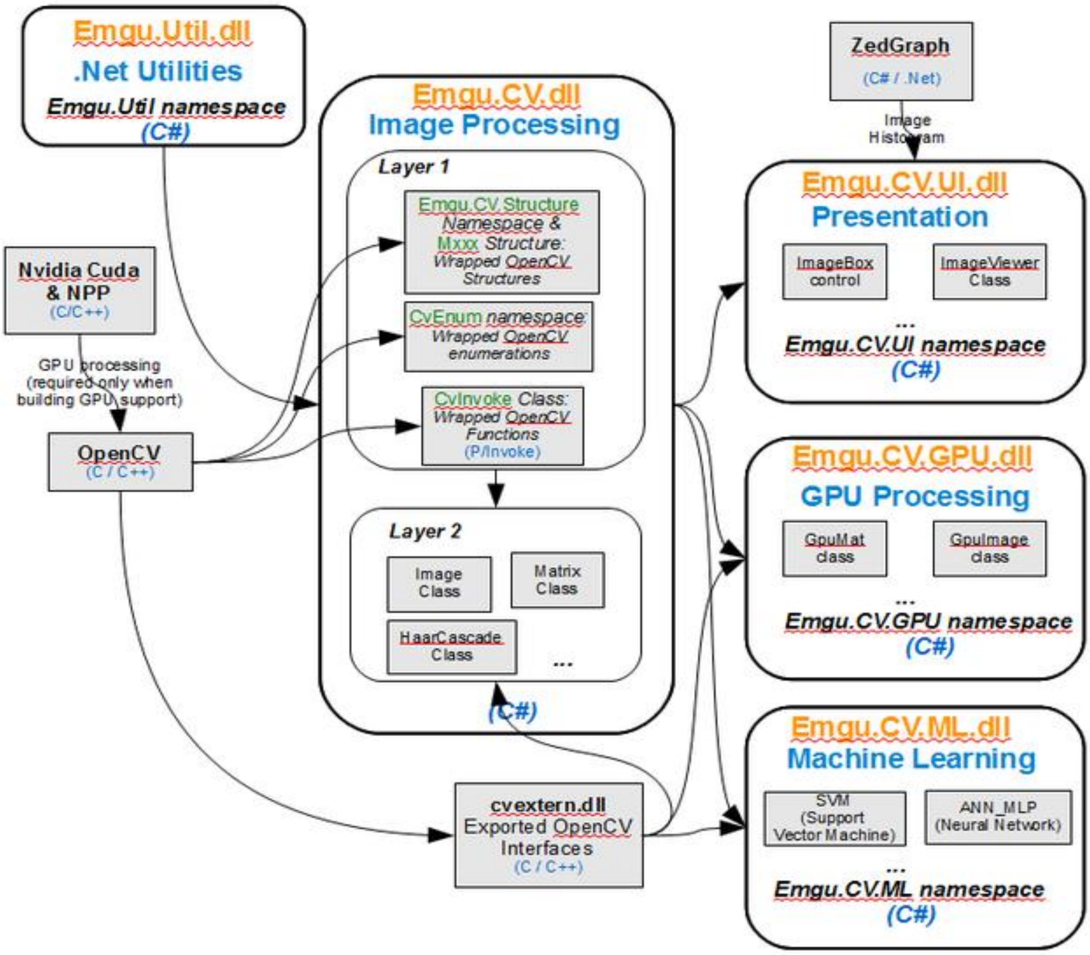


Figure 35 Overview of Emgu CV wrapper [91]

4.2.2 Advantages of Emgu CV

- Cross Platform
- Cross Language and comes with example code
- Image class with Generic Colour and Depth
- Automatic garbage collection
- XML Serializable Image
- XML Documentation and intellisense support
- The choice to either use the Image class or directly invoke functions from OpenCV
- Generic operations on image pixels

4.3 The Sample Program

This section describes the sample program provided with the Emgu CV libraries for detection of stop signs, called Traffic Sign Recognition. The author has used it as a base program to create the main software for detection of NZ based road sign as a part of the research. The sample program's important image processing functions are explained below in sub sections. The final section 4.4 analyses the program for the suitability of detecting speed signs. The sub sections below describe the most important functions/ techniques required for image processing of speed signs that have been used as a base for developing the final software. Flow chart for the program is shown below:

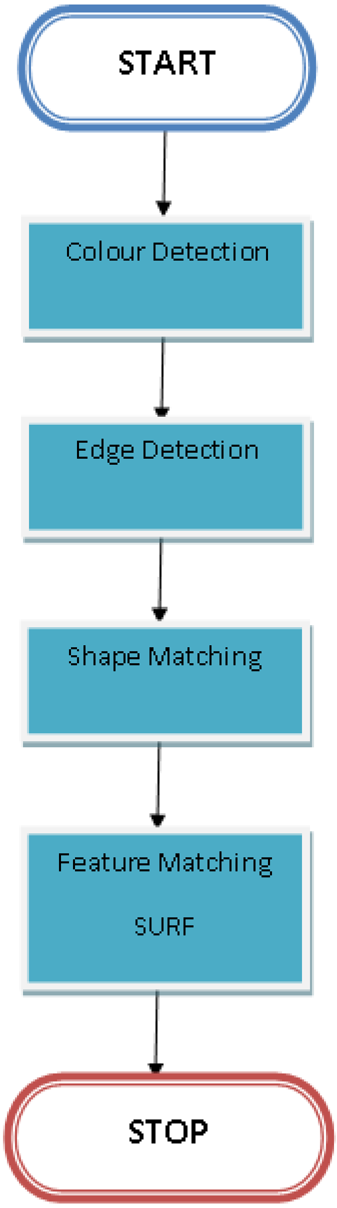


Figure 36 Flow Chart of the Program Implemented

4.3.1 Colour Segmentation

As mentioned this sample program is a basic program for detecting just the stop signs, hence colour segmentation technique used is a simple splitting technique based on HSV colour model.

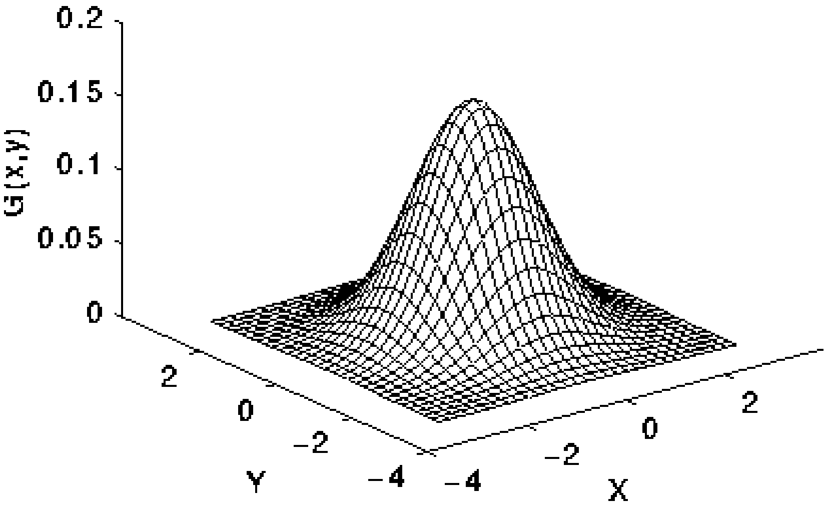
4.3.1.1 Gaussian Smoothing

The image is first smoothed using Gaussian blur, also known as Gaussian smoothing. Gaussian blur is done to remove detail, and reduce the image noise to enhance the image signal. Gaussian blur is a widely used effect in graphics software and the result of the blur is usually similar to viewing of image through a translucent screen. Gaussian blur reduces an image's high frequency components and thus is a low pass filter. The equation of a Gaussian function in two dimensions is:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Where ***x*** *is the distance from the origin in the horizontal axis,* ***y*** *is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution*.

The Gaussian kernel is represented as a normal distribution curve (bell shape), shown below:



std deviation =1 and mean =0

Figure 37 Gaussian Distribution curve in 2D[92]

Shown below is the effect of Gaussian smoothing in the program:

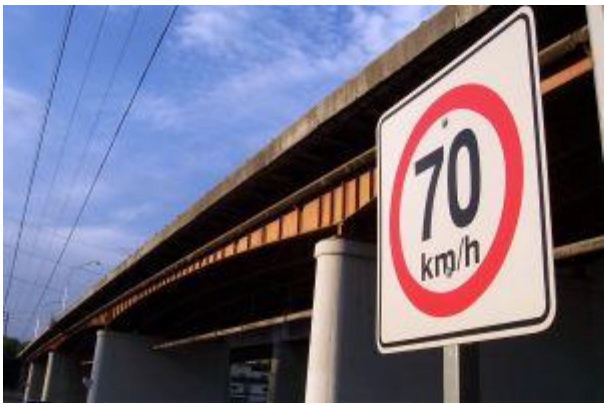


Figure 38 Original image

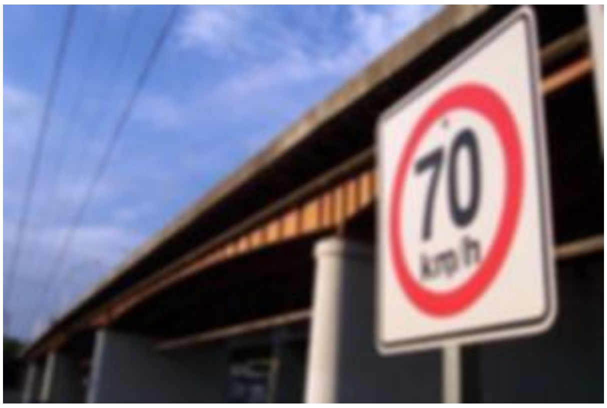


Figure 39 Gaussian Smoothing

The smoothed image is then passed to the colour detection function called ‘GetRedPixelMask’. In this function the smoothed RGB image is first converted to HSV colour space, using inbuilt convert function of Emgu CV, and then it is split into an array of hue, saturation and value property grey images named channels.

```
private static Image<Gray, Byte> GetRedPixelMask(Image<Bgr, byte> 
image)
        {
            using (Image<Hsv, Byte> hsv = image.Convert<Hsv, Byte>())
            {
                Image<Gray, Byte>[] channels = hsv.Split();
```

Code 1 ‘GetRedPixelMask’ Function

Then the masking is performed, to find the red dominant region in the entire image. This is done by performing an ‘and’ function on the grey images generated from the initial ‘channels’ images. The channels[0] is masked for red hue and is anded with channels[1], which contains saturation with white pixels filtered out. The channels[2] containing the value property is not used. The code is shown below:

```
//channels[0] is the mask for hue less than 20 or larger than 160
```

```
            CvInvoke.cvInRangeS(channels[0], new MCvScalar(20), new
MCvScalar(160), channels[0]);
            channels[0]._Not();

            //channels[1] is the mask for satuation of at least 10,
this is mainly used to filter out white pixels
            channels[1]._ThresholdBinary(new Gray(10), new
Gray(255.0));

            CvInvoke.cvAnd(channels[0], channels[1], channels[0],
IntPtr.Zero);

            channels[1].Dispose();
            channels[2].Dispose();
            return channels[0];
         }

      }
```

Code 2 ‘And’ of red hue and saturation grey images

The grey image with only red dominant region (in open CV, hue ranges from 0 to 180 with red being 0 and 180, green being 30 and blue being 120.) is returned back for further image processing using the edge detection techniques, explained next.

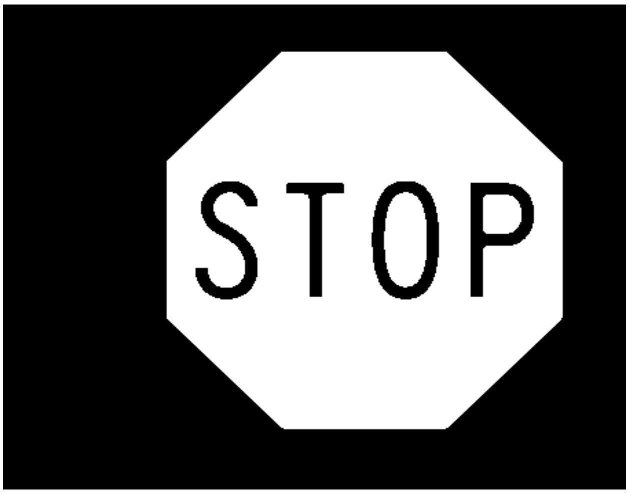


Figure 40 Image returned after ‘GetRedPixelMask’ function

4.3.2 Edge Detection

Edges characterise boundaries and are therefore a problem of fundamental importance in image processing. Edge detection significantly reduces the amount of data and filters out useless information while preserving important structural properties of an image [93].

Once the grey image is generated with the red dominant regions being white, it is then processed by two mathematical morphological functions, dilate and erode. This helps in identifying the shapes within the image, as mathematical morphology deals with the analysis and processing of geometrical structures.

4.3.2.1 Dilate

The image is first processed with the 'Dilate' function. This function as the name suggests, introduces dilation in the image using a 3x3 rectangular structuring element in our program, for probing and expanding the shapes contained in the image. Dilation in Emgu CV can be applied several times. Dilation is one of the basic operations in mathematical morphology and has been shown below using a circular structural element superimposed on a binary image consisting of a square:

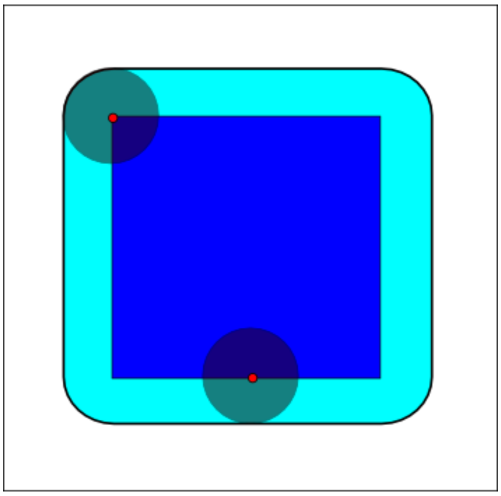
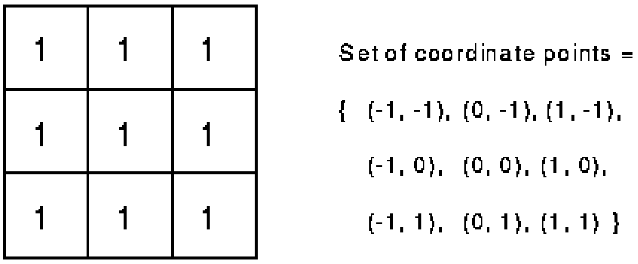


Figure 41 Dilation of the dark blue square by a disk, resulting in the light blue square [94]

4.3.2.2 Erode

Once the image is dilated, it is passed on for erosion, again with a 3x3 rectangular structuring element. The effect of erosion applied on an image is that it erodes away the boundaries of regions of foreground pixels, thus shrinking the foreground objects in size [95]. Here a 3x3 rectangular structuring element is superimposed on the binary image being eroded and checked for a foreground pixel, value 255, or a background pixel, value 0. If any of the corresponding pixels in the binary image are background pixels, then the corresponding input pixel in the structuring element is also set to background pixel, value 0. If a corresponding pixel in the base image is a foreground pixel, then the input pixel is also left as it is. This helps in reducing the foreground pixels and hence shrinks the image [95]. The effect of erosion using a 3x3 rectangular structuring element is shown below:



1	1	1
1	1	1
1	1	1

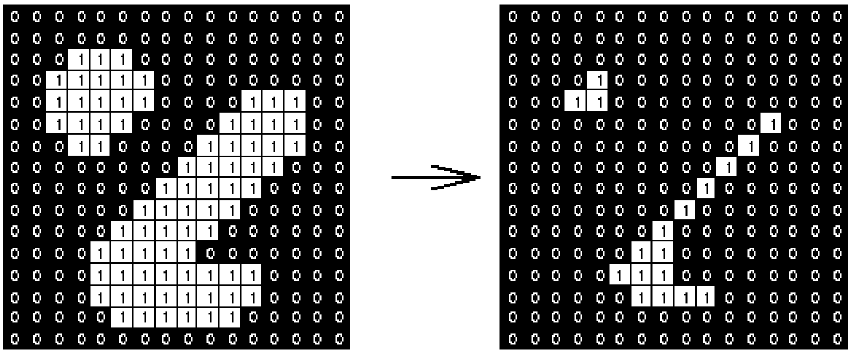
Set of coordinate points =

{ (-1, -1), (0, -1), (1, -1),

(-1, 0), (0, 0), (1, 0),

(-1, 1), (0, 1), (1, 1) }

The 3x3 structuring element



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	1	1	1	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Effect of erosion using the structuring element, shown above [95]

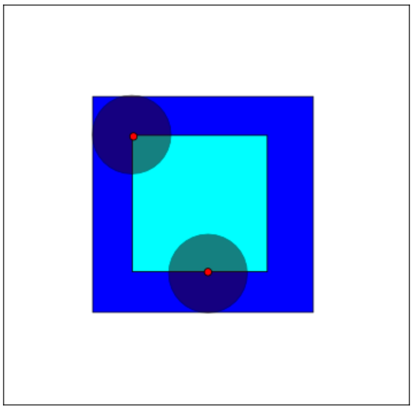


Figure 42 Erosion of the dark blue square by a disk, resulting in the light blue square [94]

Shown above is the effect of a circular structuring element on a square binary image.

4.3.2.3 Canny Edge Detector

After performing basic image processing functions of smoothing, dilating and eroding, the image is passed through an edge finding process called the canny edge detector.

The Canny Edge detection operator was developed by John F. Canny in 1986 , with the main intention of enhancing the already existing edge detectors [96]. Canny operator uses multistage algorithm to detect a wide range of edges in an image. It first smoothes the image to eliminate noise and then works on image gradient to emphasize regions with high spatial derivates. Different stages of canny algorithm are:

i. Noise Reduction: As Gaussian smoothing can be done using a simple mask, it is usually used for dealing with canny algorithm.

ii. Finding the intensity gradient: Canny algorithm uses mainly four filters to find the direction of an edge in a blurred image. The Canny edge detector in Open CV uses Sobel edge detection operator to find the first derivative in the horizontal direction (Gy) and the vertical directions (Gx), as shown:

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2} \quad ; \quad \mathbf{\Theta} = \operatorname{arctan}\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right).$$

*Where **Θ** is the edge direction angle*

iii. Non-maximum suppression: It helps in determining if the gradient magnitude assumes a local maximum in the gradient direction or not. From this stage a set of edge points in the form of a binary image is obtained as shown below:

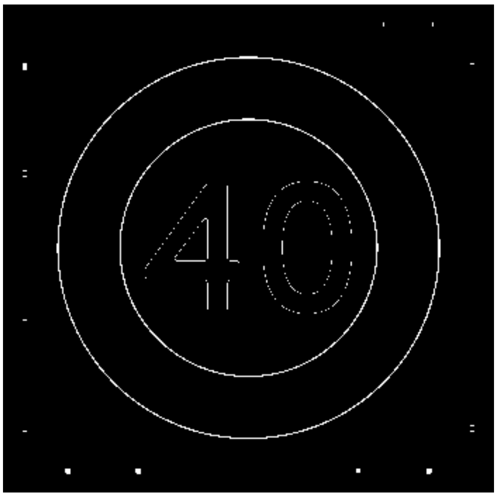


Figure 43 Image showing Canny edges in a speed sign

The image is segmented using the canny edge detection and then the inbuilt ‘FindContours’ function is used to get the contours lines in the canny image. The canny function is shown below, as applied in the program:

```
public void DetectStopSign(Image<Bgr, byte> img, List<Image<Gray,
Byte>> stopSignList, List<Rectangle> boxList)
      {
         Image<Bgr, Byte> smoothImg = img.SmoothGaussian(5, 5, 1.5,
1.5);
         Image<Gray, Byte> smoothedRedMask =
GetRedPixelMask(smoothImg);
         smoothedRedMask._Dilate(1);
         smoothedRedMask._Erode(1);
using (Image<Gray, Byte> canny =
smoothedRedMask.Erode(3).Dilate(3).Canny(new Gray(100), new Gray(50)))
         using (MemStorage stor = new MemStorage())
         {
            Contour<Point> contours = canny.FindContours(

Emgu.CV.CvEnum.CHAIN_APPROX_METHOD.CV_CHAIN_APPROX_SIMPLE,
               Emgu.CV.CvEnum.RETR_TYPE.CV_RETR_TREE,
               stor);
            FindStopSign(img, stopSignList, boxList, contours);
         }

      }
```

Code 3 Edge detection in the program

The contours are stored as a collection of points in a memory location ‘stor’ as shown above. These contours are then passed on for shape recognition to detect the stop sign.

4.3.3 Shape detection

In the sample program, the basic image’s (stop sign) shape which is an octagon is first stored in memory as a contour and then it is compared to the subject image’s contours to find the matching shape. The implementation is shown below:

```
_octagonStorage = new MemStorage();
         _octagon = new Contour<Point>(_octagonStorage);
         _octagon.PushMulti(new Point[] {
            new Point(1, 0),
            new Point(2, 0),
            new Point(3, 1),
            new Point(3, 2),
            new Point(2, 3),
            new Point(1, 3),
            new Point(0, 2),
            new Point(0, 1)},
            Emgu.CV.CvEnum.BACK_OR_FRONT.FRONT);
```

Code 4 Matching of contours using the cvMatchShapes function

Shape detection has only one main function called ‘cvMatchshapes’. This is an Open CV function that uses Hu Moments to match two given contours.

4.3.3.1 Hu Moment

An image moment is a certain particular weighted average (moment) of the image pixel’s intensities or a function of such moments. It describes the image content (or distribution) with respect to its axes and a raw moment M_{ij} of a scalar (greyscale) image with pixel intensities $I(x,y)$ is usually given by the equation:

$$M_{ij} = \sum_x \sum_y x^i y^j I(x,y)$$

Hu moment is a rotation invariant moment which is also invariant under translation, reflection and changes in scale. It was proposed by Ming-Kuei Hu in 1962 [97]. Hu described 6 basic moments and a seventh moment called skew moment. The seventh moment's sign changes with a change in reflection but is skew invariant, which enables it to distinguish mirror images of otherwise identical images. The Hu moment equations are given below:

$$
\begin{aligned}
I_1 &= \eta_{20} + \eta_{02} \\
I_2 &= (\eta_{20} - \eta_{02})^2 + (2\eta_{11})^2 \\
I_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
I_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
I_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\
&\quad (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
I_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
I_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - \\
&\quad (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2].
\end{aligned}
$$

Figure 44 Hu Moments [98]

The 'cvMatchshapes' returns a value depicting the degree of similarity between two contours. The closer the value is to '0', better is the match between the two contours. Implementation of shape detection in the program:

```
for (; contours != null; contours = contours.HNext)
          {
             contours.ApproxPoly(contours.Perimeter * 0.02, 0,
contours.Storage);
             if (contours.Area > 200)
             {
                double ratio = CvInvoke.cvMatchShapes(_octagon,
contours, Emgu.CV.CvEnum.CONTOURS_MATCH_TYPE.CV_CONTOURS_MATCH_I3, 0);
                if (((0.10 < ratio2)) || (!isEllipse)) //not a good
match of contour shape ||
             { RImage = DetectFindStopSign(img, child, stopSignList,
             boxList, probableSignboxList,listrequired
             }
```

Code 5 Matching of contours using the cvMatchShapes function

Shown above is the code for finding the degree of similarity 'ratio' for two contour curves, that are first approximated using the 'ApproxPoly' and then their shapes are matched using the ‘cvMatchshapes’ function. Once the correct ratio is found i.e. ratio < 0.1, then the contours are sent for feature matching using the SURF matching function (out of scope for this research).

4.3.4 Analysis of the Sample program (Flaws)

As mentioned earlier, this sample program was designed for detecting stop signs with simple images only. When this was tested on some random speed sign images, using circular shape as the base shape, the program failed and did not return the desired results. After careful observation and debugging of the program following problems were discovered:

I. Simple colour detection: When the program was tested with actual images provided by Geosmart, the failure rate was much higher than the success rate. The failure rate considerably increased when the images being tested were dark or had some kind of variation in illumination levels. The sample program was failing at the initial colour segmentation function where it was unable to detect the red circles correctly. This is shown in figure below.

II. Simple shape detection: The program’s shape detection function was not working accurately. The function was using a basic circle shape stored in memory and Hu moments to match it with the subject shape. The result was that it was detecting shapes similar to a circle, and detecting them as speed signs, e.g. an octagon and a rectangle were being detected as a circle.

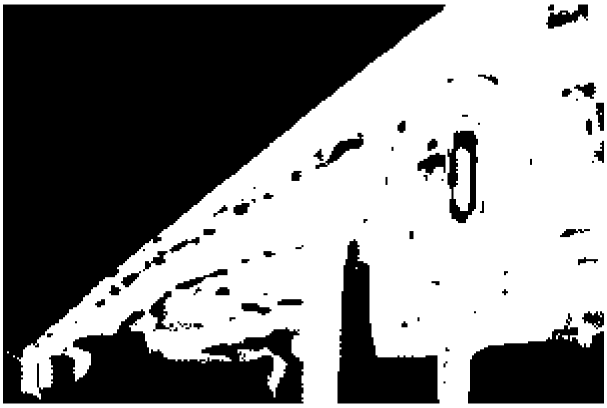


Figure 45 Result of colour segmentation using the sample program; speed sign invisible in the lower image

Colour segmentation is the first step in our image processing. Hence we focused our research on correctly segmenting the image based on colour and irrespective of illumination. The following chapter describes the problem faced while performing the colour segmentation and the solution to the problem.

5 Proposed Colour Segmentation Method and Program

5.1 Introduction

A Computer Vision system has a fundamental capability of interpreting components of an image, by segmenting it into several disjoint regions possessing different properties [31]. This capability of a computer vision system is known as colour segmentation. Colour is one of the most distinct features of an image, so researchers usually use colour for segmentation of images, as, in traffic sign detection algorithms [99].

Colour Segmentation has been applied in various experiments and methods [63, 68, 73, 80] proposed for detection of traffic signs using colour vision. This is done by various methods, such as empirically choosing the threshold values for performing segmentation of an image [80] , by separating traffic sign from the background [99] or by using histogram thresholding [100] .

The methods proposed above, segment images into achromatic and chromatic subspaces and then normalise the entire image to extract the dominant colour out of them. The achromatic grey pixels are either categorised into white or black pixels before normalisation. Although the methods proposed worked for most of the images, they eliminated many signs in images with low illumination. Hence, the problem of detection of speed signs in dark/gloomy images has been addressed in this research, as it is vital for any computer vision software to be able to correctly detect speed sign images irrespective of the conditions in which the image has been produced. The pseudo algorithm steps applied for detection of signs, in an image were:

1. Colour Segmentation
2. Edge Detection
3. Contour / Shape Detection

This chapter will discuss the need for another colour segmentation technique and the author’s approach towards successfully detecting speed signs in a dark image with less than average brightness and other image processing techniques used in the software.

The chapter has been divided into 3 more sections, with the next two sections explaining the problem and solution, respectively, in detail. The last section of the chapter explains the algorithm used and describes the software created as a result.

5.2 The Problem

In designing robust software, the tolerance of error is very small, and hence various individual methods were tested for colour segmentation of speed signs in provided images. The proposed methods mentioned in chapter 3 were initially tested against each other to measure their performance. Missing signs was discovered to be the major problem. After careful observation, it was found that these missing signs were actually the signs located in achromatic regions of the image with below average brightness in and around the sign. Further analysis of various methods mentioned in the papers above, revealed two prominent problems:

A. Detection of signs in dark images (images with less than average brightness) is limited or not possible using the above mentioned methods due to the problem of chromatic pixels falling in achromatic region [63]. This makes designing a program for recognising speed signs difficult and hence this problem needs to be addressed.

B. Even though some of the methods mention detection in the achromatic space [63, 80], no one actually discusses the problem associated with achromatic pixels which are achromatic due to noise, poor quality of the image, darkness or low illumination caused by various factors.

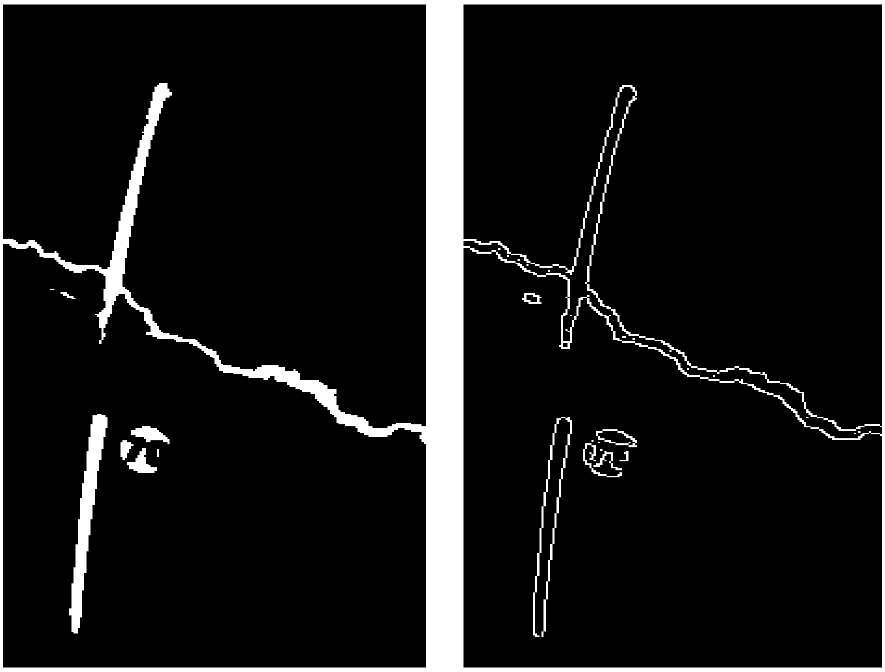
Thus these problems need to be addressed.

Colour segmentation of the sample images was done by using all the three methods mentioned in chapter 3 [63, 68, 80] . The CIECAM method [73] was not used as Emgu CV does not have the ability to use the CIECAM colour model. On most of the images, all the methods behaved quite similarly, but on some occasions signs were not detected by any of the methods. There was an inconsistency in detection of signs with similar images that were tested with all three colour segmentation methods.

When the images produced after the colour segmentation methods were carefully observed, the following was found:

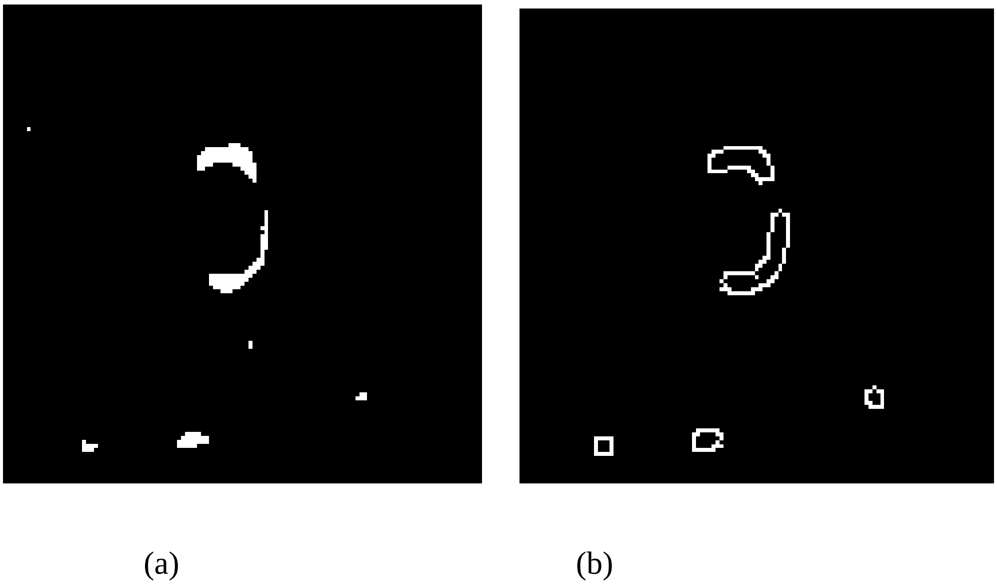
1. Some of the signs in dark images had their outer boundary merged into the background
2. Some other signs had their inner and outer boundaries distorted and merged with the background.

After analysing numerous similar, it was found that this phenomenon was due to the red pixels on the outer and inner regions of a sign turning into achromatic pixels. This conversion of chromatic pixels into achromatic pixels was due to low illumination or presence of shade in that portion of the image or was due to the induced noise in the image, and hence caused the sign to disappear in the background before the edge detection process could be applied. This caused problems in the shape detection part of the program, which follows colour segmentation. When observed after edge detection, it was found that the sign did not exist or else were distorted and did not have the circular or elliptical shape of a speed sign. Some of the images shown below depict where these methods failed in segmenting the sign from the background:



(a) Method 2006, incomplete sign (b) Method 2006 failing canny edge detection

Figure 46 Image showing distortion in speed signs



(a) Method 2010 broken sign after colour segmentation

(b) Method 2010 sign after edge detection

Figure 47 Image showing distortion in speed signs, after and canny detection

These inconsistencies of the methods depicted above are a major problem for designing commercial software. Hence a robust colour segmentation method was applied for treating the achromatic pixels located in the less illuminated portions of the image. This has been addressed in the next section.

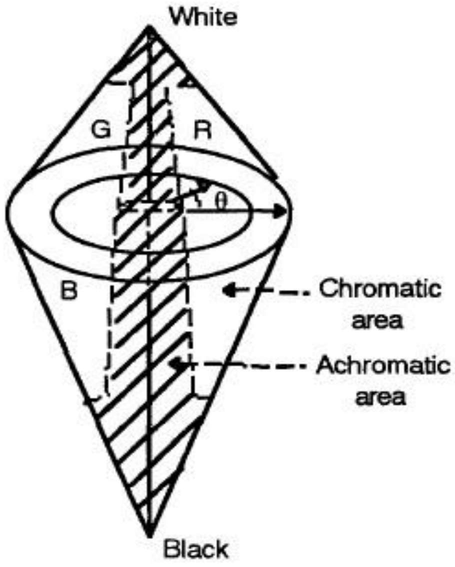
5.3 The Solution

A new colour segmentation method to process images with low illumination condition is proposed. Based on literature, the author tried different methods and combinations of methods to determine the most suitable colour segmentation technique for the traffic sign images. No method produced satisfactory results Thus, we propose a new method to segment the images based on colour. The method proposed has been designed specifically with a focus on low illuminated pixels of the image that turn into achromatic pixels and hence could not be processed by any of the earlier methods proposed. This colour segmentation technique has been designed to enhance the traffic signs in an image, irrespective of illumination, background and size of the image or traffic sign.

Based on comparison with methods such as Hue and Saturation thresholding [101] and Ohta [102] space thresholding, the RGB Normalisation method has been described as one of the most suitable methods for colour segmentation [80]. In papers [31, 63] hue

(H) and saturation (S) have been shown to be invariant to the effects of illumination variations. Hence, based on these observations, we propose a colour segmentation method that combines both the RGB and HSV colour spaces into one method.

In our proposed method we first obtain the hue, saturation, value, red, green and blue values of each pixel p(x, y) from both the colour spaces. Then, based on the techniques of image segmentation, we divide the HSV colour space into a zone model. The zone model is divided into four zones consisting of both the chromatic and achromatic space, where each pixel is placed into a zone, depending on the values of Saturation (S) and Value (V) of the pixel. The chromatic/achromatic spaces used, are based on a mixed proposition by Tseng et al [31] and Vitabile et al.[67] .We have proposed new (different) limits for various zones to cover all range of possibilities and have also added two more zones to cover all regions, as it is based on HSI model applied to HSV cylindrical shape , as opposed to the HSI conical shape proposed by Tseng.



The configuration of chromatic and achromatic areas in the *IHS* space.

Figure 48 Chromatic and achromatic regions.[31]

Shown above is the achromatic and chromatic region as proposed by Tseng and Chang [31] in a HSI conical model.

5.3.1 The Zone Model

The Zone Model consists of the following zones:

Zone 1: Definite achromatic pixel:

(S ≤ 0.125) and ((V ≤ 0.25) or (V ≥ 0.95))

When saturation is too low and Value is too high or low, the Hue has no effect. Hence depending on the value of ‘S’ and ‘V’ we define any pixel falling into this zone as white or black.

Zone 2: Achromatic pixel:

(S ≤ 0.25) and ((V ≥ 0.25) and (V ≤ 0.95))

In this zone all the chromatic coloured pixels are either converted into black or white, if their RGB values are similar or else their colour is enhanced, so that the colour can become prominent during RGB Normalisation.

Zone 3: Sub achromatic/chromatic pixel:

((((0.95 ≤ V) and (V ≤ 1)) and ((S ≥ 0.5))) or (((0 ≤ V) and (V ≤ 0.25)) and ((S ≥ 0.5))) or (((0.125 ≤ S) and (S ≤ 0.5))))

In this zone the light colour and dark colour pixels are converted to their dominant colour by enhancing their Hue value. In addition, any other pixel which has less than 50% of saturation is converted to its dominant colour.

Zone 4: Chromatic pixel:

(((0.25 ≤ V) and (V ≤ 0.95)) and ((S ≥ 0.5)))

This is just the chromatic zone, where all the pixels have their colour values left as it is, as there is no effect of shadow or variation on these pixels**.**

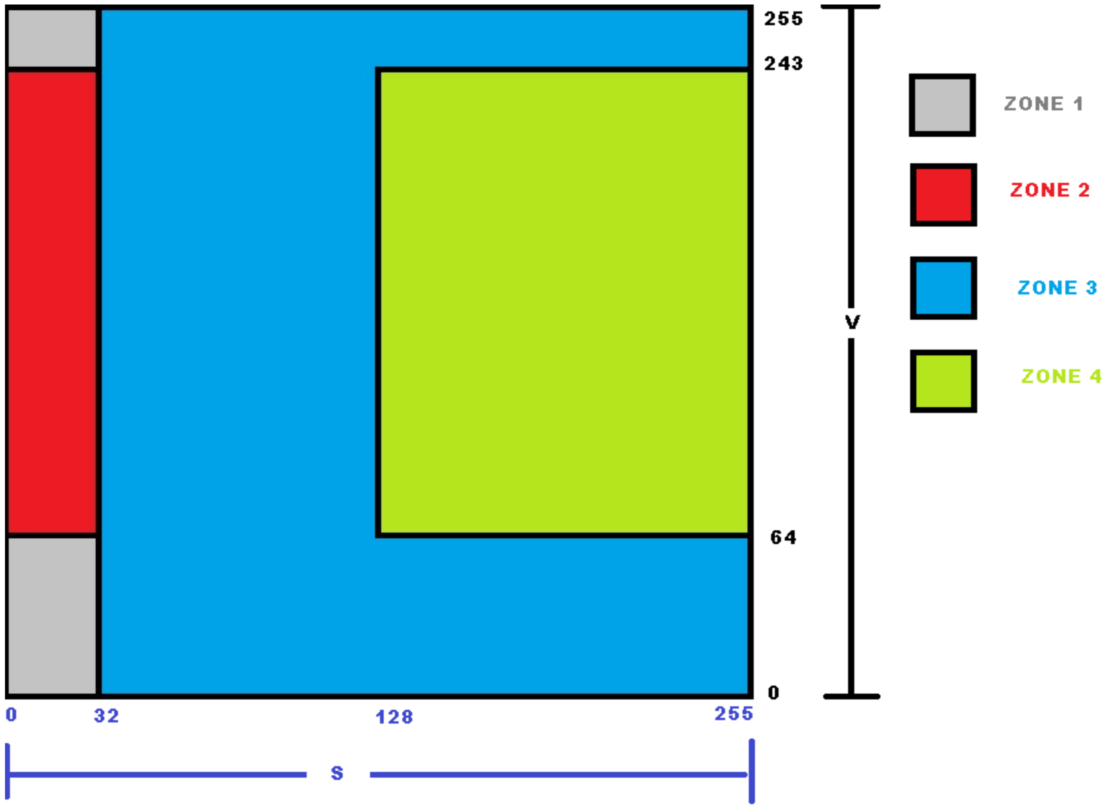


Figure 49 The Zone Model, configuration of four zones in the HSV Colour Space.

Once a pixel p(x, y) is placed in the zone model, RGB normalisation is applied to enhance the dominant colour of the pixel. This technique converts an achromatic pixel into its dominant colour, thus, discriminating the background from the sign. After the colours of the image are enhanced, we filter the 'R' red dominant pixels by using a red mask as proposed in [68]. Then this red image is converted into a grey image where red pixels are converted into white pixels and other pixels are converted into black.

After colour segmentation, the grey image obtained is further processed by edge detection and shape matching.

Using this proposition (the zone model and RGB normalisation) helped us to handle the effects of illumination in a better way than other proposed methods (results discussed in chapter 6).

Described in the next section is the implementation of the zone model.

5.4 The Algorithm

Here we describe the algorithm generated for implementing the solution proposed in section 5.3 and other image processing functions that were added to the sample program (mentioned earlier). This section has been divided into four subsections, defining the four main functions generated for developing the program and test the software created. Shown below, is the diagram depicting the algorithm implemented in creating this software:

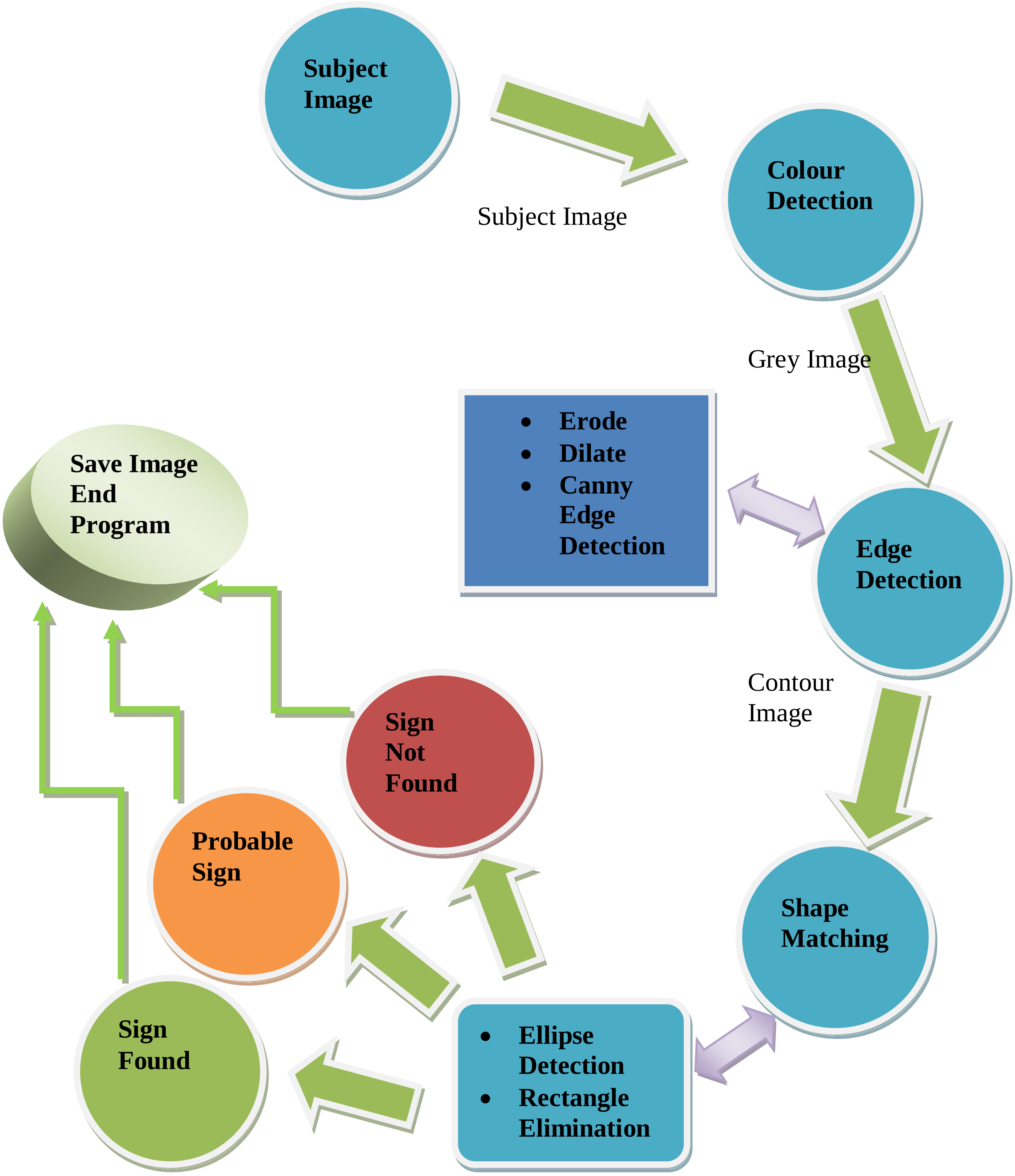


Figure 50 Diagram showing the steps in the program

5.4.1 The Colour Segmentation

The colour segmentation as mentioned earlier, was developed using HSV colour model based on the cylindrical representation. Traffic signs have prominent distinguishing colours for drivers to easily recognise them, this makes them easy for colour segmentation provided the illumination is high enough and there is no extra noise in the image. As stated earlier, these two conditions have adverse effects while determining the colours in signs; hence we have used the combination of HSV and RGB colour spaces here. We use saturation and value to determine the location of a pixel in any of the four zones mentioned above, this is shown below:

```
    //  *********  ZONE  1  *********  Definite  Achromatic  Zone  (S  ≤
    0.125), (V ≤ 0.25) or (V ≥ 0.95) ********
    if ((S <= 32) && (V <= 64))   // S < 12.5% and V < 25%
    {// means the pixel is black }
    else if ((S <= 32) && (243 <= V)) // // S < 12.5% and V > 95%
    {  // means the pixel is white }
    else if (((H >= 0) && (H <= 15)) || ((H >= 150) && (H <= 180)))
//RED
    {
    //****** ZONE 2 Achromatic Zone ******
      if (((64 < V) && (V < 243)) && ((S <= 32)))
       R = R * 2; // Intensify Red

    // ********* ZONE 3 ********* Sub Achromatic / Chromatic Pixel
    Zone except Zone2 ********
      else if ((((243 <= V) && (V <= 256)) && ((S > 128))) || (((0
      <= V) && (V <= 64)) && ((S > 128))) || (((32 <= S) && (S <=
      128))))
      R = R * 2; // Intensify Red

    // ********* ZONE 4 ********* Chromatic Zone (S > 50%)  ********
      else if (((64 <= V) && (V < 243)) && ((S > 128)))
    // Do Nothing
    }
```

Code 6 The Zone Model applied in the code

Based on its zone location, a pixel is either converted to a white or black (zone 1), or colour is enhanced if in Zone 2 or 3. The pixel is unchanged if it is in zone 4, which is the chromatic zone and is expected to have a higher value of colour which can be enhanced with the RGB normalisation, which is applied later. After the total value of a pixel is enhanced based on the location in the Zone model, RGB normalisation is applied. This operation is performed based on the method proposed in section 3.2 [68]. Here, we first find the total Intensity (I) of a pixel and then the pixel's RGB channels are normalised by the intensity, as shown below:

```
  // Normalise all the pixels
      I = (B + G + R) / 3; // Get the intensity
      b = B / I;
      g = G / I;
      r = R / I;
```

Code 7 RGB normalisation, b,g and r are the normalised values.

Once the pixel is normalised a colour mask is applied to get the dominant colour (red, blue, green or yellow) in the entire image .The masking has been suggested in [68] and can be applied based on requirements of the program. As in our case, it is the detection of speed sign; hence we apply a red mask as shown below:

```
// Apply the mask for desired colour
      r = (r - (g + b) / 2); // Red Mask

AchromImage_C[j, i] = new Bgr(B, G,r*255); // Normalised image
```

Code 8 Red colour mask to get the image with Red as dominant colour.

Once a new image is generated using the RGB normalisation, it is passed through a grey colour mask. This is done by converting the image into a gray image with all the pixels having red value between the range of 64- 255 being converted into white (255) and other pixels being converted to black (0). This gives us a grey image to process further using edge detection. One of the sample grey images is shown below:

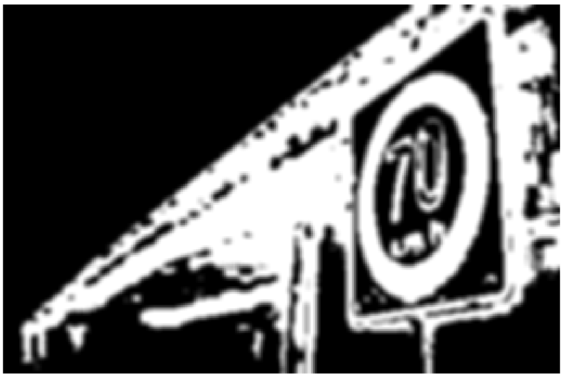


Figure 51 Grey image of a speed sign, generated after RGB normalisation

5.4.2 Edge Detection

The grey image generated after the colour segmentation is dilated and eroded as per the sample program, with the exception of the number of iterations. The number of iterations performed is reduced to 2 each for erosion and dilation after empirically testing the program on different images. Canny edge detection is applied on the images to get an image with contours for edges of geometrical structures in the image. An image showing the contours and the code for canny edge detection used is shown below:

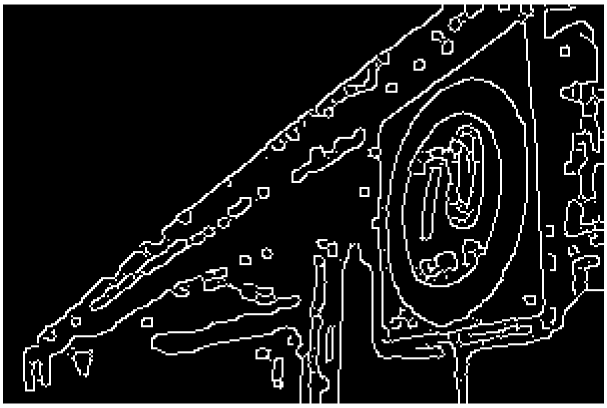


Figure 52 Image of a speed sign showing edges in form of contours generated after canny edge detection

```
// **** FIND Edges *****
canny1 = 100;
canny2 = 150;
Image<Gray, Byte> smoothImg = RedDominantImage.SmoothGaussian(5, 5, 
1.5, 1.5);
smoothImg._Dilate(1);
smoothImg._Erode(1);
smoothImg._Erode(1);
smoothImg._Dilate(1);
Image<Gray, Byte> canny = smoothImg.Canny(new Gray(canny1), new 
Gray(canny2));
return Returnimage = canny.Convert<Bgr, byte>();
```

Code 9 Canny edge detection with erosion and dilation

5.4.3 Shape recognition

For shape recognition of the image we use same procedure as mentioned in the sample program but with the difference of storing the base/sample image contour, rather than its shape as mentioned in section 4.3.3. A sample image is first selected e.g. speed sign and its largest contour is found and stored only if it is an ellipse (using the 'CheckEllipse' method). This is the preselected sample image with speed sign as the largest object in the image. The sample image (shown below) being used in the program has been taken from NZ NZTA website, as it defines specification requirements for a NZ speed sign. This sample image is matched with the subject image using the 'cvMatchShapes' function, parameters for matching are the same as used in the sample program.

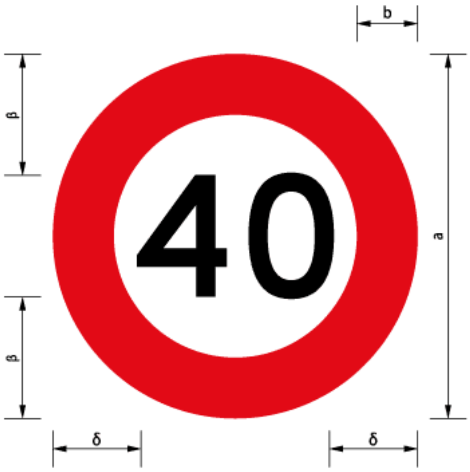


Figure 53 Sample image for speed sign 40 [103]

Shown below is the sample image's contour storing method which is created in the beginning of the program:

```
for (; SampleContours != null; SampleContours = SampleContours.HNext)
  {
       int contourcount = SampleContours.Total;
       SampleContours.ApproxPoly(SampleContours.Perimeter * 0.02, 0,
       SampleContours.Storage);
       // Check for Ellipse and the largest contour
       if (SampleContours.Area >= TempSampleContours.Area)
       {
              isEllipse = CheckEllipse(SampleContours, sampleGauss); //
              Check if the Contour is an ellipse or not ?
              if ((isEllipse) && (SampleContours.Area > 100))
                {
                // Save the Ellipse as TempContour
                     TempSampleContours = SampleContours;
                }
        }
       SampleContours = TempSampleContours;
  }
```

Code 10 Contour storing function: Stores the largest elliptical contour in the base image

5.4.3.1 The ‘CheckEllipse’ Method

This is one of the most important functions of the program that has been created to deal the problem of inconsistent shape detection in the sample program. This method, as the name suggests, checks whether a given contour is an ellipse or not. As mentioned earlier, the contours being passed to the ‘cvMatchShape’ function were being inconsistently detected irrespective of their shape, e.g. a rectangle or an octagon were being treated as circle, so this method was designed to check the shape of the contour . This method is divided into three steps:

A. Least Square Fitting

B. Elliptical Equation

5.4.3.1.1 Least Square Fitting

In this function we have used the inbuilt function of ‘EllipseLeastSquareFitting’ which uses the property of least square fitting to fit an ellipse on the given contour points. Below is a brief description of the least square fitting technique, from *[104].*

Least square fitting, is a mathematical procedure that is used to fit data among a given set of points. It finds the best fitting curve to a given set of points by minimising the sum of the squares of the offsets (“the residuals”) of the points from the curve [105]. It assumes that the best fit curve of a given type is the curve that has the minimal sum of the deviations squared (least square error) from a given set of data. The sum of squares of the offsets is used, as it allows the residuals to be treated as a continuous differentiable quantity. It is based on the method of least squares.

Considering a set of data points as (x_1,y_1) , (x_2,y_2) ..... (x_n,y_n) where x is the independent variable and y is the dependent variable. The fitting curve $f(x)$ has the deviation (error) d from each data point, i.e. $d_1=y_1 - f(x_1)$, $d_2=y_2 - f(x_2)$... $d_n = y_n - f(x_n)$. According to the method of least squares, the best fitting curve will have the property that:

$$\Pi = {d_1}^2 + {d_2}^2 + \ldots + {d_n}^2 = \sum_{i=1}^{n} {d_i}^2 = \sum_{i=1}^{n} [y_i - f(x_i)]^2 = \text{a minimum}$$

Based on the same concept, using an ellipse equation, the method of least square fitting is applied to fit an ellipse around a set of points as shown below.

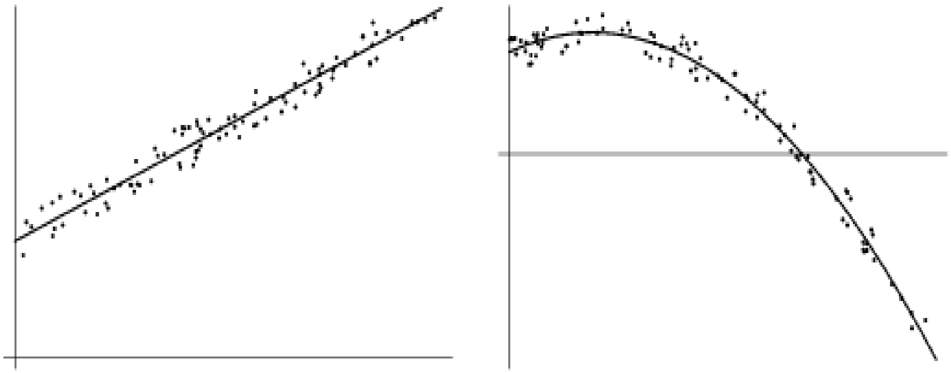


Figure 54 Image showing least square fitting [105]

Shown below is the code as applied in the program:

```
// convert Contour into Collection of Points
            foreach (Point p in EllipseContour.ToArray())
            {
                pts[count++] = new PointF(p.X, p.Y);
                _x += p.X;
                _y += p.Y;              }
Checkellipse = Emgu.CV.PointCollection.EllipseLeastSquareFitting(pts);
// Fit an ellipse using Least Square Regression
```

Code 11 Convert the contour into points and then run ‘EllipseLeastSquareFitting’ method to fit an ellipse to the points

After an ellipse is fit to the set of contour points, we get the properties of ellipse important for finding the equation of ellipse, like major axis, minor axis, centre of the ellipse and the rotation angle.

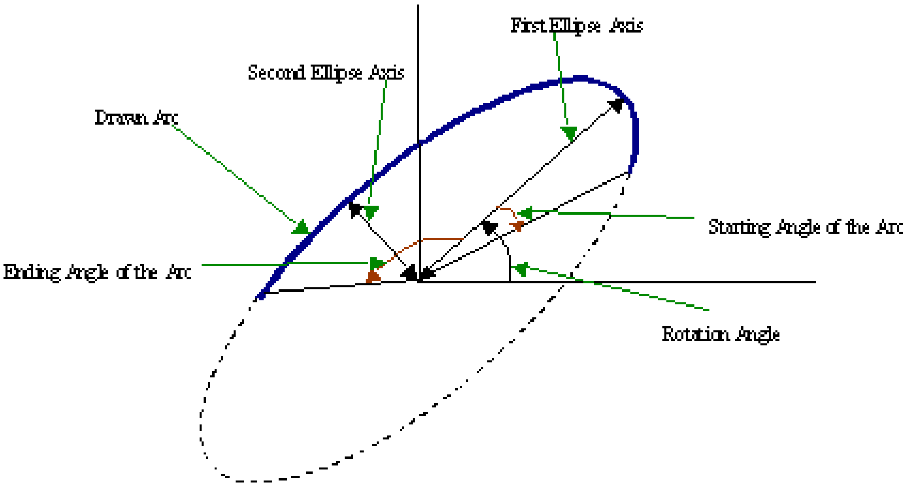


Figure 55 Image showing the properties of an ellipse [89]

5.4.3.1.2 Ellipse Equation

Once the properties of the fitted ellipse are obtained, they are put into the general equation of ellipse depending on the orientation of the ellipse. All the points are passed through the elliptical equations and the points satisfying the equation (within a range of ± 25%) are counted. The higher the number of points on the fitted ellipse, the closer is the shape of the contour to an ellipse. After empirical testing, it was decided that if seventy percent of the points satisfy the elliptical equation, a contour would be considered as an ellipse. In addition, for total number of points less than 24 in a contour, the percentage was increased to seventy five.

```
if (horizontalellipse) // check if the ellipse is vertical or
horizontal
{      eqn1 = ((Math.Pow((X - H), 2)) / (Math.Pow(_A, 2))) +
((Math.Pow((Y - K), 2)) / (Math.Pow(_B, 2)));
      if (eqn1 >= 0.75 && eqn1 <= 1.25)
      counterTrue++;
      else
      counterFalse++;
}
  else  // VERTICAL ELLIPSE
      { eqn2=((Math.Pow((X-H),2))/(Math.Pow(_B,2)))+((Math.Pow((Y-
      K),2))/(Math.Pow(_A,2)));
            if (eqn2 >= 0.75 && eqn2 <= 1.25)
            counterTrue++;
            else
            counterFalse++;
      }
```

Eqn1 and eqn2 is for a horizontal and vertical ellipse respectively

Code 12 Shown above is the code of the equations for both horizontal and vertical ellipse.

5.4.4 Overall Detection

Finally, once the shape of the contour is decided along with the matching of the shapes of the base image contour and the subject image contour, the results are passed through a simple conditional test. This is to determine if the contour is a red ellipse or not. The values of the various variables have been decided after empirical testing of real data images as provided by the company (Geosmart). The following conditions were used to test if the image actually had a red ellipse or not:

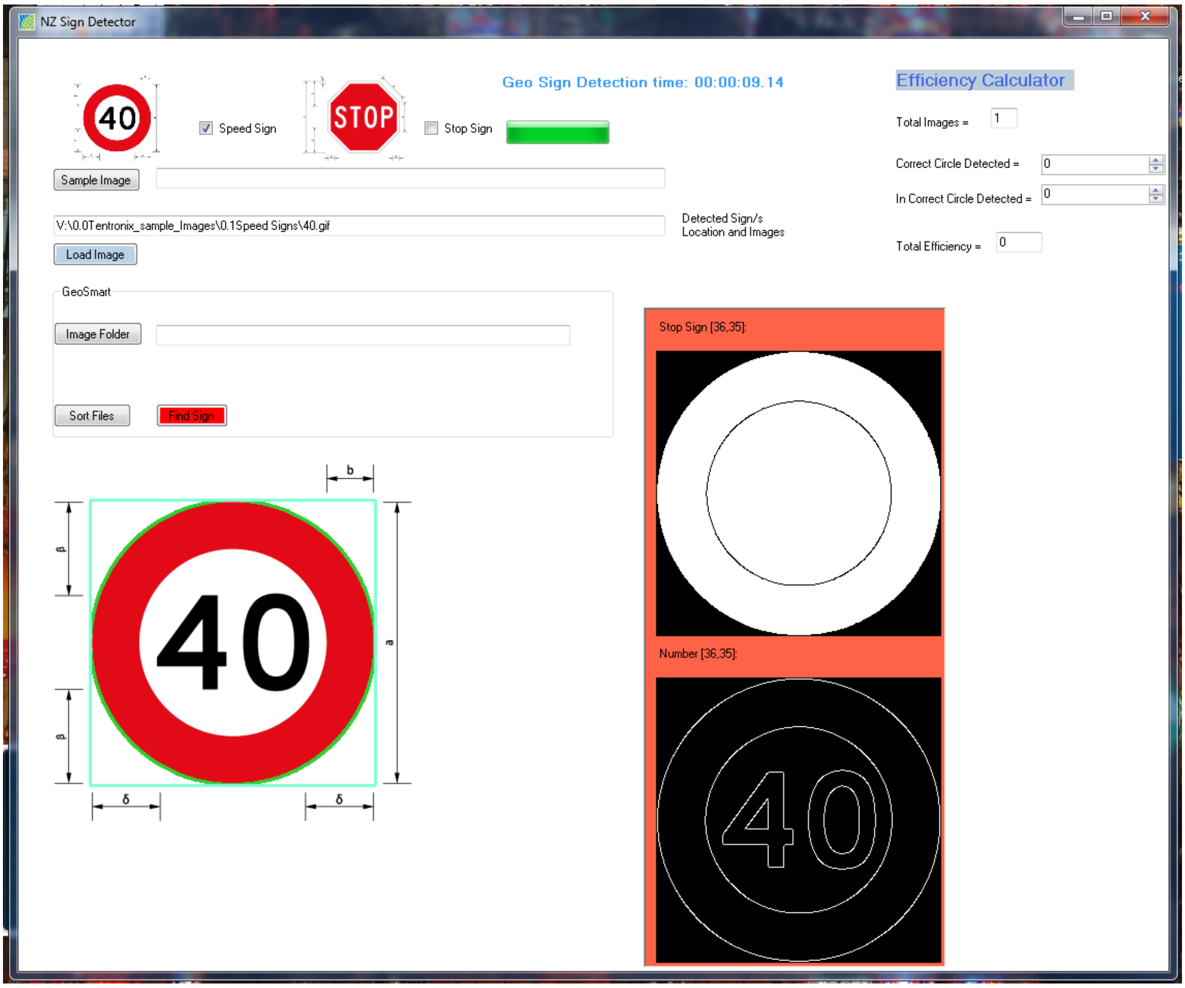
```
ratio2 = CvInvoke.cvMatchShapes(SampleContours, contours,
Emgu.CV.CvEnum.CONTOURS_MATCH_TYPE.CV_CONTOURS_MATCH_I3, 0);
isEllipse = CheckEllipse(contours, img); // Check for the shape
//****** INCORRECT CONTOUR (Not a SIGN) ********/
if (((0.10 < ratio2)) || (!isEllipse)) //not a good match of contour
shape else
{ // ***************  SIGN FOUND *************/
     if ((ROI_Area >= (0.00240 * (Img_Area))))
      { // Definite Sign }T
      else
      { // Probable Sign }
```

Code 13 Conditions for testing of sign

6 Test Results: Compare results of different methods used in the program

This chapter discusses the results of all the methods proposed in the literature review as compared with the new method proposed in this research. All the proposed solutions were tested for results using same set of images. Some images were used from the original company database and some of the images were downloaded from the internet, so as to test the program for various images with changes in illumination conditions. The subsequent sections will discuss the results of software run on the sample images, based on the papers discussed in chapter 3.

Shown below is the Graphical User Interface (GUI) developed for automated and manual testing of the images.



NZ Sign Detector

☑ Speed Sign

□ Stop Sign

Geo Sign Detection time: 00:00:09.14

V:\0.0Tentronix_sample_Images\0.1Speed Signs\40.gif

Efficiency Calculator

Total Images = 1

Correct Circle Detected = 0

In Correct Circle Detected = 0

Total Efficiency = 0

Figure 56 GUI for NZ Speed Sign Detector Software

6.1 Equipment Used

The program was tested on the following hardware and software platform:

- Processor: AMD Turion x2 mobile 64 bit
- Memory: 3 GB DDR2
- Graphics Memory: Shared 128 MB
- Disk Space: 160 GB Hard disk space with images stored on it
- Operating System: Windows 7 x64 bit
- Visual Studio 2008 was used as the IDE to develop the GUI using visual C#
- Open CV was used for image processing libraries , and
- Emgu CV was used as the C# wrapper for Open CV

The test was performed on a folder containing 538 files, with a size of 29.9 MB (31,432,704 bytes). As mentioned; these were mixed files with the actual data taken from the company's database and some images from the internet. Hence the images were of different sizes and shot under different conditions. The files were with and without signs as provided by the company, thus the aim was to test the program for detection of speed signs, false detection, missed signs and probable signs in the images.

The program was run for each of the different proposed methods with all other parameters being the same so as to compare the performance of all the methods against each other. Once the automated test was finished, two log files, one containing the detected signs and the other containing probable signs were created. The data stored in these log files was manually confirmed and was manually tested for any failures in detection using the efficiency calculator in the GUI. Further sections discuss the test results produced by various methods as proposed in the literature review and our proposed RGB Normalisation method.

Testing of the program primarily focused on the colour detection method.

6.2 Results of First Run

Out of 536 files tested, 37 files contained signs easily visible by eye. All the other images were without signs.

The efficiency of a method has been calculated using the following formula:

Correct Sign + Probable + Missed = 37

Efficiency = ((Correct + Probable) / 37)*100

The table below shows the comparison between the results of different methods used for the same set of images.

Method Name	**Time Taken (sec)**	**Correct Signs**	**Probable Signs**	**Missed Signs**	**Correct Percentage** ***(Rounded off)***	**False Detection**
Method 2006	823	16	3	18	51%	20
Method 2008	1001	26	3	8	78%	18
Method 2010	900	19	2	16	57%	7
Our Method ***(RGB Norm)***	2718	25	6	6	84%	10

Table XI A Table comparing the results of first run of software

6.3 Comparison of results for different methods

Shown below are the images that were missed by the different methods for different reasons. Some of the incorrect images have been shown below depicting their behaviour with other colour detection methods as well.

6.3.1 Missed Sign Images

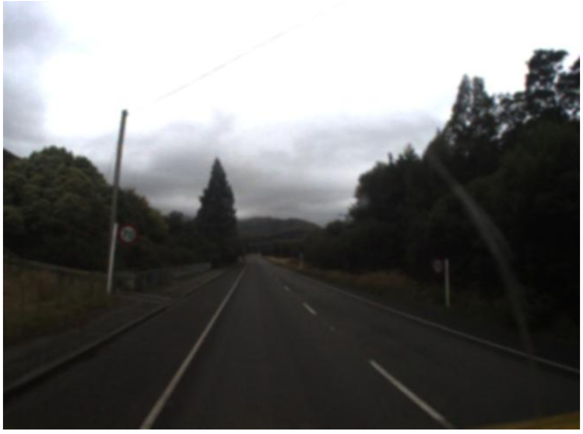
Image 1: 2_517428.823 – Dark Image

Method in which the image failed: Our Method and Method 2006

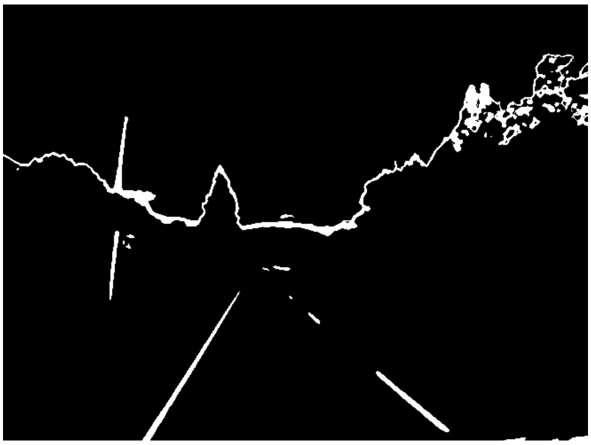
Reason:

- ➢ Our Method: Distorted shape due to white pole being added as noise with the circle.
- ➢ Method 2006: Image is too dark.

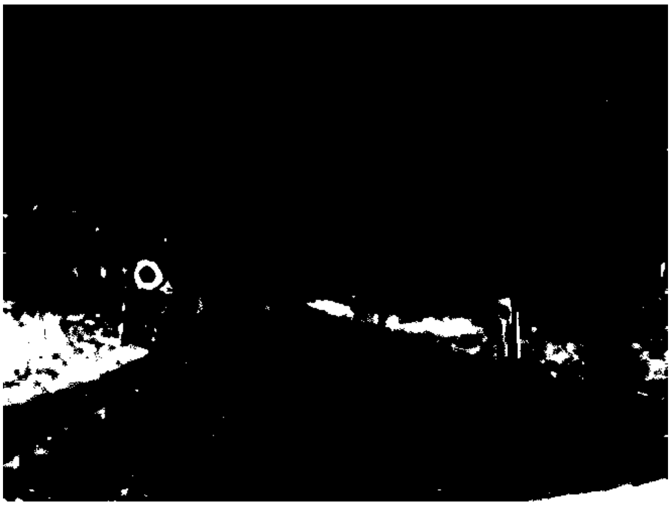
a) Original Image



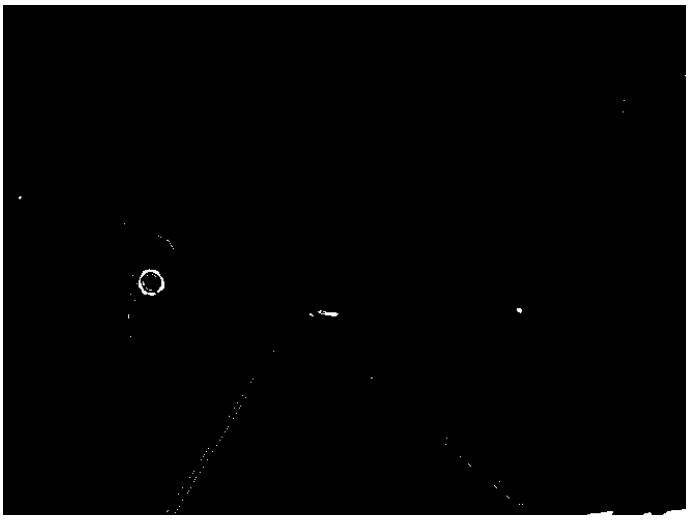
b) Method 2006



c) Method 2008



d) Method 2010



e) Our Method

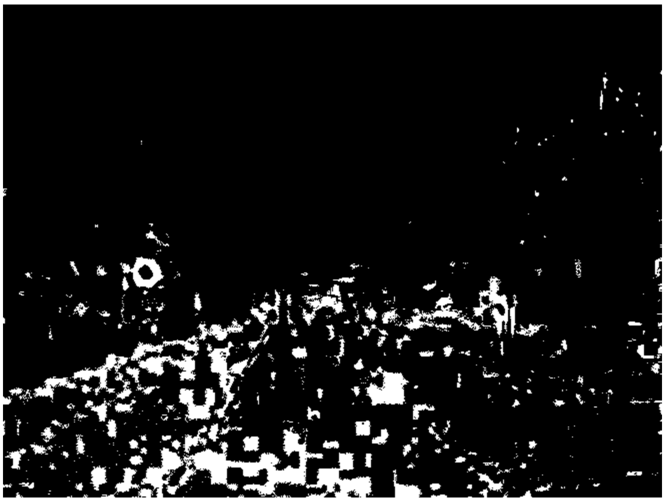


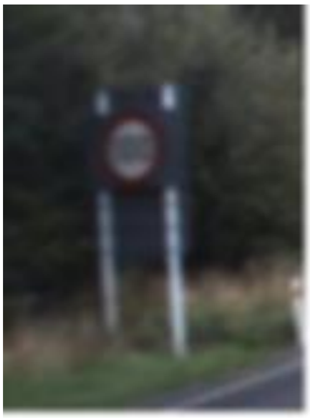
Image 2: 5_517412.924 –Bright image with noisy sign

Method in which the image failed: Method 2008 and Method 2010

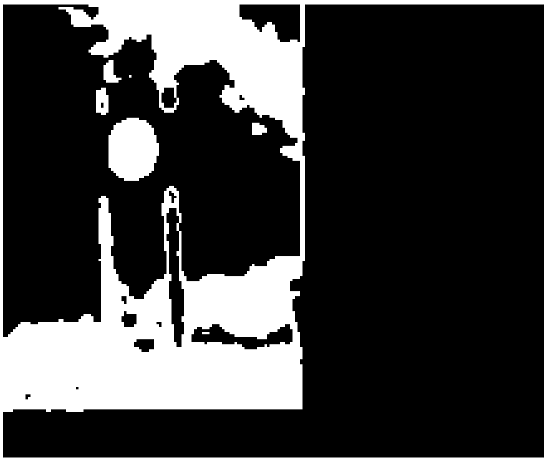
Reason:

- Method 2008: Red circle in achromatic Zone due to poor image quality.
- Method 2010: Red circle doesn't fit the range.

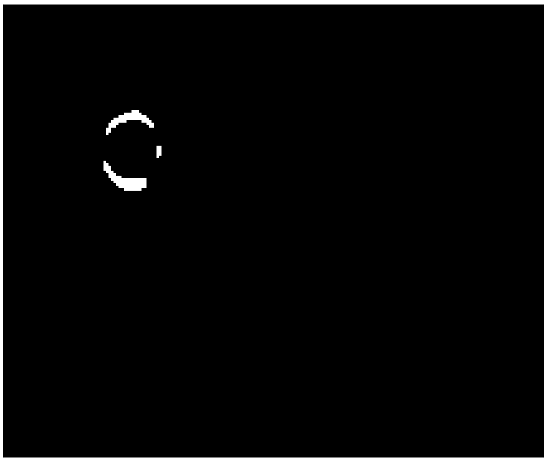
a) Original Image



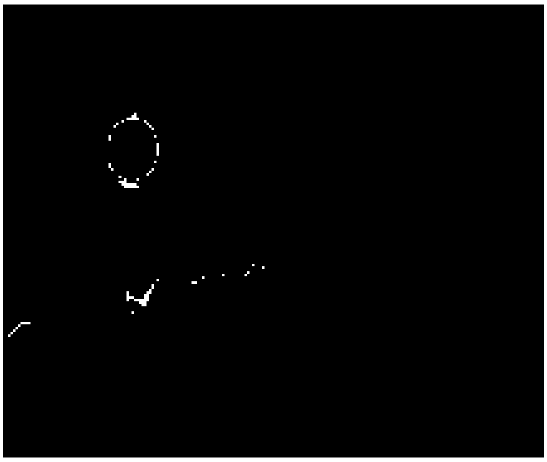
b) Method 2006



c) Method 2008



d) Method 2010 : Blank image



e) Our Method

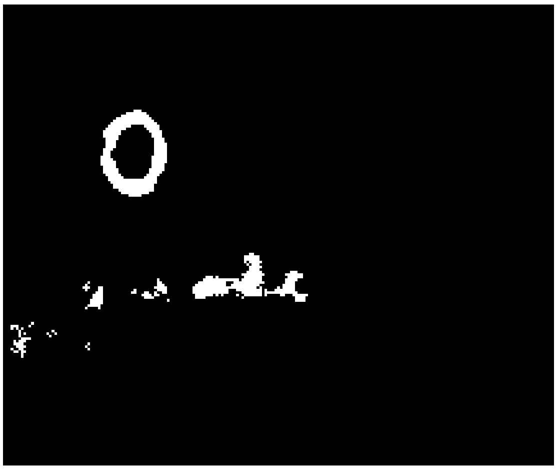


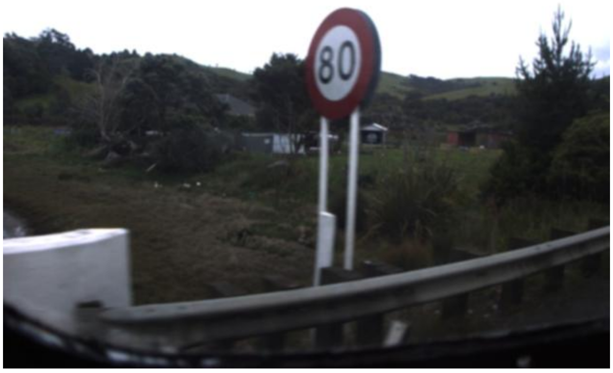
Image 3: 1_509790.134 –Lower half of the image is gloomy

Method in which the image failed: Method 2010 and Method 2006

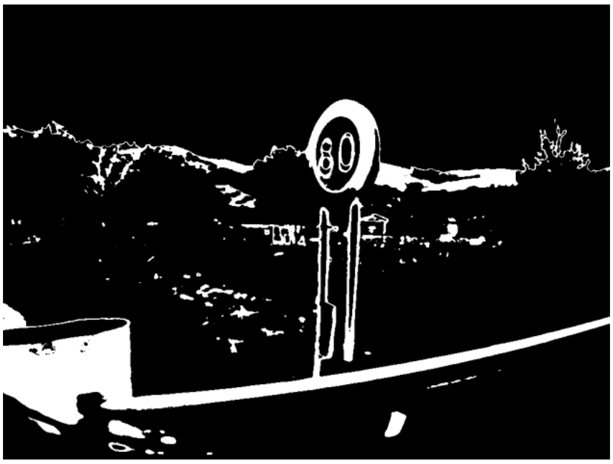
Reason:

- Method 2010: Red circle's top region does not fit in the threshold given in the method's paper.
- Method 2006: Lower part of the image is dark for the method, hence mixes part of '8' with the inner circle, resulting in a distorted edge.

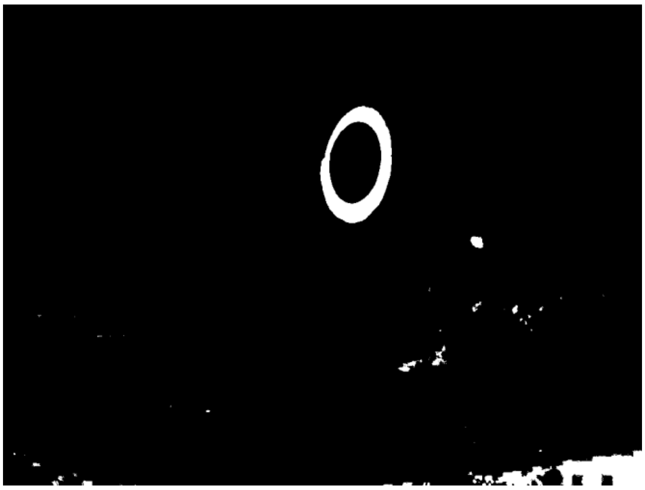
a) Original Image



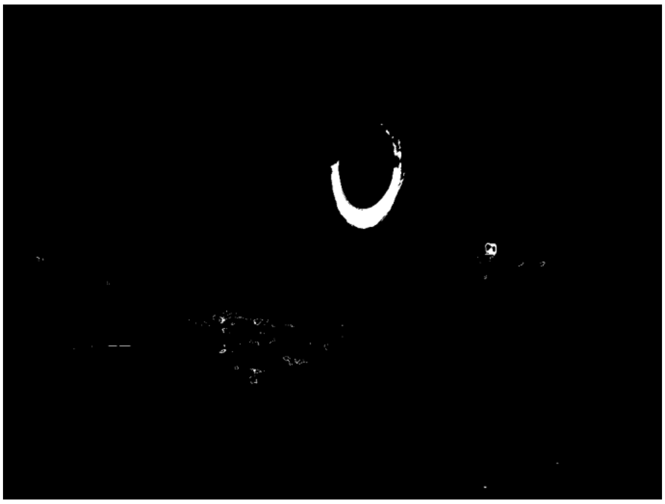
b) Method 2006



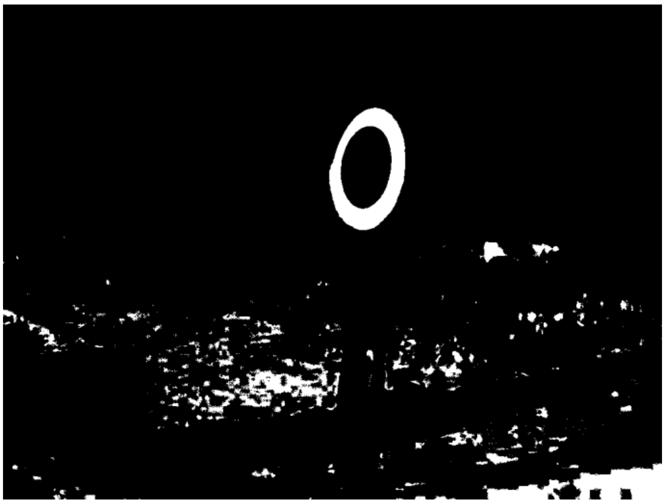
c) Method 2008



d) Method 2010



e) Our Method



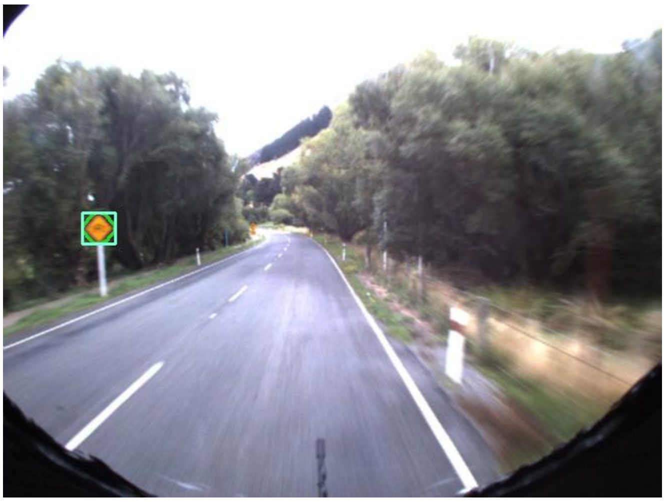
6.3.2 Falsely Detected Images

Image A: 5_517401.823: Yellow Sign

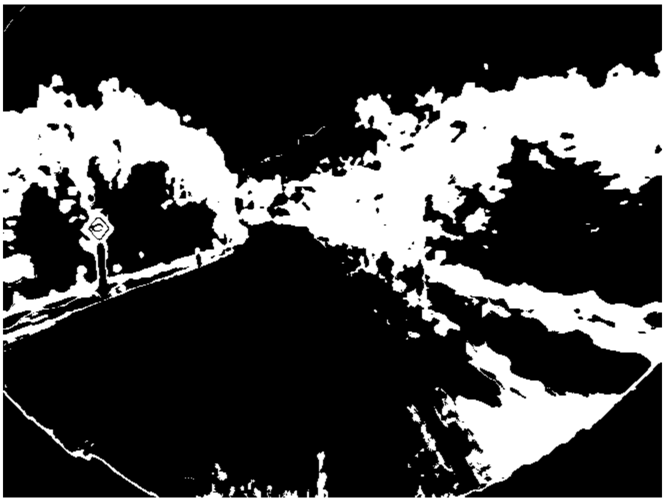
Method in which the image failed: Method 2008 and Method 2010

Reason: Yellow sign being treated as Red circle.

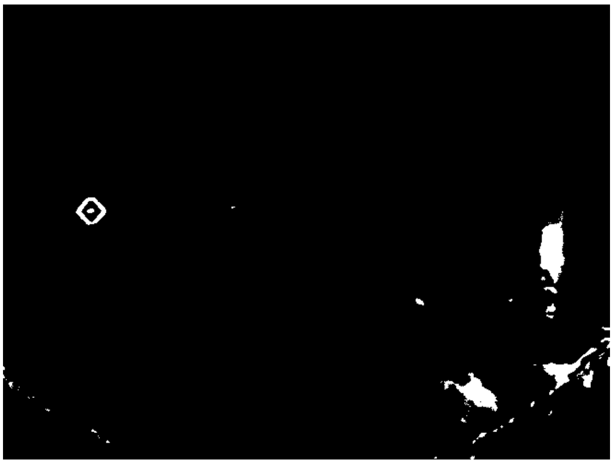
a) Result Image



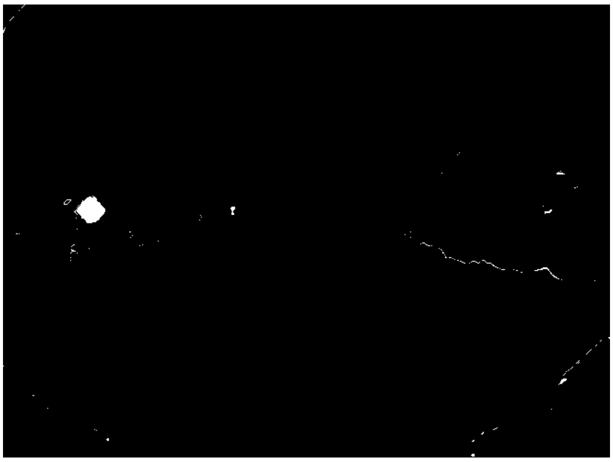
b) Method 2006



c) Method 2008



d) Method 2010



e) Our Method

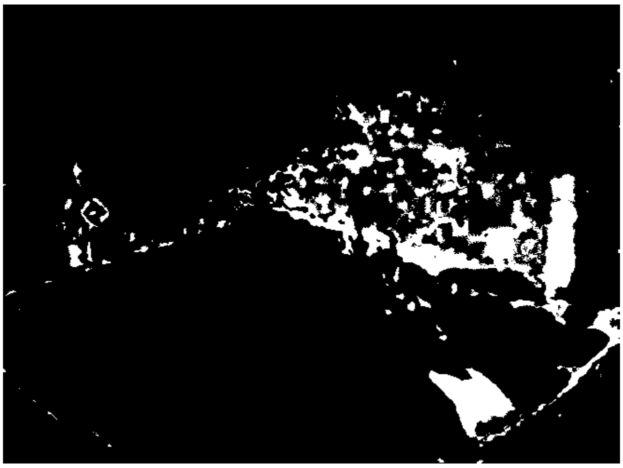


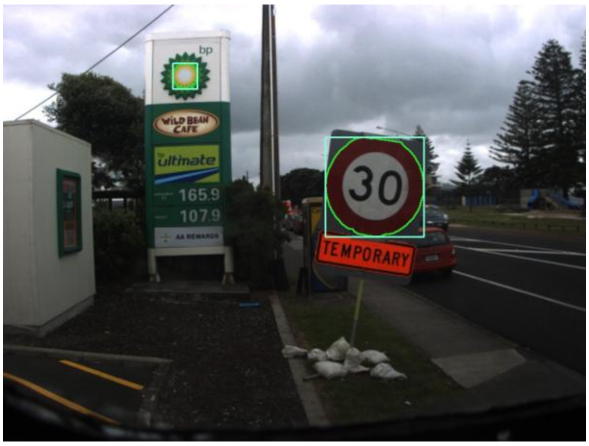
Image B: 1_508851.734: Yellow Circle and Green Circle

Method in which the image failed: Method 2010 and Method 2006

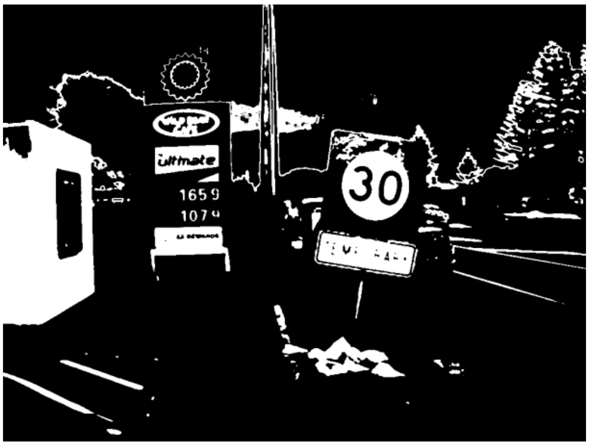
Reason:

- Method 2010: Yellow circle in achromatic Zone being treated as a red circle due to use of threshold. .
- Method 2006: Green circle is being treated as Red circle and misses the sign as the red circle is distorted due to noise on the actual sign.

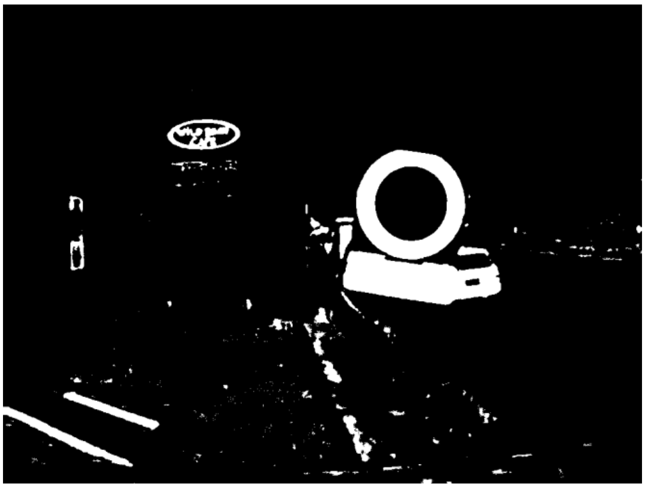
a) Result Image



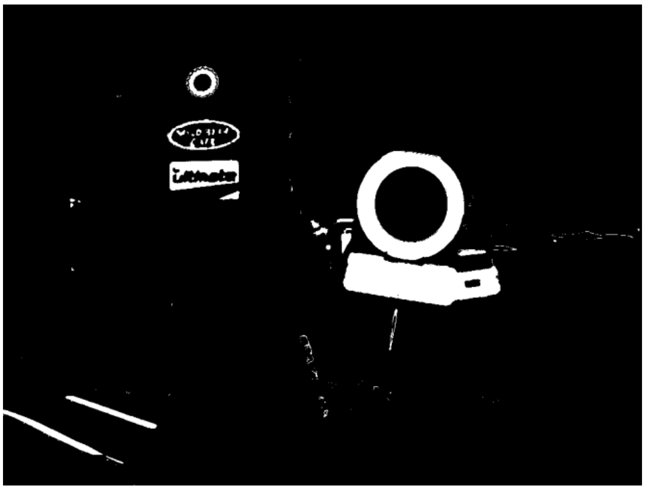
b) Method 2006



c) Method 2008



d) Method 2010



e) Our Method

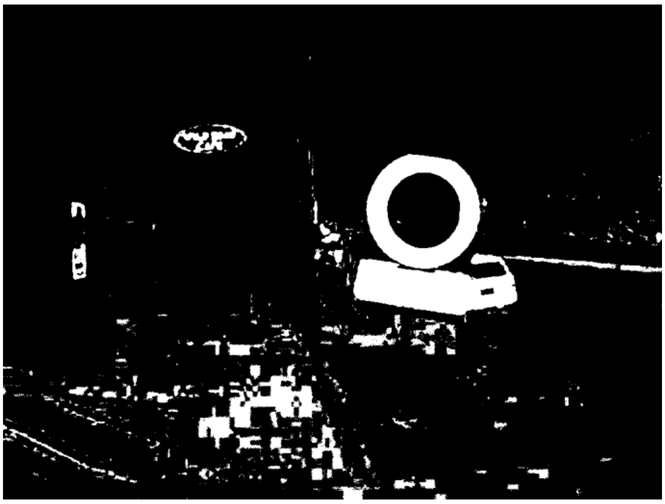
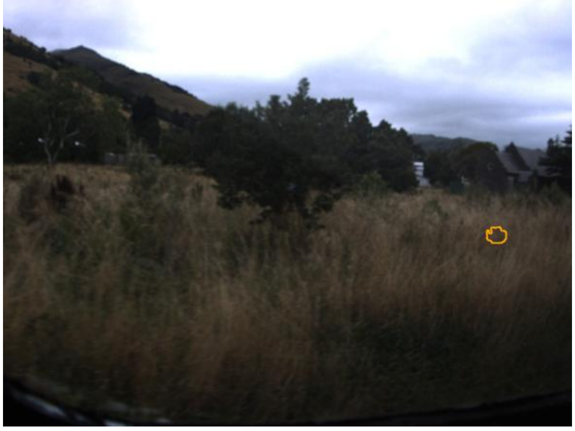


Image C: 1_517413.923: Noise in the Grass

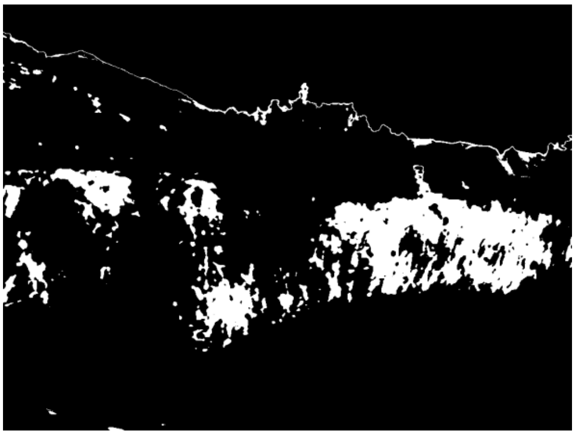
Method in which the image failed: Our Method

Reason: Added noise in the grass due to conversion of achromatic pixels into chromatic red.

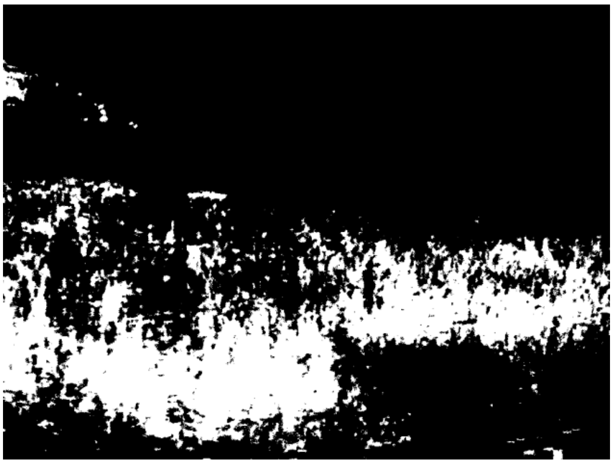
a) Result Image



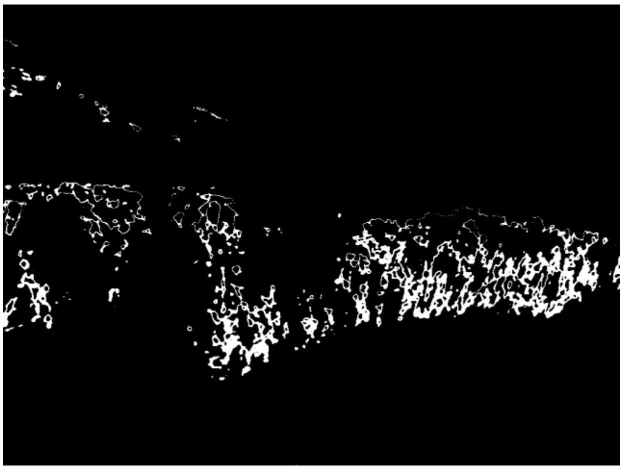
b) Method 2006



c) Method 2008



d) Method 2010



e) Our Method

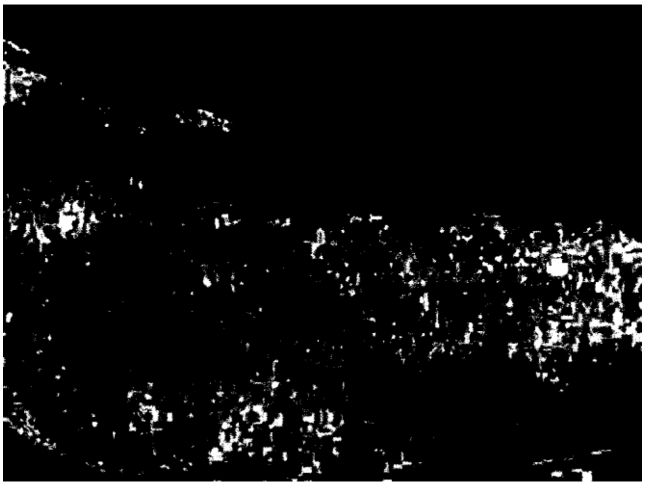
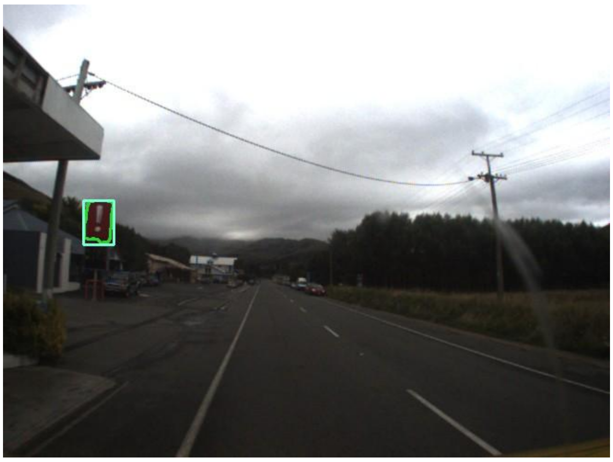


Image D: 2_517445.523: Rectangular Red Sign

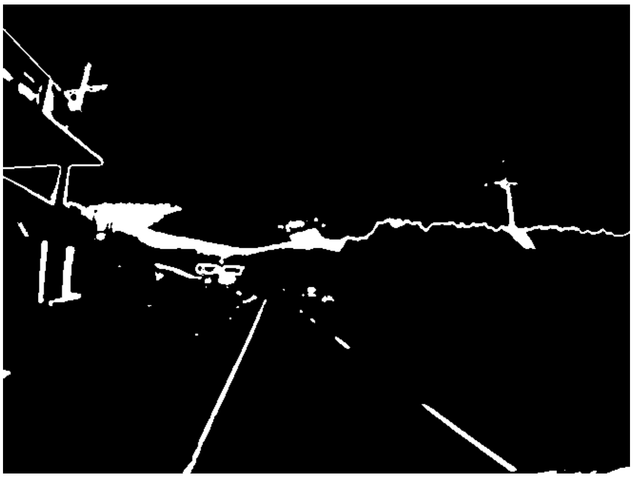
Method in which the image failed: Our Method, Method 2010 and Method 2008

Reason: Red rectangle being detected as ellipse.

a) Result Image



b) Method 2006



c) Method 2008

d) Method 2010

e) Our Method

6.4 Analysis of results

After the first test run described in the section above, the log files generated were observed. The manual analysis was done for the missed signs and the falsely detected signs.

After analysing the results given in section 6.2 and section 6.3, the following was observed:

1. The new method had the best results as compared to other methods, 84% success rate, with the highest success rate in the detection of images with low illumination.

2. The new proposed method was quite slow as compare to other methods; it took almost twice the time of the slowest of other three methods (1001 seconds).

3. The next best method was Method 2008 that proposes use of RGB Normalisation but without any use of the zone model and was also slowest among other three methods with a success rate of 78%.

4. The fastest method was Method 2006 but with lowest accuracy of 51%.

5. False detection of signs was high, due to selection of rectangular signs, with Method 2010 having lowest false detection rate amongst all the method.

Our method mainly failed due to added noise because of conversion of non red colour pixels that were on margin, into the red colour e.g. as shown in image1 of section 6.3. This image turned into a distorted ellipse due to the illumination of white pole which was enhanced when RGB normalisation was applied.

The method proposed though, was more robust on the image set than other methods with increased performance in poor light conditions, but it was very slow taking almost 2 minutes for each MB of image data. It took 2718 sec or 45.30 minutes for 28.7 MB of image data. In addition, it was adding noise in the unwanted areas, due to conversion of achromatic pixels and then enhancing of dominant colour in them, in this case, red colour. Hence a quicker and more robust method for colour detection was required.

The next chapter discusses the revision made to the software in the colour detection method and in the 'Check Ellipse' method to filter out rectangular signs.

7 An Improved Colour Segmentation Technique

This chapter discusses the shortcomings of the method proposed and proposes a revised version of the method proposed in chapter 5. First we discuss the main problems found in the design of the proposed method and then the following sections examine the new colour segmentation method proposed. Results are discussed at the end of this chapter.

7.1 A Review of the current technique

The review of the proposed method in previous chapter found problems with the design as mentioned below:

I. The design was adding extra noise based on the red colour in the actual grey achromatic pixels.

II. The method also was slow due to involvement of two colour spaces; HSV and RGB followed by RGB Normalisation on each pixel.

III. It was irregularly selecting achromatic zone as the zone model was initially based on HSV conical model that was applied to the HSV cylindrical model

IV. The achromatic region defined in the space model was quite simple and hence did not cover all the possible options of a red pixel falling in the achromatic and sub achromatic zones.

As shown in the previous chapter, these problems led to the inefficiency of the new system and hence the author had to redesign the system with refinements in the colour segmentation technique. In addition to the colour segmentation, a rectangle elimination test was also added to the 'ChechkEllipse' method, this has been shown in section 7.3.

7.2 The Improved Technique

This section describes a step by step guide on how the existing technique was revised for a better and more robust colour segmentation technique. The idea behind this was to be able to use just one colour space as opposed to two, to speed up the program

7.2.1 HSV Colour Model

To address the problems mentioned in the section 7.1 author had to reanalyse the zone model that was based on the HSV cylindrical colour space. As mentioned, the zone model that defined the achromatic zone was simple and thus caused problems in the program, pertaining to the pixels lying in the achromatic and the sub achromatic regions.

To be able to analyse and redesign the zone model, we studied the relation between hue (H), saturation (S) and value (V) with respect to the location of achromatic zone in the cylindrical shape. When visually observed the HSV cylindrical representation (shown below), the shape of achromatic region for all the different types of hue, was a curve at the centre, as visible below:

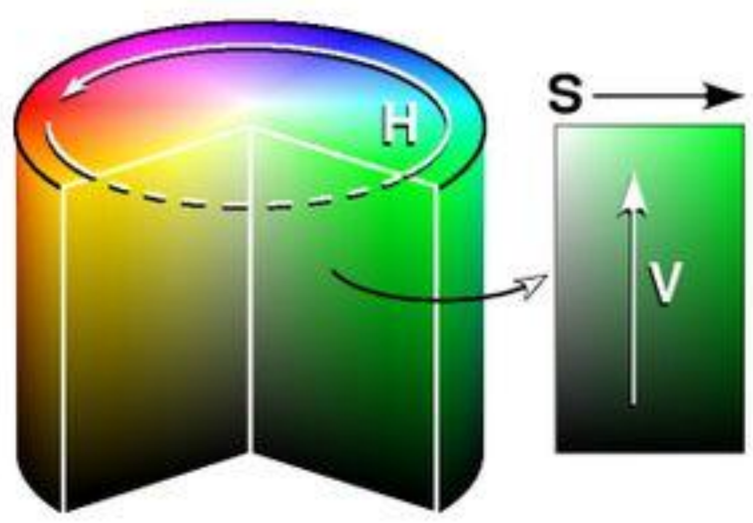
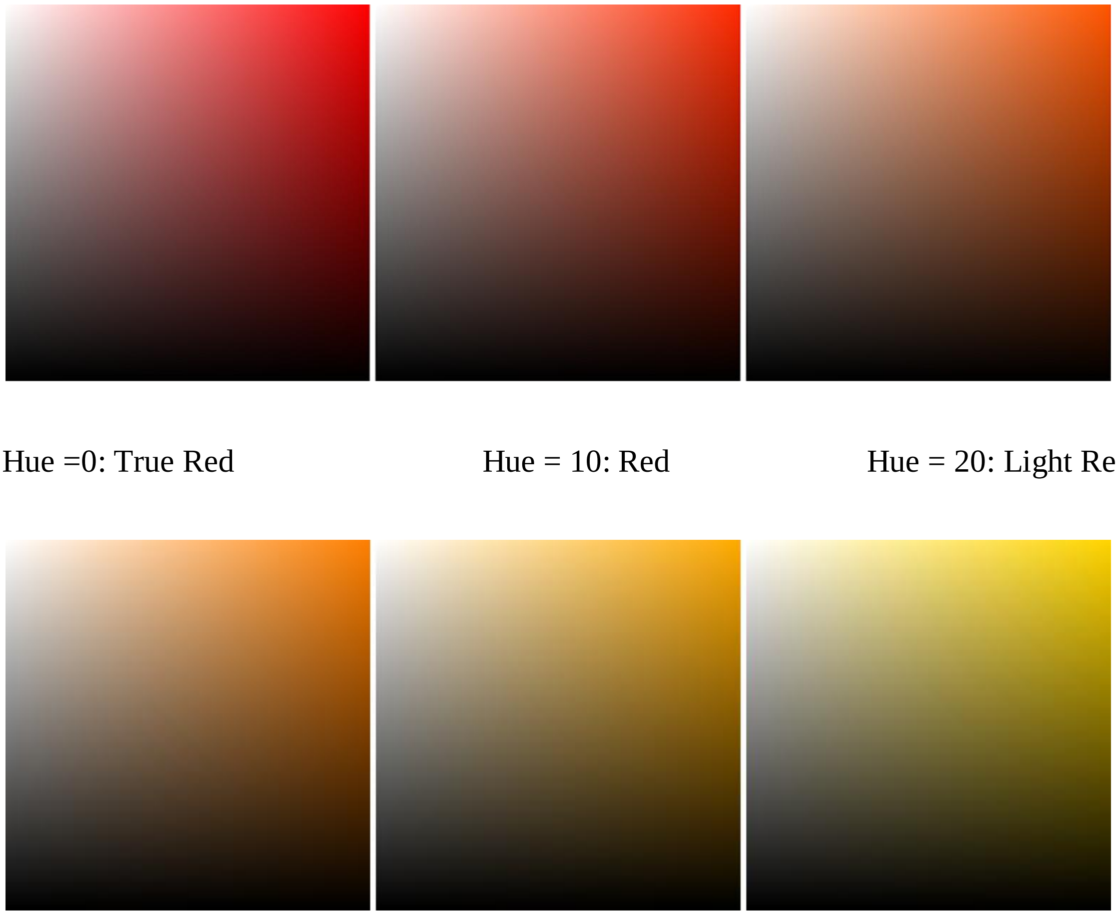


Figure 57 HSV colour space representation in an cylindrical model [106]

It was found that for the hue region located between 0 degrees and 30 degrees and 300 degrees to 360 degrees (related to colour red), the achromatic zone's shape was quite similar throughout. Hence, to further confirm the placement of the achromatic zone, we used the program to generate coloured patterns displaying saturation (S) vs. value (V) for different values of hue (H) on matrix of 9x9 and 256x256. These patterns were visually analysed for finding the line of demarcation between the achromatic and chromatic zones.

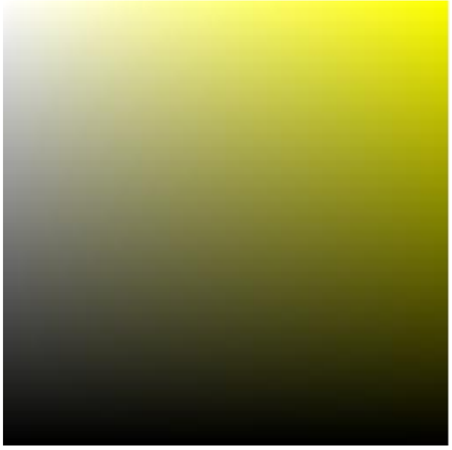
Images below, show the patterns (256x256 pixels) generated for hue with a variation of 10 degrees each, starting at hue being 0 and going up to 60 degrees in the cylinder to show yellowish-orange colour at 50 degrees and yellow at 60 degree (as shown in the cylindrical diagram for HSV colour space, all values in degrees).



Hue =0: True Red Hue = 10: Red Hue = 20: Light Red

Hue =30: Reddish Orange Hue=40: Orange Hue =50: Yellowish Orange

Figure 58 Saturation vs. Value spread, for different values of Hue



Hue = 60 degrees = Yellow

Figure 59 Saturation vs. Value for Hue= 60 degress, yellow

After the visual inspection of the generated patterns for all the seven hues (only red colour), the shape of the line demarcating the achromatic and chromatic regions was found to be a curve, thus changing the limits in the zone model and hence identifying a the cause of underperforming by the colour segmentation method as described in section 5.4.1. To make it simpler, we have shown below the curve, representing the line of demarcation between the two regions of definite achromatic and definite chromatic zones in hue spread at 0 degrees:

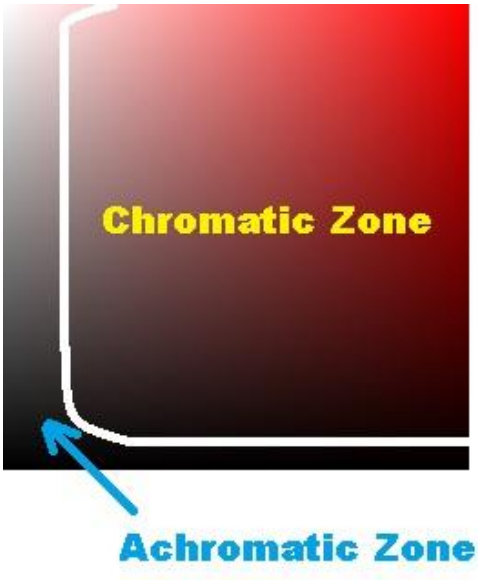


Figure 60 Definite achromatic and chromatic zones shown in hue spread at 0 degrees

Since, computing a curve using a computer program could use significant computational resources, the author derived a new shape closely related to the visually analysed curve in form of a polygon. The shape of the line of demarcation between the two regions is shown below, in the new zone model, defining only chromatic and achromatic zones:

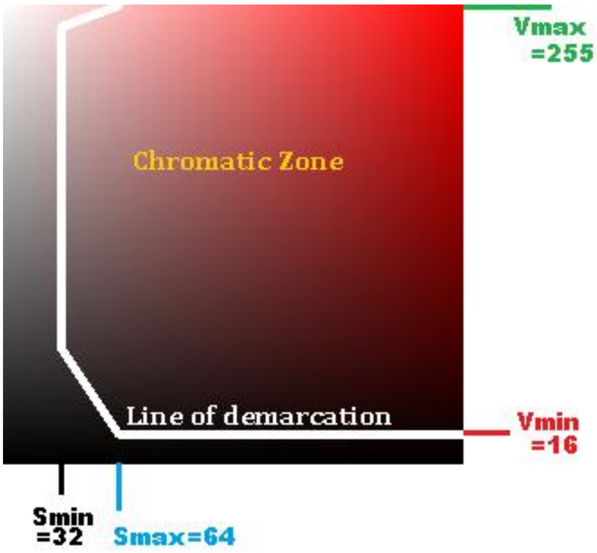


Figure 61 Image from the program showing the new line of demarcation and new zone model.

The minimum and maximum limits of this new line were initially kept same as proposed in the zone model except V_{max} which was changed to 255 from 243 i.e. $S_{max} = 64; S_{min} = 32; V_{min} = 16; V_{max} = 255$. As the curvy shape of the line of demarcation was changed into a polygon, there were two diagonal lines which represented the curve part (shown above) joining the maximum limit and minimum limit lines. It is clear by visual inspection, that the region below diagonal 1 would be black and the region above diagonal 2 would be black as well, as both these regions are achromatic regions.

These diagonals had their coordinates empirically selected after visual inspection of the S and V behaviour model (as described above). In addition, these limits were decided by vigorous testing of different kind of images, both provided in the database and from the internet. To be able decide whether a pixel was to be above or below a diagonal line, the following equation was derived using the two-point form equation of a line:

$$(S-S_1)/ (V-V_1) = (S_2-S_1)/ (V_2-V_1)\text{......................} (1)$$

Where (S_1, V_1) and (S_2, V_2) are two points of a line with $S_1 \neq S2$

Using (1), we get

$$(V_2 - V_1) * S + (S_1 - S_2) * V + V_1 * S_2 - S_1 * V_2 = 0\text{.........................} (2)$$

Hence for a point (S, V) to be on this line the equation (2) must be satisfied.

The expression in **(2)** can be evaluated for the S and V values of a pixel to decide whether the pixel is in the chromatic or achromatic zone. Shown below is the table that was used to decide which zone a pixel is in. White represents the chromatic zone and, and black represents the achromatic zone. A pixel is chromatic if it is not achromatic for all lines.

V_2-V_1	**S_1-S_2**	**> 0**	**< 0**
+	+	Chromatic	Achromatic
+	+	Chromatic	Achromatic
-	+	Achromatic	Chromatic
-	-	Achromatic	Chromatic

Horizontal and vertical lines are handled separately (before applying this table) because the calculations are simpler. This improves the speed of the program.

Table XII The deciding conditions for the points in the new zone model

Once the shape was created it was put in the program for testing. To speed up the program, the HSV coloured image was directly converted into a grey coloured image with only two pixel values, black or white (as shown in the table above). This eliminated the need for normalising a pixel and hence reduced the load on the program, thus increasing its speed.

A subject image after Gaussian smoothing was passed on to the colour segmentation method as an HSV image, where each of its pixel's location in the zone model was determined based on its S and V values. Once the position was determined in the zone model, the pixel was either turned white or black, in the new grey image.

It was also found that the value of hue used initially in the colour segmentation method was quite broad and hence it falsely detected yellowish-red colour signs as red. To reduce this effect and after visual analysis of the HSV colour space for red colour, the front range of H was reduced to 0- 20 degrees from 0-60 and increased to 270 – 360 degree from the back range of 300-360. This gave a more predominant red colour region while testing and hence again increased the robustness of the program.(Note that Open CV uses hue values from 0 to 180 instead of 0 to 360)

Shown below is the program code for the new colour segmentation method: Table XI is only applied to pixels with $S_{min}<S<S_{max}$ and $V_{min}<V<V_{max}$.

```
private Image<Gray, Byte> Method2_HSV_Achro_DominantColour(Image<Hsv,
byte> image)
        {
            LineSegment2D[] Lines =  // Points depend on each hue
            {   new LineSegment2D ( new Point (32,243), new Point
(64,255)),
                new LineSegment2D ( new Point (64,16), new Point
(32,64)),
            };
            Vmax = 255;
            Vmin = 16;
            Smax = 64;
            Smin = 32;
            // Normalise each pixel of the image and return a new
image
            for (int i = 0; i < width; i++)
            {
                for (int j = 0; j < height; j++)
                {

                    Gray = GreyImg_2[j, i].Intensity;
                    V = image[j, i].Value; // Checks the Brightness
                    S = image[j, i].Satuation; // Decides intensity of
the colour
                    H = image[j, i].Hue; // Decides the Colour
                    int s = (int)S;
                    int v = (int)V;
                    if (H > 10 && H < 135) // Check for not RED
                    {
                        GreyImg_2[j, i] = Black;
                    }
                    else
                    {// Check horizontal and vertical lines
                        if (Vmax < V || V < Vmin || S < Smin)
                            GreyImg_2[j, i] = Black;
                        else if (S > Smax)
                            GreyImg_2[j, i] = White;
                        else
                        {// use table XI
                            GreyImg_2[j, i] = White;
                            for (int n = 0; n < Lines.Length; n++)   //
number of lines
                            {
                                int S1 = Lines[n].P1.X;
                                int V1 = Lines[n].P1.Y;
                                int S2 = Lines[n].P2.X;
                                int V2 = Lines[n].P2.Y;
      if ((V2 - V1) > 0)
      {
         if ((((V2 - V1) * s) + ((S1 - S2) * v) + (V1 * S2 - S1 * V2))
< 0)
          GreyImg_2[j, i] = Black;
         }
      else
         {
          if ((((V2 - V1) * s) + ((S1 - S2) * v) + (V1 * S2 - S1 *
V2)) > 0)
          GreyImg_2[j, i] = Black;
         }
      }
        }
          }
           }
            }
  return GreyImg_2;
}
```

Code 14 New Colour segmentation method

Next section shows the second addition to the original program, to reduce the number of false detections based on shape detection. This is the implementation of rectangular shape elimination in the 'CheckEllipse' method. This method was designed to eliminate any four cornered contour, but has been kept simple and flexible, so that distorted ellipses are not eliminated and the overall speed of the program is not reduced significantly due to this check.

7.3 Rectangle Elimination

In the 'Check Ellipse' method, once a contour is found as an ellipse, it is tested for rectangular shape by checking the number of vertices and area of the contour. The contour is first approximated and then it is tested for number of vertices, if it has vertices less than 4 then it is considered not to be an ellipse.

```
if (isEllipse)
{
  Contour<Point> approxContour =
EllipseContour.ApproxPoly(EllipseContour.Perimeter * 0.030,
_hullStorage);
  if (approxContour.Area > 100 && approxContour.Total <= 4)
  {
      // Check For rectangle
      isEllipse = false;
  }
}
```

Code 15 Checking for the shape of the ellipse

7.4 Analysis of the results after modifications: HSV Achromatic method

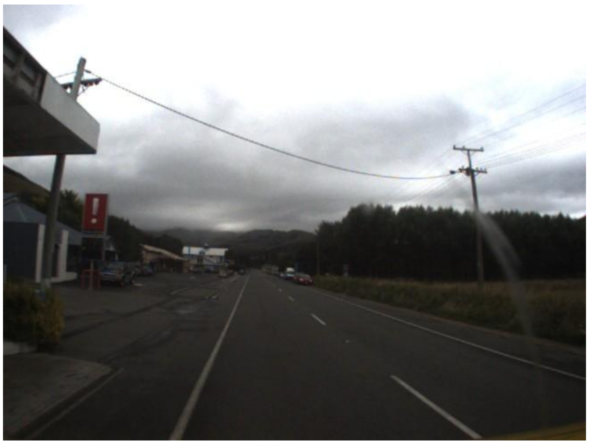
7.4.1 Results analysed

Image A: 2_517445.523: Rectangular Red Sign

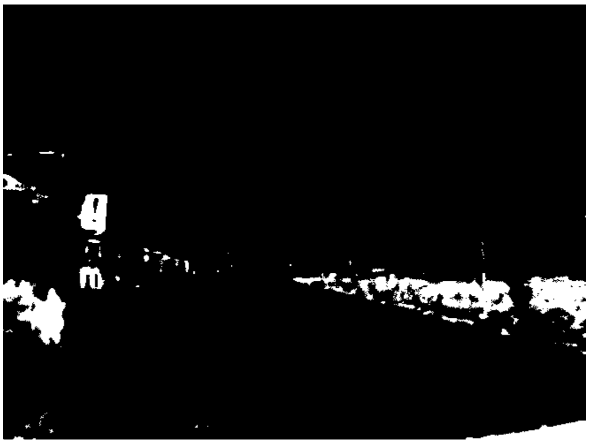
Method in which the image failed: Our Method, Method 2010 and Method 2008

Reason: Red rectangle being picked up as ellipse.

a) Original Image



b) Method 2008



c) Method 2010

d) Our Method

e) HSV Achrom

This is not detected as ellipse due to the introduction of rectangle elimination.

Image B: 1_508851.734: Yellow Circle and Green Circle

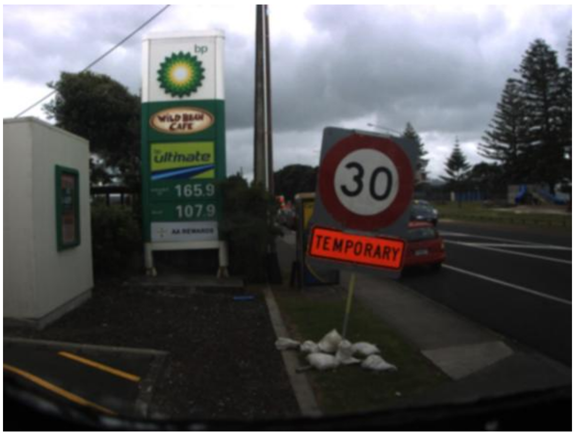
Method in which the image failed: Method 2010 and Method 2006

Reason:

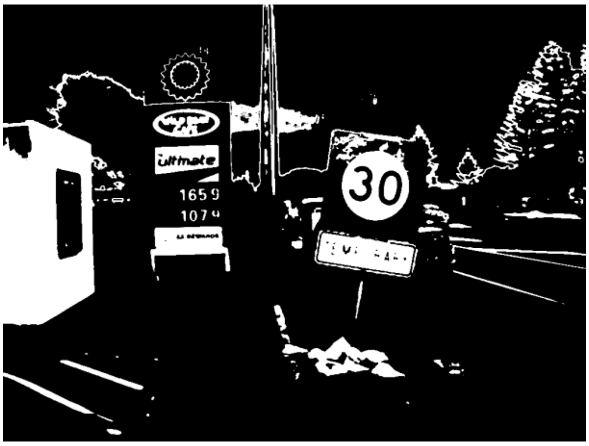
- Method 2010: Yellow circle in achromatic Zone being treated as a red circle due to use of threshold.

- Method 2006: Green circle is being treated as Red circle and misses the sign as the red circle is distorted due to noise on the actual sign.

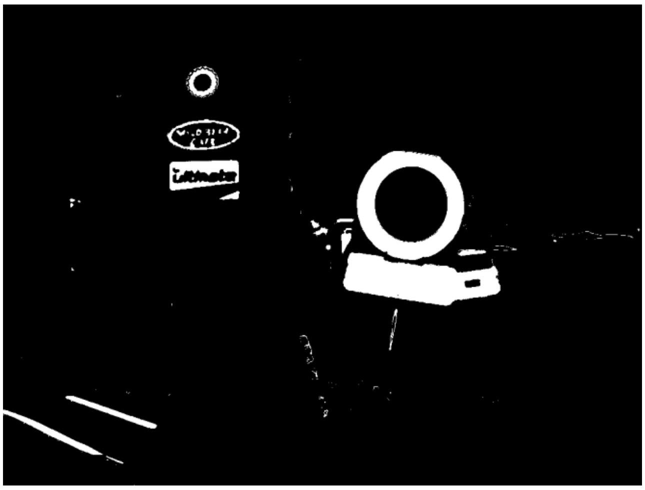
a) Original Image



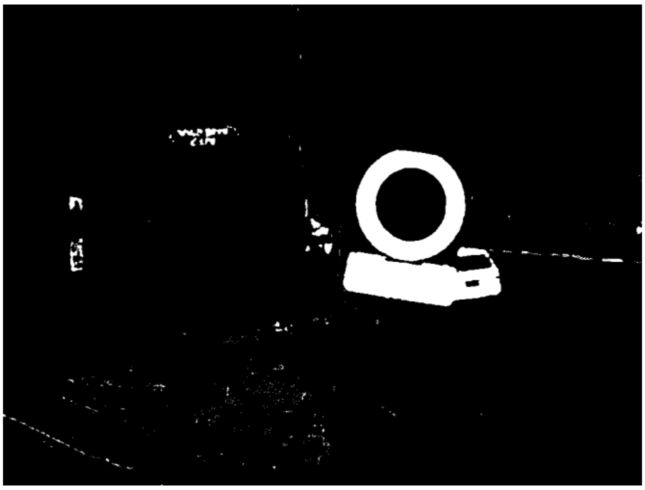
b) Method 2006



c) Method 2010



d) HSV Achrom



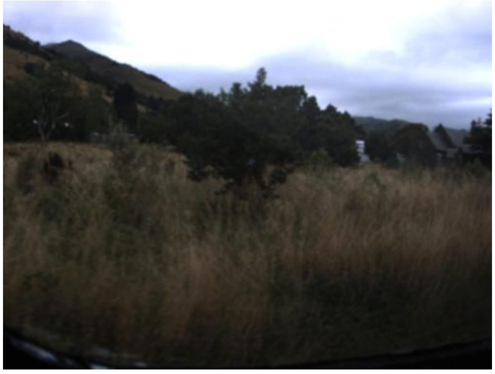
No circle detected due to new HSV achromatic zone model.

Image C: 1_517413.923: Noise in the Grass

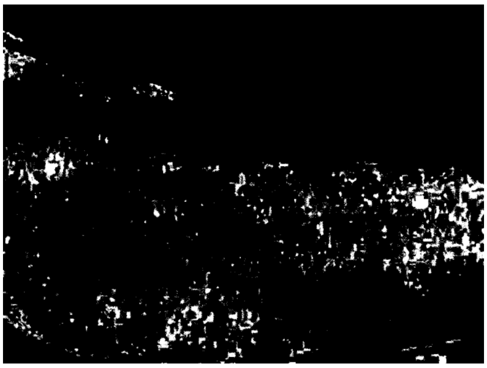
Method in which the image failed: Our Method

Reason: Added noise in the grass due to conversion of achromatic pixels into chromatic red.

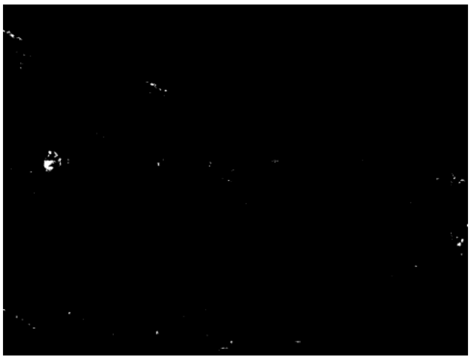
a) Original Image



b) Our Method



c) HSV Achrom



No added noise due to new achromatic zone model.

7.5 Analysis of the new modifications

Once the above mentioned modifications were applied in the program, it was tested with the initial set of images. The result for the new method (HSV Achrom), as compared to initial method (RGB Normalisation) and method 2008, has been given below in the table.

Method Name	**Time Taken (sec)**	**Correct Signs**	**Probable Signs**	**Missed Signs**	**Correct Percentage** ***(Rounded off)***	**False Detection**
HSV Achrom	848	21	13	3	92%	12
RGB Norm	2718	25	6	6	84%	10
Method 2008	1001	26	3	8	78%	18

Table XIII Results for the final run

The results above show a considerable increase in the speed as compared to the initial method and also increase in the correct percentage by around 8 %.

Next section will conclude the thesis and also discuss future work.

8 Conclusion and Future Work

8.1 Conclusion

The literature review of the colour segmentation methods revealed that there have been numerous experiments carried out for detection of traffic signs. Mainly these experiments have been conducted with generic pictures under different illumination conditions, but none of the reviewed methods has highlighted the effect of colour segmentation on dark and gloomy images.

The colour segmentation method consisting of the zone model in HSV cylindrical shape, described herein addresses the limitations of past and current experiments, when used for colour segmentation in traffic sign images. Firstly, the method proposed is invariant to different illumination conditions including poor lighting conditions due to weather conditions or poor quality images. The combination of zone model and the HSV colour space (our proposed method) while performing at comparable speeds, outperforms other proposed methods when used for detecting speed signs in images provided from company's database. Secondly, the method proposed successfully eliminates unwanted achromatic pixels while retaining useful chromatic pixels, thus providing good, clear figures for edge detection without any extra induced noise. The proposed colour segmentation method performs very well under poor illumination conditions and is invariant to shade and illumination. Thirdly, shape matching has been implemented in a more robust way, with the introduction of ellipse detection and rectangle elimination. Finally, we have successfully developed more complex software for detecting of NZ based speed signs, based on an open source library (Open CV). The software developed outperforms the sample program provided by Emgu CV community and has a high overall success rate of 92%, better than other methods in the literature review. The software performs at a satisfactory speed and has been tested using a sample data set of actual images provided by the company.

Though the colour segmentation method implemented using the HSV colour space instead of the RGB normalisation increased the speed of the program considerably, it might not be enough to replace a human being for testing images. It can certainly be used to reduce the database of images to possible sign images only, rather than requiring a person to view the entire image database. This newly developed software performs at optimal satisfactory speed, but the author feels there is some more room for improvement in this area.

The experimental results show that the new achromatic subspace defined for HSV cylindrical shape performs better than the other methods reviewed in the literature, but possibly a combination of RGB normalisation and HSV colour space could be a good option as well if it can be refined for extra noise that the combination adds to the images.

Even though the shape detection has been designed to eliminate non elliptical shapes, it still requires some improvement. This will be a more complicated process and will slow down the program; hence it is subject to future research.

Speed sign number matching has been briefly applied in the program, but due to the limitations of time it has been left for future research. Also no feature matching has been applied in the current program.

It could be possible that all the above problems are resolvable, and the work involved would be subject to future research.

8.2 Future Work

The method proposed, looks at each pixel for its location in the zone model, thus slowing down the program. Speed may be able to be increased by using some special techniques like split and merge algorithm, regions splitting method [100] ,T-pyramid [79], quad tree histogram method [107] or by using the hexagonal network structure as mentioned in [108].

Some more work can be done on shape detection, where ellipse detection can be made more accurate using ellipse inclined at angles. This can help in determining a correct ellipse. In addition, a more complicated algorithm could be developed for finding distorted shapes which appear to be ellipses.

Improvement in the sign detection can be made by checking the inside of a detected sign for numbers and white colour. This will make the program more robust and will reduce false detections.

SURF matching can be applied to match features of the sign with the base image and thus can help make the software more robust and less error prone.

It is hoped that more research in the field of shape detection, feature matching and optimising of the overall program can solve these problems efficiently, thus leading to superior image detecting software.

9 References

[1] Aryuanto, *et al.*, "Intelligent Machine Vision System for Road Traffic Sign Recognition," presented at the Prosiding Seminar Nasional Teknoin, 2008.

[2] V. S, *et al.*, "A Road Traffic Recognition System based on Template matching employing Tree classifier," presented at the International Conference on Computational Intelligence and Multimedia Applications, 2007.

[3] A. Ruta*, et al.,* "Real-timetraffic sign recognition from video by class-specific discriminative features," *Pattern Recognition,* vol. 43, pp. 416--430, 2009.

[4] C. Olson and D. Huttenlocher, "Automatic target recognition by matching oriented edge pixels," *IEEE Transactions on Image Processing,* vol. 6, pp. 103-113, 1997.

[5] S. Lu and A. Szeto, "Hierarchical Artificial Neural Networks for Edge Enhancement, Pattern Recognition," vol. 26, pp. 41149-1163, 1993.

[6] L. Fletcher*, et al.*, "Correlating Driver Gaze with the road scene for driver assistance systems," *science direct,* 2005.

[7] A. Ruta*, et al.,* "Traffic Sign recognition using discriminative local features," in *7th International Symposium on Intelligent Data Analyis*, 2007, pp. 355-366.

[8] M. O. Rahman, *et al.*, "Real Time Road Sign Recognition System Using Artificial Neural Networks for Bengali Textual Information Box," *European Journal of Scientific Research,* vol. 25, pp. 478-487, 2009.

[9] G. N. Limited. (2009, 18 August 2009). *About Us* [Web page]. Available: http://www.geosmart.co.nz/about

[10] S. J. Sangwine, "Colour in image processing," *Electronics & Communication Engineering Journal*, vol. 12, pp. 211-219, 2000.

[11] K. P. Ngoi and J. C. Jia, "A colour contrast model for edge and contour detection," in *TENCON '94. IEEE Region 10's Ninth Annual International Conference. Theme: Frontiers of Computer Technology. Proceedings of 1994*, 1994, pp. 309-313 vol.1.

[12] D. W. Ball, "The Baseline: Color," *Spectroscopy*, vol. 24, p. 16, 2009.

[13] B. A. Harrison and D. L. B. Jupp, *Introduction to image processing*. Canberra: Commonwealth Scientific and Industrial Research Organization, 1990.

[14] J. M.DiCarlo and J. C. Sabataitis. (1998, 1 November). *CIECAM97 Color Appearance Model*. Available: http://scien.stanford.edu/pages/labsite/1998/psych221/projects/98/ciecam/Project.html

[15] M. D. Fairchild, *Color Appearance Models*. Sussex: John Wiley & Sons, 2005.

[16] M. D. Fairchild, "Color Appearance Models: CIECAM02 and Beyond," in *IS&T/SID 12th Color Imaging Conference*, 2004.

[17] M. C. Stone, *A field guide to digital color*. Natick, Mass: AK Peters, 2003.

[18] A. Peck, "Color," in *Beginning GIMP*, ed: Apress, 2009, pp. 293-341.

[19] A. H. W. N. Sproson, Bristol, "Colour Science in Television and Display Systems," *Color Research & Application*, vol. 11, p. 221, 1983.

[20] H. J. Trussell, *et al.*, "Color image processing [basics and special issue overview]," *Signal Processing Magazine, IEEE*, vol. 22, pp. 14-22, 2005.

[21] Y. Ohta, *Knowledge-based interpretation of outdoor natural color scenes*. Marshfield, MA, USA: Pitman Publishing, Inc., 1985.

[22] G. Tonnquist, "Philosophy of perceptive color order systems," *Color Research & Application*, vol. 11, pp. 51-55, 1986.

[23] M. Strintzis, *et al.*, "Coding of Two-Dimensional and Three-Dimensional Color Image Sequences," in *Color Image Processing*, ed: CRC Press, 2006, pp. 503-523.

[24] H. Palus, "Color Image Segmentation," in *Color Image Processing*, ed: CRC Press, 2006, pp. 103-128.

[25] J. Sturges and T. W. A. Whitfield, "Salient Features of Munsell Colour Space as a Function of Monolexemic Naming and Response Latencies," *Vision Research*, vol. 37, pp. 307-313, 1997.

[26] A. H. Munsell. (20 April). *Munsell Color* [webpage]. Available: http://www.xrite.com/top_munsell.aspx

[27] J. Hongyong, *et al.*, "Testing of the Uniformity of Color Appearance Space," in *Computer Science and Information Engineering, 2009 WRI World Congress on*, 2009, pp. 307-311.

[28] A. P. Plummer, "Colour makes it easy," *Sensor Review*, vol. 10, pp. 117-120, July 1990.

[29] D. Pascale. (2003, 6th October). A Review of RGB Color Spaces.... from xyY to R'G'B'. 35. Available: http://www.babelcolor.com/download/A%20review%20of%20RGB%20color%20spaces.pdf

[30] (2011, 10th May 2011). *Color Model* [Web page]. Available: http://en.wikipedia.org/wiki/Color_model

[31] C.-H. C. Din-Chang Tseng "Color Segmentation Using Perceptual Attributes," in *11th IAPR International Conference*, 1992, pp. 228 - 231.

[32] W. Niblack, *An introductino to digital image processing*. Birkeroed, Denmark: Strandberg Publishing Company 1985.

[33] (2011, 9th April 2011). *RGB Color Model* [Web Page]. Available: http://en.wikipedia.org/wiki/RGB_color_model

[34] C. Poynton, *Digital Video and HDTV: Algorithms and Interfaces*. San Francisco: Morgan Kaufmann Publishers, 2003.

[35] C. Wen, *et al.*, "Identifying Computer Graphics using HSV Color Model and Statistical Moments of Characteristic Functions," in *Multimedia and Expo, 2007 IEEE International Conference on*, 2007, pp. 1123-1126.

[36] E. C. Carter, "The reproduction of colour, 5th edition, by R. W. G. Hunt," *Color Research & Application*, vol. 23, pp. 116-117, 1998.

[37] J. O. Y.Wang, and Y. Zhang, *Video Processing and Communications*. NJ: Prentice Hall, 2002.

[38] M. Burge*, et al.*, *Principles of digital image processing: fundamental techniques* vol. 7592. London: Springer, 2009.

[39] Wikipedia. (2010, 30th August 2010). *RGB Color Space* [Web Page]. Available: http://en.wikipedia.org/wiki/RGB_color_spaces

[40] C. Lin and C.-H. Su, "Colour image segmentation using the relative values of RGB," presented at the Proceedings of the 9th WSEAS international conference on Applications of computer engineering, Penang, Malaysia, 2010.

[41] (2001, 15 August 2010). *NTSC and its properties* [Web page]. Available: http://sepwww.stanford.edu/public/docs/sep110/nick1/paper_html/node2.html

[42] J. F. Blinn, "NTSC: nice technology, super color," *Computer Graphics and Applications, IEEE,* vol. 13, pp. 17-23, 1993.

[43] (2011, 12th April). *YIQ* [Web Page]. Available: http://en.wikipedia.org/wiki/YIQ

[44] M. A. Michael Stokes, Srinivasan Chandrasekar , Ricardo Motta. (1996). *A Standard Default Color Space for the Internet - sRGB (1.10 ed.)*. Available: http://www.w3.org/Graphics/Color/sRGB.html

[45] (2011, 30th May 2011). *Color Triangle* [Web Page]. Available: http://en.wikipedia.org/wiki/Color_triangle

[46] T. Smith and J. Guild, "The C.I.E. colorimetric standards and their use," *Transactions of the Optical Society,* vol. 33, 4 March 1932 1931.

[47] W. D. Wright, "A re-determination of the trichromatic coefficients of the spectral colours," *Transactions of the Optical Society,* vol. 30, p. 141, 1929.

[48] J. Guild, "The Colorimetric Properties of the Spectrum," in *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*. vol. 230, ed: The Royal Society, 1932, pp. 149-187.

[49] M. C. Stone, "Representing colors as three numbers [color graphics]," *Computer Graphics and Applications, IEEE,* vol. 25, pp. 78-85, 2005.

[50] G. H. Joblove and D. Greenberg, "Color spaces for computer graphics," presented at the Proceedings of the 5th annual conference on Computer graphics and interactive techniques, 1978.

[51] N. Herodotou, *et al.*, "A color segmentation scheme for object-based video coding," in *Advances in Digital Filtering and Signal Processing, 1998 IEEE Symposium on,* 1998, pp. 25-29.

[52] A. Hanbury, "The Taming of the Hue, Saturation and Brightness Colour Space," presented at the In Proceedings of the 7th CVWW, 2002.

[53] J. Rus. (2010, 12th December 2010). *HSL and HSV* [Image]. Available: http://en.wikipedia.org/wiki/File:Hsl-hsv_models.svg

[54] H. Fritsch. (2010, 18 June 2011). *Building a color world map*. Available: http://itooktheredpill.dyndns.org/2010/building-a-color-world-map/

[55] M. I. Vardavoulia, *et al.*, "Vector ordering and morphological operations for colour image processing: Fundamentals and applications," *PATTERN ANALYSIS AND APPLICATIONS,* vol. 5, pp. 271-287, 2002.

[56] K. Cho, "Adaptive skin-color filter," *Pattern Recognition*, vol. 34, pp. 1067-1073, 2001.

[57] A. R. Smith, "Color gamut transform pairs," *SIGGRAPH Comput. Graph.*, vol. 12, pp. 12-19, 1978.

[58] M. K. Nikos Koutsias, and Emlllo Chuvieco, "The Use of Intensity-Hue-Saturation Transformation of Landsat-5 Thematic Mapper Data for Burned Land Mapping," *PHOTOGRAMMETRIC ENGINEERING AND REMOTE SENSING*, vol. 66, pp. 829-840, July 2000.

[59] J. J. Báez Rojas and M. A. Alonso Pérez, "Conversion from n bands color space to HSI n color space," *Optical Review*, vol. 16, pp. 91-98, 2009.

[60] J. Zhang*, et al.*, "Color image segmentation in HSI space for automotive applications," *Journal of Real-Time Image Processing*, vol. 3, pp. 311-322, 2008.

[61] N. Moroney*, et al.*, "The CIECAM02 Color Appearance Model," presented at the IS&T/SID 10 th Color Imaging Conference, 2002.

[62] O. Tulet*, et al.*, "Image Rendering Based on a Spatial Extension of the CIECAM02," in *Applications of Computer Vision, 2008. WACV 2008. IEEE Workshop on*, 2008, pp. 1-6.

[63] H. Fleyeh, "Shadow And Highlight Invariant Colour Segmentation Algorithm For Traffic Signs," in *Cybernetics and Intelligent Systems, 2006 IEEE Conference on*, 2006, pp. 1-7.

[64] S. Vitabile*, et al.*, "Road signs recognition using a dynamic pixel aggregation technique in the HSV color space," in *Image Analysis and Processing, 2001. Proceedings. 11th International Conference on*, 2001, pp. 572-577.

[65] S. D. Buluswar and B. A. Draper, "Color recognition in outdoor images," in *Computer Vision, 1998. Sixth International Conference on*, 1998, pp. 171-177.

[66] S. D. Buluswar, "non-parametric classification of pixels under varying outdoor illumination," in *In Proceedings: Image Understanding Workshop*, ed: ARPA, Morgan Kaufmann, 1994, pp. 1619--1626.

[67] S. Vitabile*, et al.*, "A neural network based automatic road signs recognizer," in *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on*, 2002, pp. 2315-2320.

[68] L. King Hann*, et al.*, "New hybrid technique for traffic sign recognition," in *International Symposium on Intelligent Signal Processing and Communications Systems, 2008. ISPACS 2008.* , 2009, pp. 1-4.

[69] A. de la Escalera*, et al.*, "Road traffic sign detection and classification," *Industrial Electronics, IEEE Transactions on*, vol. 44, pp. 848-859, 1997.

[70] D. Ghica*, et al.*, "Recognition of traffic signs by artificial neural network," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, 1995, pp. 1444-1449 vol.3.

[71] Y. Wang*, et al.*, "Traffic Signs Detection and Recognition by Improved RBFNN," in *Computational Intelligence and Security, 2007 International Conference on*, 2007, pp. 433-437.

[72] L. Itti, *et al.*, "A model of saliency-based visual attention for rapid scene analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 20, pp. 1254-1259, 1998.

[73] G. Xiaohong, *et al.*, "Colour Vision Model-Based Approach for Segmentation of Traffic Signs," *EURASIP Journal on Image and Video Processing,* vol. 2008, pp. 1-8, 2008.

[74] W. G. Shadeed, *et al.*, "Road traffic sign detection in color images," in *Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003 10th IEEE International Conference on*, 2003, pp. 890-893 Vol.2.

[75] D. B. Judd, *et al.*, "Spectral Distribution of Typical Daylight as a Function of Correlated Color Temperature," *J. Opt. Soc. Am.,* vol. 54, pp. 1031-1036, 1964.

[76] P. Jainski, *et al.*, "Recommendations for Surface Colours for Visual Signalling," International Committee on Illumination / Commissino Internationale de I'Eclairage CIE 039.2-1983, 1983.

[77] H. Fleyeh, "Color detection and segmentation for road and traffic signs," in *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, 2004, pp. 809-814.

[78] X. Wang, "Modelling of Colour Appearance," Ph.D., Loughborough University, Leics,UK, 1994.

[79] M. Sonka, *et al.*, *Image Processing: Analysis and Machine Vision*. London: Thompson Computer Press, 1996.

[80] H. Gomez-Moreno, *et al.*, "Goal Evaluation of Segmentation Algorithms for Traffic Sign Recognition," *Intelligent Transportation Systems, IEEE Transactions on,* vol. 11, pp. 917-930, 2010.

[81] S. Maldonado-Bascon, *et al.*, "Road-Sign Detection and Recognition Based on Support Vector Machines," *Intelligent Transportation Systems, IEEE Transactions on,* vol. 8, pp. 264-278, 2007.

[82] P. G. Jime´nez, *et al.*, "Traffic sign shape classification and localization based on the normalized FFT of the signature of blobs and 2D homographies," *Signal Process.,* vol. 88, pp. 2943-2955, 2008.

[83] K. N. Plataniotis and A. N. Venetsanopoulos, *Color Image Processing and Applications*: Springer-Verlag, 2000.

[84] H. D. Cheng, *et al.*, "Color image segmentation: advances and prospects," *Pattern Recognition,* vol. 34, pp. 2259-2281, 2001.

[85] H. Kamada, *et al.*, "A compact navigation system using image processing and fuzzy control," in *Southeastcon '90. Proceedings., IEEE*, 1990, pp. 337-342 vol.1.

[86] R. Janssen, *et al.*, "Hybrid Approach For Traffic Sign Recognition," in *Intelligent Vehicles '93 Symposium*, 1993, pp. 390-395.

[87] 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. 57-68, 2004.

[88] M. D. Heath, *et al.*, "A robust visual method for assessing the relative performance of edge-detection algorithms," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 19, pp. 1338-1359, 1997.

[89] O. Community. (2006, 01 July 2010). *Full OpenCV Wiki* [Online Web page]. Available: http://opencv.willowgarage.com/wiki/

[90] G. Agam. (2006, 15th August). *Introduction to programming with OpenCV*. Available: http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/index.html

[91] EMGU. (2011, 10 May). *Emgu CV Main Page* [Website]. Available: http://www.emgu.com/wiki/index.php/Main_Page

[92] R. Fisher*, et al.* (2003, 14th June). *Gaussian Smoothing*. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm

[93] W. E. Green. (2002, 14th June). *Canny Edge Detection Tutorial*. Available: http://www.pages.drexel.edu/~weg22/can_tut.html

[94] R. Keshet. (2008, 9th September 2010). *Dilation* [Image]. Available: http://en.wikipedia.org/wiki/File:Dilation.png

[95] R. Fisher*, et al.* (2004, 14th June). *Erosion* [Online Tutorial]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/erode.htm

[96] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.,* vol. 8, pp. 679-698, 1986.

[97] M.-K. Hu, "Visual pattern recognition by moment invariants," *Information Theory, IRE Transactions on,* vol. 8, pp. 179-187, 1962.

[98] J. Shutler. (2002, 12th December). *Hu invariant set* [Webpage]. Available: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/SHUTLER3/node8.html

[99] L. Ling and L. Xiying, "A Method to Search for Color Segmentation Threshold in Traffic Sign Detection," in *Image and Graphics, 2009. ICIG '09. Fifth International Conference on*, 2009, pp. 774-777.

[100] R. Ohlander*, et al.*, "Picture segmentation using a recursive region splitting method," *Computer Graphics and Image Processing,* vol. 8, pp. 313-333, 1978.

[101] S. Lafuente-arroyo*, et al.*, "S. Lafuente-arroyo and P. García-díaz and F. j. Acevedo-rodríguez and P. Gil-jiménez and S. Maldonado," in *Advanced Concepts for Intelligent Vision Systems*, Brussels, Belgium, 2004.

[102] Y.-I. Ohta*, et al.*, "Color Information for Region Segmentation," *Computer Graphics and Image Processing,* vol. 13, pp. 222-241, July 1980.

[103] N. T. Agency. (2010, 15th August). *Speed Limit Standard 40 km/h sign* [Traffic Sign Specification]. Available: http://www.nzta.govt.nz/resources/traffic-control-devices-manual/sign-specifications/sign-display-page.html?CatID=30&ID=4

[104] I. eFunda. (2011, 15th June 2011). *The Method of Least Squares* [Web Page]. Available: http://www.efunda.com/math/leastsquares/leastsquares.cfm

[105] E. W. Weisstein. (15th June). *Least Square Fitting*. Available: http://mathworld.wolfram.com/LeastSquaresFitting.html

[106] M. Aubry*, et al.* (2005, 26th January). *FrozenCameleon*. Available: http://www.virtual-drums.com/frozen-cameleon.php

[107] X. W. Gao, *et al.*, "Recognition of traffic signs based on their colour and shape features extracted using human vision models," *Journal of Visual Communication and Image Representation,* vol. 17, pp. 675-685, 2006.

[108] D. Burdescu, *et al.*, "A New Method for Segmentation of Images Represented in a HSV Color Space," in *Advanced Concepts for Intelligent Vision Systems*. vol. 5807, J. Blanc-Talon, *et al.*, Eds., ed: Springer Berlin / Heidelberg, 2009, pp. 606-617.