REAL-TIME TRAFFIC SIGN DETECTION AND RECOGNITION ON FPGA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HÜSEYİN YALÇIN

IN PARTIAL FULLFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2013

Approval of the thesis:

REAL-TIME TRAFFIC SIGN DETECTION AND RECOGNITION ON FPGA

submitted by **Hüseyin YALÇIN** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan ÖZGEN
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Gönül Turhan SAYAN
Head of Department, **Electrical and Electronics Eng.** _____

Assoc. Prof. Dr. Mehmet Mete BULUT
Supervisor, **Electrical and Electronics Eng. Dept.,METU** _____

Prof. Dr. Gözde Bozdağı Akar
Co-Supervisor, **Electrical and Electronics Eng. Dept.,METU** _____

Examining Committee Members:

Prof. Dr. Aydın Alatan
Electrical and Electronics Eng. Dept.,METU _____

Assoc. Prof. Dr. Mehmet Mete Bulut
Electrical and Electronics Eng. Dept.,METU _____

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Eng. Dept.,METU _____

Assoc. Prof. Dr. Cüneyt Bazlamaçcı
Electrical and Electronics Eng. Dept.,METU _____

M.Sc. Hasan IRMAK
REHIS/TTD-AGTM, ASELSAN INC. _____

Date: 05.09.2013

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: Hüseyin Yalçın

Signature:

ABSTRACT

Yalçın, Hüseyin
M. Sc., Department of Electrical and Electronics Engineering
Supervisor: Assoc. Prof. Dr. Mehmet Mete Bulut
Co-Supervisor: Prof. Dr. Gözde Bozdağı Akar

September 2013, 78 pages

In this thesis, an embedded system for traffic sign detection and recognition is proposed. Proposed system is first designed in MATLAB and optimized. After optimization process, system design is transferred to FPGA and Virtex-V FX70 FPGA is selected for implementation platform.

640x480 sized image in RGB format is sent to FPGA system via computer interface. This image is segmented for red, blue, and yellow colors. Red and blue color maps are divided into 8x8 sub-blocks. Yellow color map is divided into 32x32 sub-blocks. After that, blocks which have more than 30 % valid pixels are marked as valid. Red and blue maps are converted into 80x60 and blobs are detected in these maps. Size of the detected blob is examined first and appropriate sized regions are marked as valid. Next, to be marked as traffic sign, valid regions are expected to meet one of the three conditions: Vertical neighboring, yellow sign neighboring and pole detection below the region.

Regions marked as traffic sign are interpolated to 60x60. Color segmentation operation is applied to these regions and edges are extracted in segmented images. Hough Transform is applied to edge images and contour information is extracted. Finally, Informative Pixel Percentage (IPP) operation is applied to inner region of the signs and results are obtained.

FPGA system is composed of two sub-parts: Processor system and hardware (logic) system. Color segmentation, region detection and pole detection modules are implemented in hardware system. Rest of the operations is implemented in the processor system.

To test success of the implemented system, 137 images, which include 162 traffic signs, are sent to this system and 90.1 % detection and 90.4 % recognition rates are achieved.

Keywords: FPGA, Embedded Processor, Color Segmentation, Traffic Sign Detection, Traffic Sign Recognition, Real-Time.

ÖZ

Yalçın, Hüseyin
Yüksek Lisans., Elektrik Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Mehmet Mete Bulut
Ortak Tez Yöneticisi: Prof. Dr. Gözde Bozdağı Akar

Eylül 2013, 78 sayfa

Bu tezde, trafik işareti bulma ve tanımlama için gömülü bir sistem önerilmiştir. Önerilen sistem önce MATLAB üzerinde tasarlanıp optimize edilmiştir. Optimizasyon işleminden sonra sistem tasarımı FPGA'e aktarılmıştır ve gerçekleme ortamı olarak Virtex-V FX70 model FPGA seçilmiştir.

640x480 boyutunda RGB formatındaki görüntü, bilgisayar arayüzü aracılığıyla FPGA sistemine gönderilmektedir. Bu görüntü kırmızı, mavi ve sarı renkler için sınıflandırılmaktadır. Kırmızı ve mavi renk haritaları 8x8'lik bloklara, sarı renk haritasıysa 32x32'lik bloklara ayrılır. Bu bloklarda bulunan piksellerin en az % 30'u uygun renkteyse, blok geçerli olarak işaretlenir. Bu işlem sonrasında mavi ve kırmızı renk haritaları 80x60 boyutuna indirilmiş olur ve bu haritalarda bölge tespiti yapılır. Bulunan bölgelerin önce boyutları incelenir ve uygun boyuttaki bölgeler geçerli olarak işaretlenir. Daha sonra geçerli bölgelerin, trafik işareti olarak işaretlenmesi için üç şarttan birini sağlanması beklenmektedir: Sarı işarete komşu olması, dikeyde başka bir bölgeye komşu olması ve altında işaret direği tespit edilmesi.

Trafik işareti olarak işaretlenen bölgeler 60x60 boyutuna boyutlandırılır ve bu bölgelere sırasıyla renk sınıflandırma ve kenar oluşturma işlemleri uygulanır. Kenar görüntüleri üzerine Hough Dönüşümü uygulandıktan sonra şeklin kenar bilgisi elde edilir. Son olarak, şeklin iç bölgesinde Bilgisel Piksel Yüzdesi (BPY) yüzdesi uygulanarak sonuçlar elde edilir.

FPGA sistemi iki bölümden oluşmaktadır: İşlemci sistemi ve donanım (mantık kapıları) sistemi. Renk ayrıştırma, bölge bulma ve direk bulma işlemleri donanım üzerinde yapılırken, geri kalan işlemler işlemci üzerinde yapılmaktadır.

Gerçeklenen sistemi test etmek için 162 trafik işareti içeren 132 görüntü sistem tarafından işlenmiştir. Bu görüntüler üzerinde % 90,1 bulma, %90,4 tanımlama başarısına ulaşılmıştır.

Anahtar Kelimeler: FPGA, Gömülü İşlemci, Renk Ayrıştırma, Trafik İşareti Bulma, Trafik İşareti Tanımlama, Gerçek Zamanlı.

To My Family and "Zuzu"

ACKNOWLEDGEMENTS

First of all, I wish to express my deepest thanksgiving to my supervisors Assoc. Prof. Dr. Mehmet Mete Bulut and Prof. Dr. Gözde Bozdağı Akar for their valuable critics, excellent supervision and endless support.

I would like to thank ASELSAN Inc. for facilities provided for the completion of this thesis.

I would like to express my thanks especially to Yağmur Demircan, Hasan Irmak and all my friends for their support and fellowship.

I would also like to thank TÜBİTAK-BİDEB for their financial support during my graduate education.

I would like to express my special appreciation to my family, Gülsüm Yalçın, Ali Yalçın and Onur Yalçın, for their continuous support and encouragements.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

Starting from the last decade, several innovations in computer, mechanical and material science are made. These advancements are also reflected to automotive industry. As a result, high engine efficiency and improved passenger safety have become common properties in new generation cars. To improve safety of passengers, two main approaches are followed [1],[2]. First approach is to ease damages of an accident, named as passive safety. For this purpose, number of airbags is increased and chassis designs are made accordingly. Second approach is to decrease probability of accidents by taking extra measures, named as active safety. In order to achieve this approach, Advanced Driver Assistance Systems (ADAS) are employed. Following systems can be given as examples of ADAS:

- Lane Following Systems
- Blind Spot Warning Systems
- Automated Parking Assistance Systems
- Predictive Emergency Braking System
- Traffic Sign Recognition Systems

According to the report of Republic of Turkey General Directorate of Highways (August, 2012) [3], 90.2% of traffic accidents occurred due to driver faults in 2011. This report shows that the best way of saving passengers and decreasing number of traffic accidents is to increase the awareness of drivers by tolerating human factor. Therefore, ADAS have a great importance for decreasing traffic accidents. Furthermore, it seems that more than half of these accidents occurred due to the traffic rule violations, such as speed limit and priority rule violations, etc. These statistics can be interpreted that increasing awareness of drivers by the aid of Traffic Sign Recognition (TSR) systems can provide sharp decrease in number of traffic accidents.

1.1 Background of Traffic Sign Detection and Recognition

Since the beginning of 2000's, traffic sign recognition has become a popular topic in computer vision. Therefore, several works are published about this issue. When these works are examined, it seems that some common approaches are widely used. One of the most preferred methods is to deal with the detection and recognition of traffic signs, separately. For detection operation, two basic approaches are suggested. When operation speed is the main concern, color based methods are preferred [4][5][6][7][8][9][10][11][12][13][14]. On the other hand, illumination condition is

considered as reliability problem in some works, and shape based detection methods are used [15][16][17][18][19].

Traffic signs are designed such that they can easily distinguished by drivers. Therefore, they have attractive colors like blue, red, yellow [20]. The incidence of these colors' occurrence is considerably low on roads; therefore, applying color segmentation to a scene can eliminate irrelevant regions. As a result, computation time can be reduced, significantly. For segmentation purpose, mostly RGB [4][8][12] and HSI/HSV [7][9][10][13] color spaces are preferred. On the other hand, spaces like L*a*b* [11] and YC_BC_R [21] are also employed. In the literature, two parameters affect preference of color space. First one is effect of changing illumination, and the other one is conversion time of one space to another.

Among the color spaces stated above, the most sensitive color space to illumination variations among the color spaces stated above is given as RGB color space. In [5], this phenomenon is explained with RGB sensor characteristics. These characteristics can be converged to form given in equation 1, 2, and 3.

$$C_R = E.M_b(n,s) * k_R \tag{1}$$

$$C_G = E.M_b(n,s) * k_G \tag{2}$$

$$C_B = E.M_b(n,s) * k_B \tag{3}$$

In these equations, C_R, C_G and C_B are sensor responses for red, green and blue colors, respectively. Illumination intensity is given with *e*, and k_R, k_G and k_B are sensor responses and surface albedo (reflection coefficient), $m_b(n,s)$ defines surface orientation and illumination direction. According to the presented information, above, it seems that illumination and orientation of shape affect R, G and B values of sensor output. Although this variation is a disadvantage for a detection system, problem is solved to some extend using relative values of R, G and B components [22], [14]. According to equation 4, using relative values eliminates *e* and $m_b(n,s)$ components so more reliable color information from RGB spaces can be obtained.

$$\frac{C_R}{C_B} = \frac{E.M_b(n,s) * k_R}{E.M_b(n,s) * k_B} => \frac{C_R}{C_B} = \frac{k_R}{k_B} \tag{4}$$

Benallal and Meunier [23] also experimented that R,G and B values of a red traffic sign changes hour to hour. However in day time, relative value of red component is always dominant. In Figure 1.1 samples taken with intervals of 30 minutes are presented. According to these data, between 6:30 A.M. (x=13) and 8:30 P.M. (x=41), the relative values of color component are highly appropriate for segmentation.

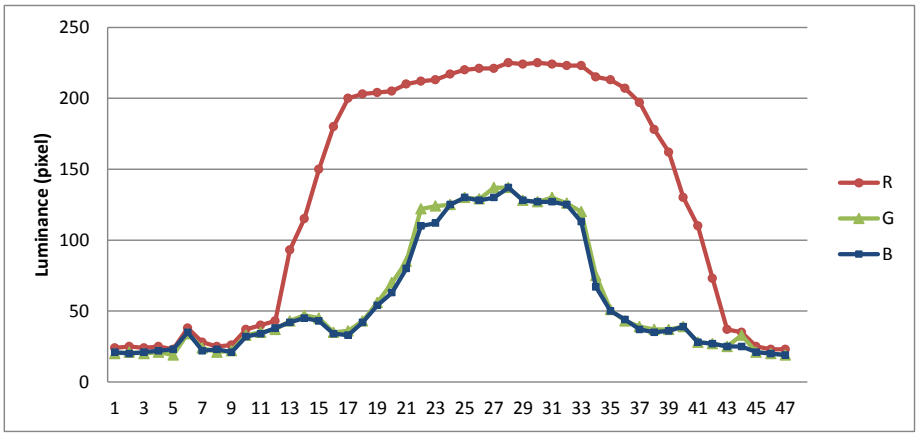

Figure 1.1: Change of R, G and B components of a red traffic sign through hours[23]

These enhancements make RGB color space more suitable for segmentation than it actually is. However, the most important reason of its preference is to get rid of computation burden of conversion [24]. On the other hand, Hue Saturation Intensity (HSI) color space shows better performance against illumination changes [25]. Therefore, it is widely preferred in sign detection applications. In Figure 1.2, a color cylinder of Hue Saturation Value (HSV) color space is presented. In this color space, hue describes dominant wavelength, in other words color. Saturation describes dominance of hue component. Finally, value describes lightness of color. Therefore, hue value is independent of illumination changes and highly preferable for outdoor color segmentation.

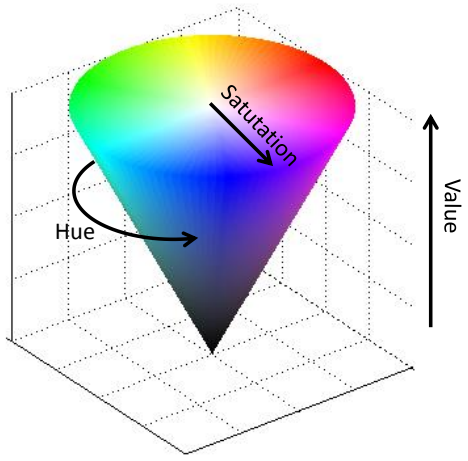

Figure 1.2: Color cylinder of HSV color space

On the other hand, most of the imaging devices give output either in RGB or YC_BC_R format. Therefore, a conversion must be applied to initial image in order to obtain it in HSV format. Conversion function from RGB to HSV is given in the equation 5.

$$
\begin{aligned}
&\mathrm{V} = \max(\mathrm{R}, \mathrm{G}, \mathrm{B}) \\
&\mathrm{S} = \begin{cases} [\mathrm{V} - \min(\mathrm{R}, \mathrm{G}, \mathrm{B})], \text{if } \mathrm{V} \neq 0 \\ 0, \qquad\qquad \text{else} \end{cases} \\
&\mathrm{S} = \begin{cases} (\mathrm{G} - \mathrm{B}) * 60/\mathrm{S}, \text{if } \mathrm{V} = \mathrm{R} \\ 180 + (\mathrm{B} - \mathrm{R}) * 60/\mathrm{S}, \text{if } \mathrm{V} = \mathrm{G} \\ 240 + (\mathrm{R} - \mathrm{G}) * 60/\mathrm{S}, \text{if } \mathrm{V} = \mathrm{B} \end{cases} \\
&H = H + 360, if\ H < 0
\end{aligned}
\tag{5}
$$

In eqn. 5, an iterative approach is given and division operations exist. Division, operation is a costly logical operation for both CPUs and FPGAs. In Core i5 processor, division latency is given as at least 20-25 cycles [26]. If FPGA is considered, in [7] division operation is used for RGB to HSV conversion and its latency is 17 cycles. Therefore, requirement of several division operations for conversion of each pixel can threaten real time operation.

In order to enhance color under different illumination conditions, white balancing is also employed. In [8], road is used as reference point. It is assumed that road should be grey and characteristic of light source can be obtained from its variation. Parameters are obtained calculating color deviation of road and these correction parameters are applied to whole image.

Instead of white balancing, reflectance calculation is preferred in [4] for colors correction. The original image is downsampled, Gaussian filtered and given a reflectance processor as input. After that, reflectance is obtained and each component of RGB space is adjusted, accordingly.

As number of operations increase, better segmentation results can be achieved. However, each additional process increases the latency, therefore, trade-offs must be considered for color segmentation operation.

Although processes given above make enhancements, under adverse illumination and weather conditions color based methods are not reliable. Therefore, shape based detection methods are preferred to detect sign regions[15][16][17][18][27][19][28].

As a common approach, algorithms are applied to obtain edge information. Then different feature extraction techniques are used to detect closed form shapes in obtained edge information.

In order to obtain edge information, [18] used sobel filter on grayscale image. Applying fast radial symmetry detector to edge information, circular speed signs are searched in the input image. Besides applying fast radial symmetry transform to edge information, Hough transform is also used [28][19][16]. Applying circular or linear Hough transform, circular and rectangular signs can be obtained.

In [19], false detections are not considered as a problem and it is proposed that false detections can be eliminated in recognition process. On the other hand, [28] employed Scale Invariant Feature Transform (SIFT) descriptor besides Hough transform. In this work, SIFT is applied to input image and obtained descriptors are evaluated with nearest neighbor. After evaluation, results of this stage combined with results of Hough Transform at refinement stage and highly reliable results are obtained.

Besides edge information, gradient information is also used to detect signs. Keller et al. calculated normalized gradients of an image [17]. After that, magnitude and orientation of each gradient are evaluated. Proper ones make contribution on an accumulator. After voting process is completed, accumulator entries which exceed threshold are considered as center points of speed signs.

Liu et al. preferred to use gradient of logarithm of grayscale image [15] and extract edges from magnitudes of gradients After applying noise reduction on edge image, Fast radial symmetry algorithm is used to detect ROIs. On the other hand Cao and Deng, used gradient information to compute angle of pixels [27]. After angle calculation, angle values are binarized as given in the Figure 1.3 and one Integral Map (IMap) is calculated for each bin.

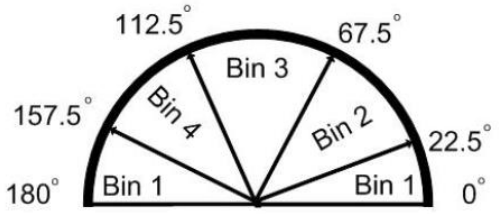

Figure 1.3: Binarization scheme of calculated angles of gradients

Due to only one sign is handled in this work; detection and recognition are combined in the proceeding steps. A sliding window is applied to input image and in the window area, selected Histogram of Gaussians (HoG) features are extracted from IMap data for each pixel. Finally, HoG values exceeding predetermined thresholds are used for recognition.

Although HoG calculation is a method for sign recognition, it is not one of the most preferred methods. The most preferred classification methods are Neural Network (NN) [21], [16], [19], [22], [28], correlation [5], [15], [25], [26], [27] and Support Vector Machine (SVM) [17], [18], [28], [29], [30] based methods.

Almost the same methodology is used by both Escalera et al. [22] and Garrido et al. [16]. 2 NN's are employed in both works. One is used to recognize circular signs and the other one is used for triangular signs. Results of detection part are normalized to a pre-determined size. For normalization, it is stated that bilinear and the nearest neighbor approaches gives similar results and the nearest neighbor approach is used because of its lower calculation complexity [22]. Besides trained signs, no sign output is also included in both works. In a similar way, Prieto and Allen prefer to divide sign into 16 blocks and use these block as input to Self-Organizing Map (SOM) [21]. On the other hand, Moutarde et al.is not used whole sign as input to NN [23]. First, characters in the speed signs are segmented and each digit is normalized. Normalized digits are used as inputs to multilayer perceptron (MLP) and for each digit this MLP gives positive output at one of ten outputs. This method enables system to recognize speed limit signs which are not trained to system before.

Höferlin and Zimmermann do not use image as input of NN directly [24]. First, SIFT is applied to detected sign. Then, obtained feature vector is given to MLP as input. Moreover, at the output of NN, each sign is classified with 3 different quality levels, which are perfect, medium and bad.

Besides NN, correlation based methods are preferred for sign classification. Usually, there is no need of training images to system. This is the basic advantage of correlation based methods over NNs. Changing traffic signs in database is easier than doing same in NN based classifiers.

Comparing unknown sign and templates in the database several methods are used in literature. Khan et al. [11] and Harasthy et al. [13] prefer to use Fourier transform. In [11], a joint image is generated by putting input scene and template side-by-side, then Fourier transform is applied to joint image. A fringe filter is applied to this tranform and then inverse transform is taken. If input scene and template are higly similar, peaks exist in the output. This method is named as Fringe-Adjusted Joint Transform Correlation (FJTC). Harasthy et al. prefers to generate a joing image by putting input scene and all templates together. Optical fourier transform is applied on a cambridge correlator and position of peaks in the transformed image gives the best match information.

Instead of calculating Fourier Transform, Waite and Oruklu prefer to compute dissimilarities between input scene and templates [7]. For this purpose, input scene is resized to a predefined dimension. After that Hausdorff distance between input scene and all templates are calculated. Housdorff distance between image F and G can be computed with equation 6 and 7.

$$D_H(F,G) = \max\{h(F,G), h(G,F)\} \tag{6}$$

where

$$h(F,G) = \max_{f \epsilon F}(\min_{g \epsilon G}(f,g)) \tag{7}$$

With this equation, the minimum distance between each pixel in the first image and every pixel in the second image is calculated. Also same is calculated for each pixel in the second image. Finally, maximum of these minimums give a measure of most out of place pixel, or dissimilarity between images.

Ruta et al. also resize input scene to a predefined dimension [29]. However instead of calculating dissimilarities pixel by pixel, it is preferred to divide input scene to 4x4 blocks. Therefore, they have 225 regions to compare with templates. Dissimilarities between these regions and template regions are calculated via Color Distance Transform (CDT). In CDT, discretized color information is also included in comparisons.

As well as NNs and correlation based methods, Support Vector Machine (SVM) is frequently preferred for sign classification. SVMs are used in different ways to handle sign classification problem.

Park et al. prefer to use binary decision tree (BDT) architecture due to its computational performance. Computation time is reduced up to 14 times when BDT is compared with 1-vs-all and 1-vs-1 architectures[4]. Liu et al. also use BDT [15] to classify speed signs as given in Figure 1.4.

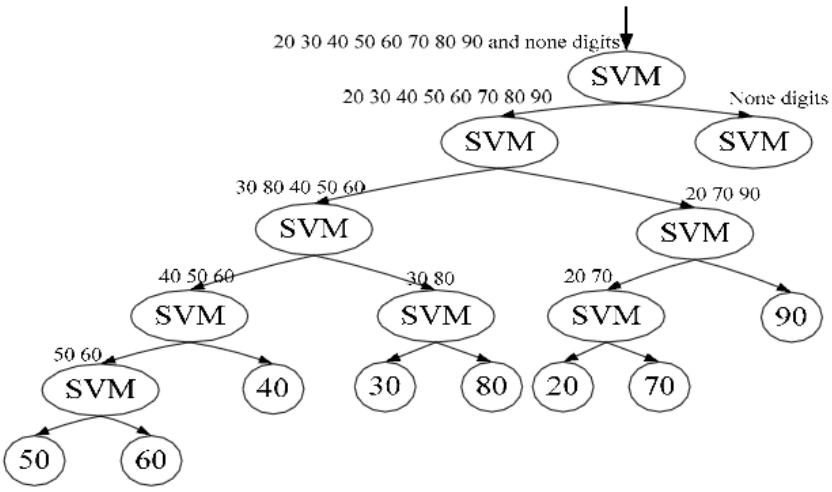

Figure 1.4: Binay decision tree architecture used in [15] to classify speed signs

Although BDT is used in both works, completely two different methods are employed to extract feature vectors. In [4], SIFT is used because of its scale and rotation invariance and robustness. On the other hand, in [15] a complex algorithm is proposed to achieve rotation invariance. First, input scene f(x,y) is converted to polar coordinates g(r,θ). The origin is set to center of mass of the image. Then 1-D fourier transform is applied along polar angle axis. Then 2-D wavelet transform is applied and 1024 dimensional feature vector is obtained. Finally, F-score method is applied to reduce number of dimension and enhance the generalization.

In both [30] and [6], Zernike Moments are used to extract feature vectors. While, Shi et al. obtain the best results with linear kernels; Chen et al. prefer to use a Gaussian kernel. Maldonado-bascon et al. also prefer to use Gaussian kernels for classification[31]. In their work, first, a class is assigned to every sign according to its shape and color, which are determined at detection part. Then, a 1-vs-all SVM architecture is implemented for each class.

Besides, popular classification methods mentioned above, Linear Discriminant Analysis (LDA) [17] and nearest neighbor [9] classification methods are also utilized to recognize traffic signs. All classification methods are achieved to a satisfactory classification rate. Moreover, in some works recognition is moved one step further with tracking detected sign [16], [29], [32], [33]. In these works, tracking is used to reduce computation load. Furthermore, in [19] tracking is also employed and it is used to increase reliability of the system. Sign recognition results obtained from consecutive frames are compared and a combined result is generated if there is consistency.

1.2 Scope of Thesis

In this thesis, a real-time road sign detection and recognition system is proposed. Purpose of this system is to detect and recognize in 640x480 pixel sized input images. For detection and classification of traffic signs, each input image passes through the dedicated system parts given in Figure 1.5.

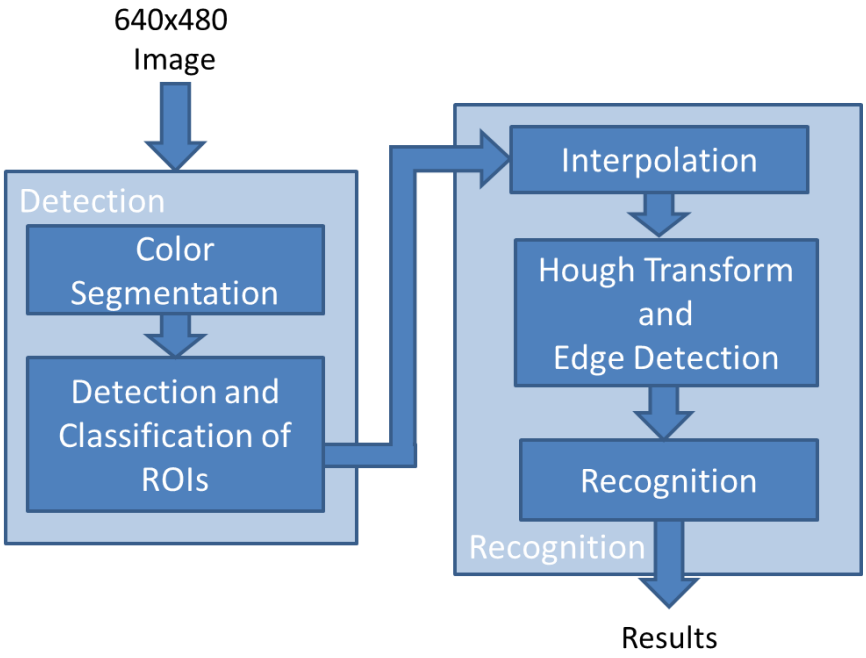

Figure 1.5: Flow diagram of the proposed system

Proposed system is composed of 2 main parts, which are detection and recognition. In detection part, input image is searched for traffic signs. For this purpose, first, color segmentation is applied to input image and blue, red and yellow regions are extracted. After that, segmented regions are downsampled and blobs are detected. These detected blobs are classified according to several conditions, which are detailed in Chapter 2.2, and labeled as interest or non-interest regions. Regions that meet all classification conditions are considered as traffic signs and transferred to recognition system.

Recognition part is the second stage of the system. In this part, detected regions are interpolated to a common size, which is 60x60. After that, color segmentation is applied to these regions in order to eliminate noisy structures and Hough Transform is applied to segmented regions. Output of Hough transform is used to detect exact contour and shape of the traffic signs. Finally, based on shape and color information of traffic sign, Informational Pixel Percentage (IPP) [34] classification method is applied to signs and final results are obtained.

Algorithm development stage for this system is implemented in MATLAB. Implementing algorithms and observing their result is considerably fast in MATLAB environment. Therefore, whole algorithm is developed and confirmed with several images in MATLAB, before implementation in Field Programmable Gate Array (FPGA).

In order to implement developed algorithm for a real time application, high processing power and pipelining capability is needed. Furthermore, same operation is executed for different parameters like blob detection for results of different color segmentation operations. Therefore, parallel processing capability is also a key parameter. These requirements lead this work to be implemented in FPGA. An FPGA with rich memory resources and embedded processor is the optimum platform for proposed system. Therefore, proposed system is implemented on Xilinx ML507 evaluation board, which includes a Virtex-V FX70T FPGA. This FPGA includes a PPC440 processor and rich logic resources.

1.3 Thesis Outline

In this thesis, in Chapter 1, introductory information about traffic signs and traffic sign recognition systems is presented. Moreover, in this chapter general outline of implemented system is also given.

Region of Interest (ROI) detection and classification algorithms is given in Chapter 2. In this chapter, color segmentation algorithm is presented and blob generation discussed for detection process. After that, classification methods for detected regions are introduced.

First resizing ROIs are presented in Chapter 3. After resizing, segmentation and edge extraction are presented. After that Hough Transform implementation for exact contour information and shape classification is discussed. Finally, IPP method for recognizing detected signs is presented.

In Chapter 4, first implementation environment is introduced. Then, low level hardware implementations of used algorithms are presented with details. Moreover, algorithms implemented in software are described with pseudo-codes.

Test results and equipment of the system are presented in Chapter 5. Besides results, strong and weak points of the system are also discussed.

Finally, in Chapter 6, conclusion of the thesis and future plans about the system are given.

CHAPTER 2

DETECTION OF TRAFFIC SIGNS

2.1 Color Segmentation

Red, blue and green are the most common colors used for traffic signs. Although it is not as common as these colors, yellow is also used for informative traffic signs. Red color is used for warning, blue is for informative and green is used for highway signs[20]. In this thesis, it is aimed to handle red and blue signs. Furthermore, yellow signs are used in the verification process of red and blue signs which will be explained in details in Chapter 2.2. Because of the given reasons, it is desired to detect specific tones of red, blue and yellow regions in the given frames.

Color segmentation is the first process applied to frames given to the system. Its outputs have great importance due to following 2 reasons;

- If any sign region could not be detected in segmentation process, that sign is discarded and could not be recognized.
- Most of the irrelevant information can be eliminated; therefore, there would be no need for time consuming algorithms.

In this work, it is preferred to process as many regions as possible and eliminate highly irrelevant regions. Therefore, a wide tone range of desired colors is employed. Moreover, that decreases the effects of illumination changes in daytime. Although, it is more likely to design robust systems against illumination changes in HSV space, hardware complexity and speed requirements prevent the usage of this space. On the other hand, the utilization of the segmentation algorithm given at [22] is optimal for implementation in hardware. Moreover, by using true threshold values, it gives significantly successful results.

Equation 8 defines the color segmentation algorithm used in this thesis for red color. In this equation, R(x,y), G(x,y) and B(x,y) are the red, green and blue components of the pixel at x- y position. R_a is the minimum and R_b is the maximum threshold values for the red component of the given pixel. G'_a and G'_b are the minimum and maximum thresholds of green to red components ratio of the pixel, respectively. B'_a and B'_b are the minimum and maximum thresholds of blue to red components ratios of the pixel, respectively.

$$r(x,y) = k_1 \begin{cases} R_a \le R(x,y) \\ G'_a \le \dfrac{G(x,y)}{R(x,y)} \le G'_b \\ B'_a \le \dfrac{B(x,y)}{R(x,y)} \le B'_b \end{cases} \quad (8)$$

$r(x,y) = k_2$; all other situations

$$b(x,y) = k_1 \begin{cases} B_a \le B(x,y) \\ G''_a \le \dfrac{G(x,y)}{B(x,y)} \le G''_b \\ B''_a \le \dfrac{R(x,y)}{B(x,y)} \le B''_b \end{cases} \quad (9)$$

$b(x,y) = k_2$; all other situations

$$y(x,y) = k_1 \begin{cases} B(x,y) \le B_b \\ R'''_a \le \dfrac{R(x,y)}{G(x,y)} \le R'''_b \\ B'''_a \le \dfrac{B(x,y)}{R(x,y)} \le B'''_b \\ B'''_a \le \dfrac{B(x,y)}{G(x,y)} \le B'''_b \end{cases} \quad (10)$$

$y(x,y) = k_2$; all other situations

For blue and yellow color segmentations, variations of equation 8 (equation 9 and 10) are utilized, respectively. In Figure 2-1, a sample frame used for segmentation is presented.

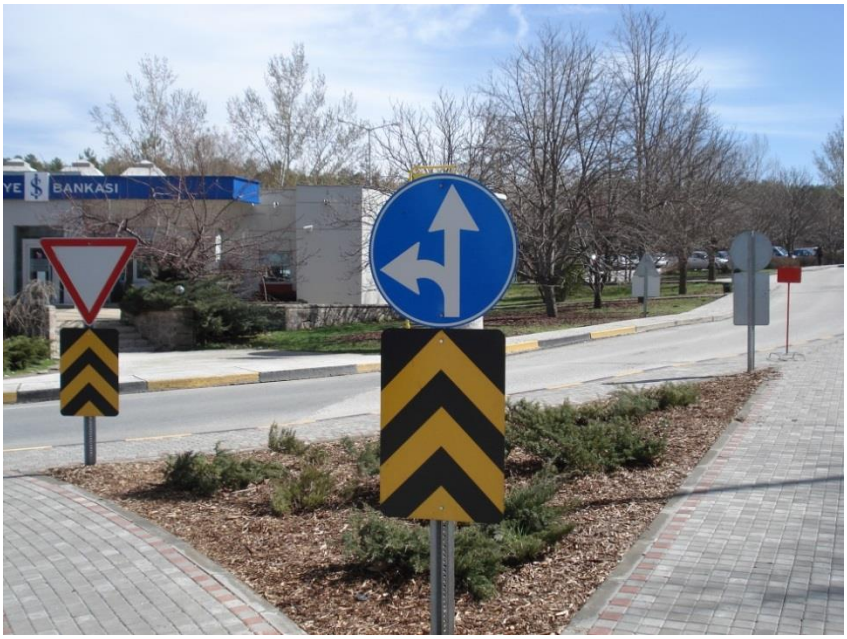

Figure 2.1: Sample Frame for Color Segmentation

The red (Fig. 2.2), blue (Fig. 2.3) and yellow (Fig. 2.4) segmentation results of this sample (fig. 2.1) are given as the following.

Figure 2.2: Red Segmentation of the Sample Frame

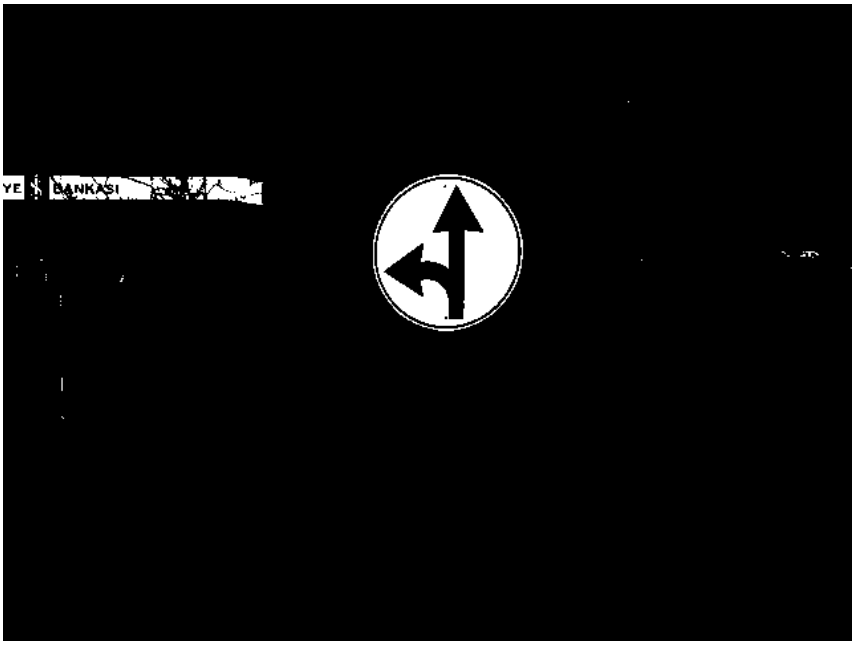

Figure 2.3: Blue Segmentation of the Sample Frame

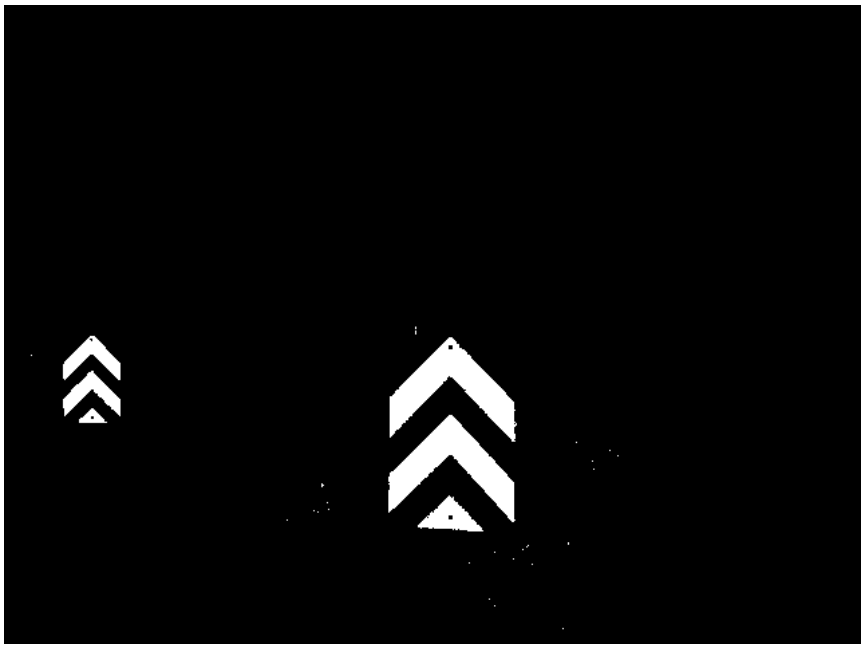

Figure 2.4: Yellow Segmentation of the Sample Frame

2.2 Detection and Classification of Region of Interests

In the color segmentation process, every pixel in the given frame is processed. By means of this process, irrelevant structures with appropriate shapes can be eliminated. However, each of three color maps has 640x480 pixel information. Furthermore, neighboring relation should be established between same colored pixels and blobs should be detected. Nevertheless, it is a time consuming process to examine all neighboring relations and decide resultant blobs. Therefore, initially a downsampling operation should be applied. For this operation, a method derived from [29] is used. In this method, first of all, each color map is divided into 8x8 sub-regions. Division of an input frame is given in the Figure 2-5. After this operation, pixels, which give valid result for desired color, are counted for each sub-region. Finally, valid pixel counts in each sub-region are compared with a threshold value and ones which exceed this threshold are marked as valid. This downsampling process, both reduces time complexity of applied algorithm and eliminates noisy regions. In Figure 2-6, application of this algorithm and its result for blue color map is given.

Figure 2.5: 8x8 Division of the Sample Frame

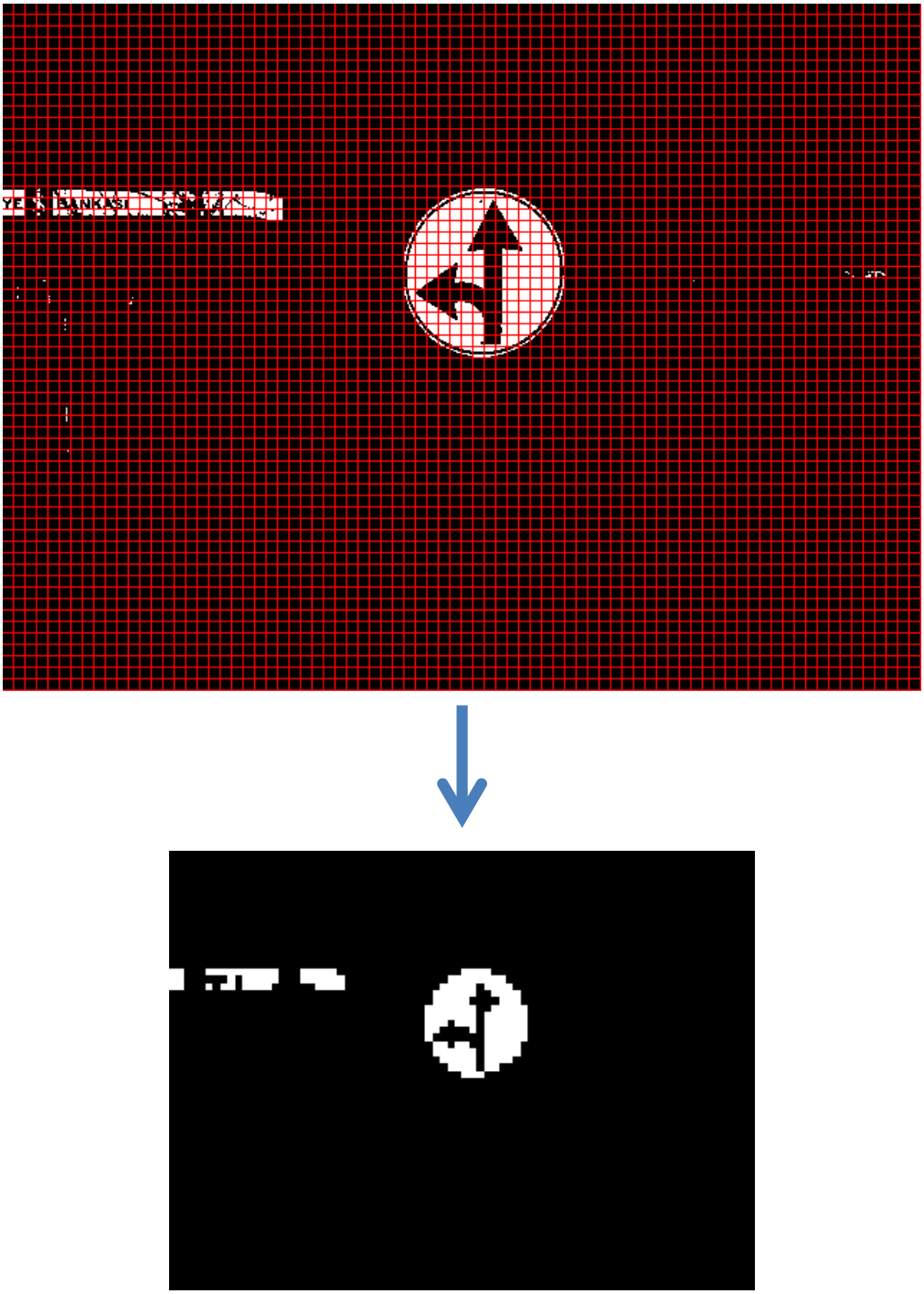

Figure 2.6: Downsampling Process of Blue Color Map

In this figure, it is clearly seen that no critical information is lost during downsampling and noisy structures are eliminated.

When downsampling operation is finished, an 80x60 color map is obtained for red and blue color. As it is also stated in chapter 2.1, yellow color is not used to detect traffic signs; therefore, it is downsampled to 20x15. Its detailed use will be given in the following parts.

After 80x60 red and blue color maps are obtained, blobs in these maps should be obtained; because each blob in these maps is a traffic sign candidate. For this purpose, each map is progressively scanned, as it is seen in Figure 2-7.

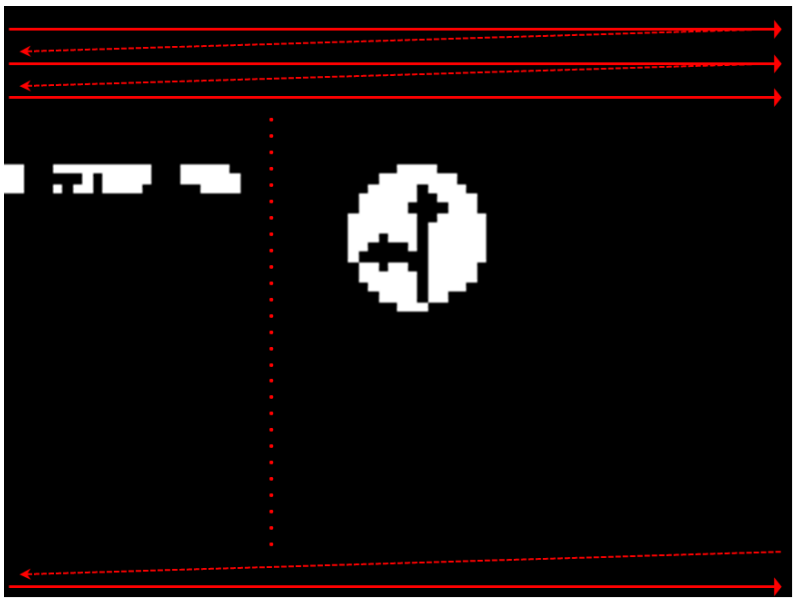

Figure 2.7: Scanning of Blue Map during Blob Detection

During this process, when first valid pixel is found, its coordinates is written in the first position of an array. After that, each valid pixel position is compared with the found blobs in the array. If any neighboring relation is not found, a new entry is opened in the array. On the other hand, when neighboring relation is found between a valid pixel and a blob, coordinates of the blob is updated to cover the valid pixel. Each blob in this array is defined by 5 parameters. X_0 is the starting and the X_1 is the ending points of the blob in X axis. In a similar way, Y_0 is the starting and the Y_1 is the ending points of the blob in Y axis. Final parameter is the color information of the blob. These parameters and orientation of axes are given in the Figure 2.8.

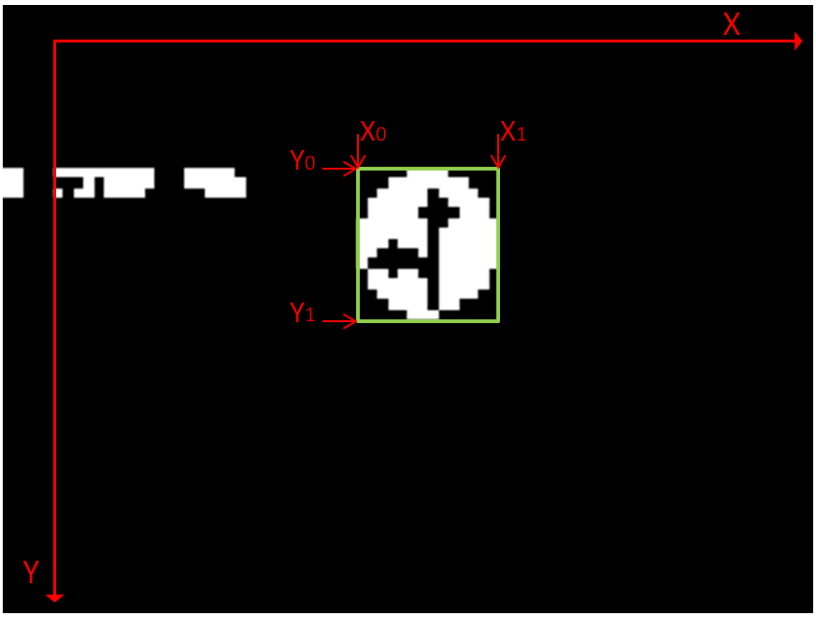

Figure 2.8: Orientation of Axes and Position Parameters Defining a Blob

During neighboring detection process between valid pixels and blobs, if a pixel will be included in a blob, one of two required conditions must be fulfilled. First, if bottom part of blob and the valid pixel are in the same row and end position of blob and valid pixel are side by side, valid pixel is included in the blob. This enables growth in x direction and is shown in Figure 2.9. In this figure, blob region is shown with green rectangle and valid pixel being processed is shown with small red square.

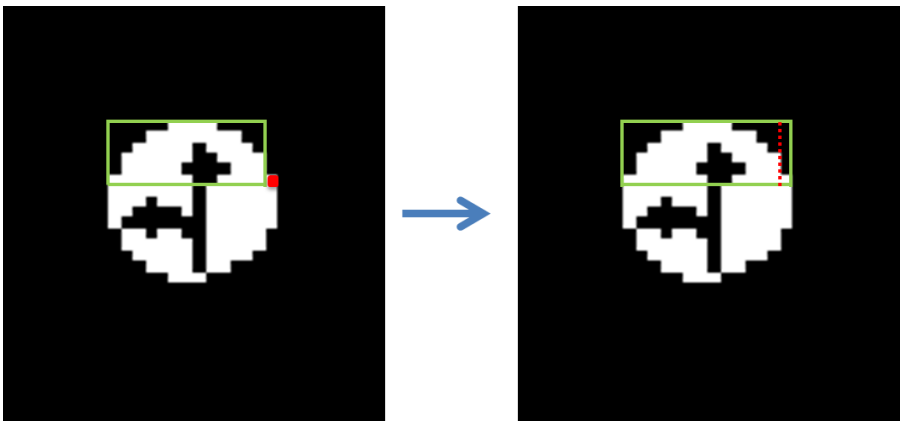

Figure 2.9: Blob Growth in case of Fulfillment of First Condition

To fulfill second condition of inclusion of a pixel to a blob, pixel should be in the next row of the bottom row of the blob. Moreover, its position should be between the (X_0-3) and (X_1+3) positions in the X axis. This case is shown in Figure 2.9. In this figure blob region is shown with green rectangle and acceptable region to include pixels to blob is shown with blue rectangle. Furthermore, small red square is the valid pixel being processed.

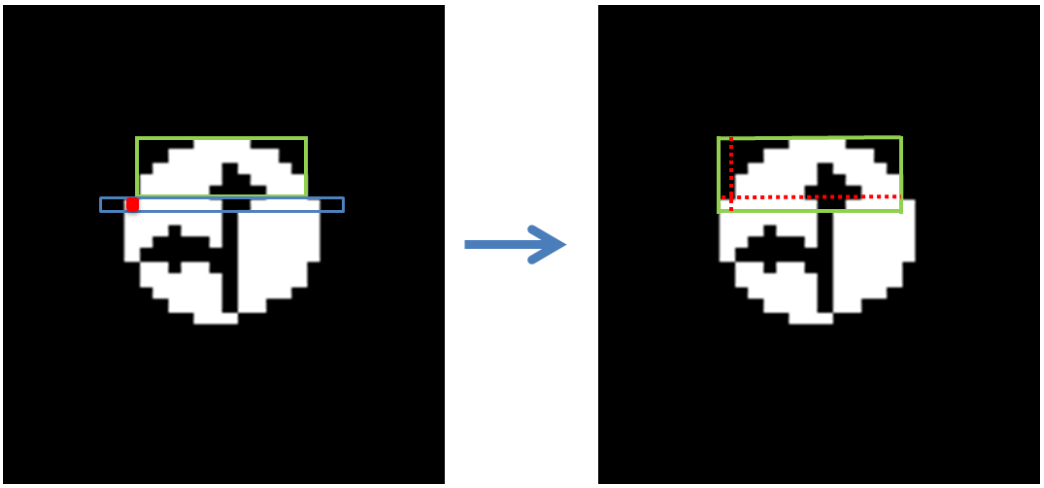

Figure 2.10: Blob Growth in case of Fulfillment of First Condition

After scanning blue and red color map is completed, coordinates of the blobs in these maps are obtained. This process is summarized in the logic diagram given in Figure 2.11.

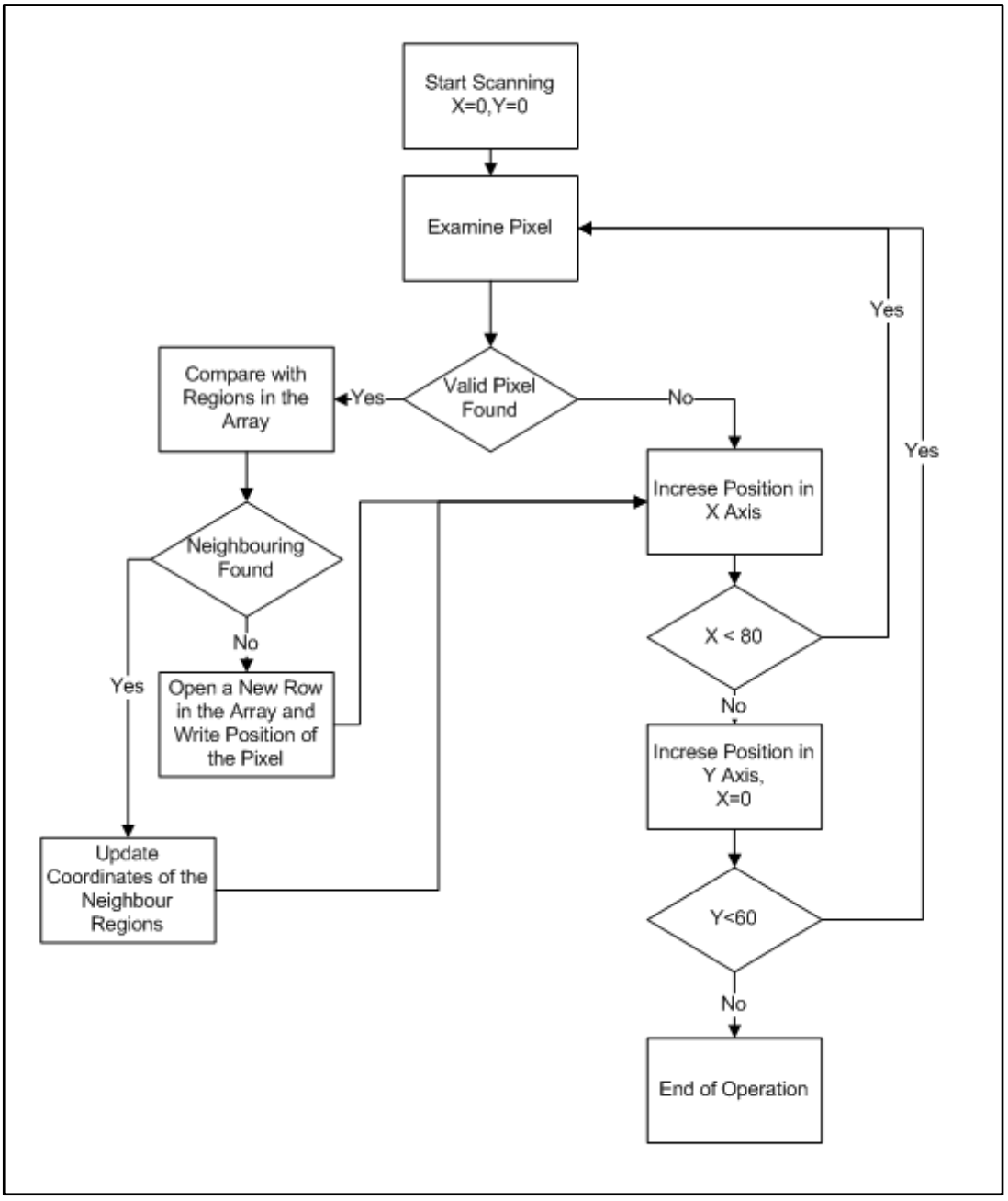

Figure 2.11: Flow Diagram of Blob Detection Process

It should be noted that, all regions in the frame with proper colors are detected in the previous step. Figure 2.12 and 2.13 present that seven regions, which are encircled with red rectangles, are detected in the downsampled blue and red maps. It is a well organized process to reduce candidate region number with low computation complexity; however, only a few or none of the detected regions belong to traffic signs. Therefore, these results should be refined to obtain ROIs, which have high probability of being traffic sign.

Figure 2.12: Detected Regions in the Red Color Map of Sample Figure

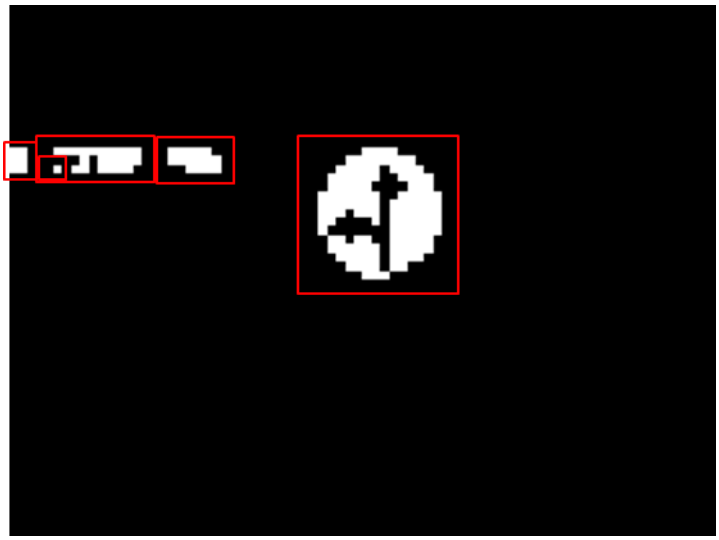

Figure 2.13: Detected Regions in the Blue Color Map of Sample Figure

In order to eliminate uninterested regions, some restrictions should be applied to all regions. First of all, dimensions of traffic sign candidates must be in the range of predefined thresholds. Length and width of a traffic sign candidate must be at least 40 pixels. Smaller regions are eliminated, because recognition algorithm cannot work properly on the smaller regions. Furthermore, if system operation is considered, small traffic signs can be captured with more appropriate dimensions in the following frames. In a similar manner, regions larger than 240 pixels are also eliminated, since in this situation,

traffic sign is too close to vehicle and recognition of this sign becomes meaningless. Furthermore, due to the frame processing speed of the system, which will be detailed in the chapter 4, the same sign can already be caught and recognized in the previous frames.

When height and width of the sign is in the predefined sizes, ratio between height and width is controlled. Standard dimensions of traffic signs in Turkey are defined by [20] (Table 2.1).

Table 2.1: Standard Dimensions of Traffic Signs Defined by Turkish Government

Shape Of Traffic Sign	**Standard Dimensions (mm)**
Circle	600
Triangle	900
Square	600
Rectangle	500x750
	600x900
Octagon	900

Based on this information (excluding rectangular signs) it is expected that the ratio between height and width is usually around 1. When rectangular signs are considered this ratio takes a value around 1.5. In the light of this information, even a reasonable perspective distortion is included, height to width ratio of a traffic sign is not expected to exceed 2. Therefore, if this ratio exceeds 2, there are two meanings of this. First, this region is not a traffic sign. Second, there is too much perspective distortion that recognition algorithm cannot be applied to this sign, successfully. For both cases, these regions have no meaning for the rest of the algorithm; therefore, regions, whose aspect ratio larger than 2 or smaller than 0.5 are discarded.

After these operations, proper sized and shaped regions remain. However, further refinement operations should be applied to these regions, to evaluate them in the recognition process and prevent false results. For this purpose, merging algorithm is applied to all of the remaining regions. In this process, coordinates of the regions are compared and intersections are found. If area of an intersection is larger than 50% of the area of the smaller area, shapes are united and coordinates of one of them is updated to cover both regions. The other region is labeled and excluded for next operations. There are two main reasons of use of this step. First reason is to detect signs which have both red and blue colors. For example, “No Parking” and “No Parking and Stopping” signs are composed of both blue and red colors. Moreover, these signs can be interpreted as valid

regions by both red and blue sign detection structures. Therefore, one sign may cause two ROIs and in the recognition process time is wasted for recognizing same sign twice. Furthermore, if one of the shapes is not eliminated, same sign would have to exist in the database of both red and blue colored signs. In Figure 2.14, process outputs are given for a "No Parking or Stopping" sign. According to these outputs, two valid regions are obtained from one sign but this problem is solved by merging operation.

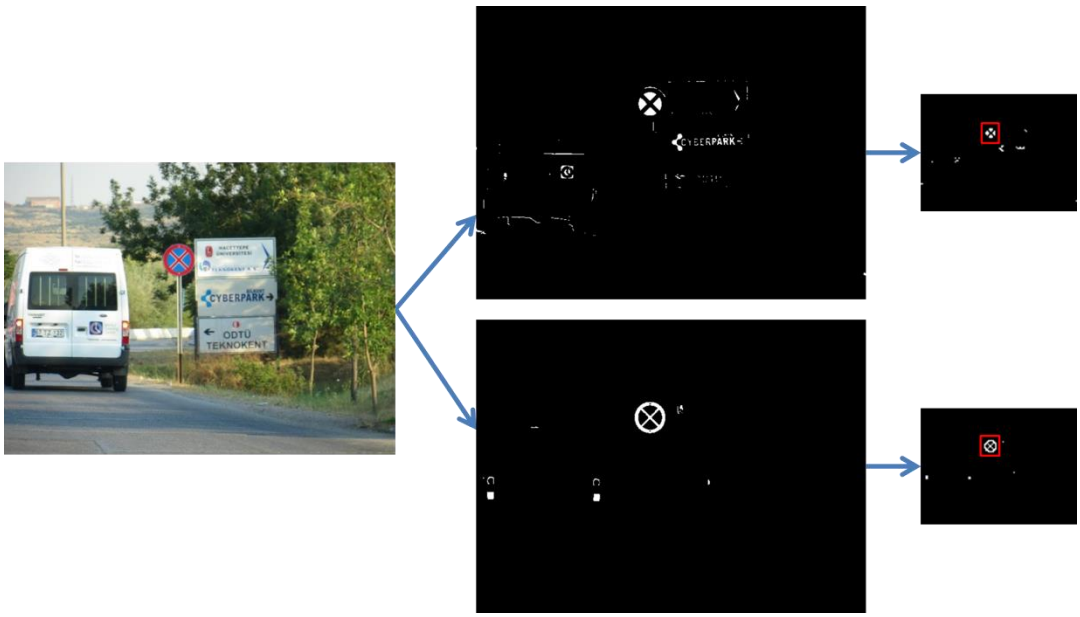

Figure 2.14: Segmentation and Detection Process Outputs of a "No Parking or Stopping" Sign

The second reason of application of merging algorithm is occlusion. Some occlusions may cause discontinuity in color map of shapes. In some certain cases, these discontinuities can result in two different regions due to the working principle of detection module. Furthermore, these regions may have proper shape and size properties and cannot be eliminated until merging process. This case is given in the Figure 2.15. In this figure, one of the regions is enclosed by the other one and it is easy to eliminate smaller region with application of merging algorithm.

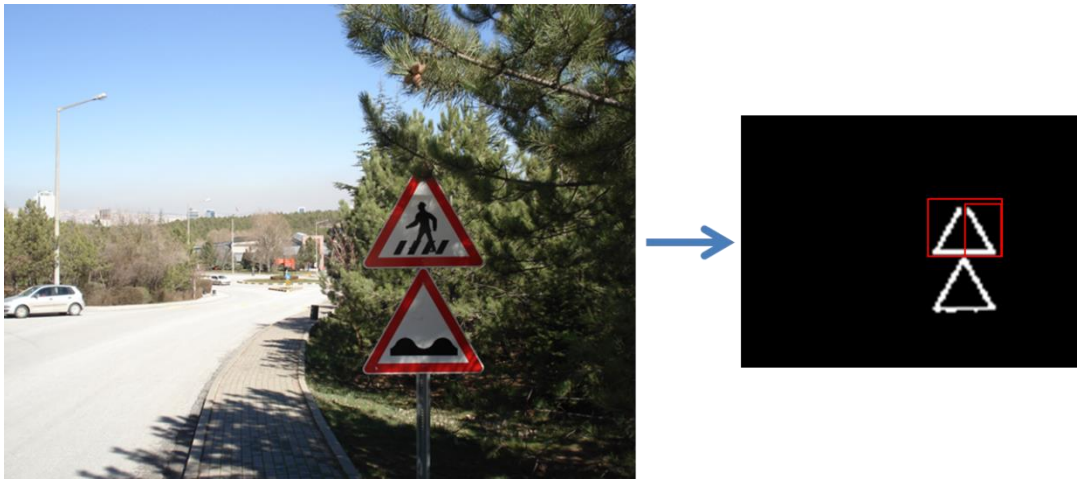

Figure 2.15: Detection of Two Regions from a Sign Caused by Discontinuity in the Color Map

Operations employed up to this point increases reliability of the system and decrease complexity in the recognition part by eliminating irrelevant regions. Although most of the irrelevant regions are eliminated, a proper colored and shaped region can pass through all of these processes. Therefore, a more specific classification step must be applied to all of the remaining regions to increase reliability. For this purpose, properties, which can discriminate a sign from a colored region, are investigated. By examining placements of traffic signs in the urban area, it is observed that signs are placed mostly utilizing one of three ways. First kinds of traffic signs are the single signs are hold by poles and this is highly expected. Second kinds are multiple traffic signs vertically placed on the same pole. Third of the considered signs are informative signs that placed over yellow “Additional Traffic Island Sign” or “Additional Traffic Circle Sign”. One example for each kind of traffic signs are given in the Figure 2.16.

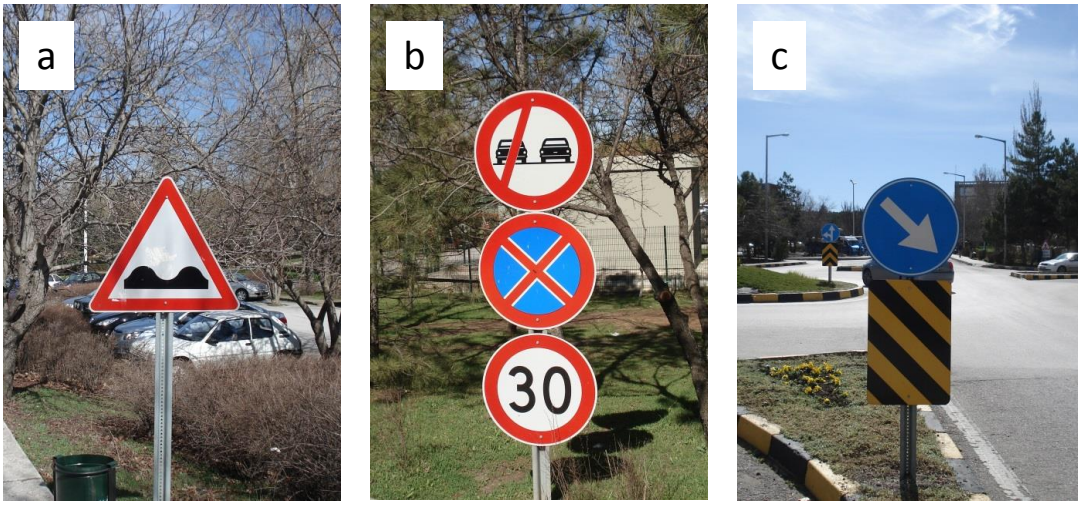

Figure 2.16: Most Commonly Observed Traffic Sign Placements. (a) Single Traffic Signs Placed Over a Pole, (b) Multiple traffic signs placed vertically, (c) Informative traffic signs placed over an "Additional Traffic Island" sign.

In order to discriminate a traffic sign belonging to one of these three kinds from the non-traffic sign regions, three different classification methods are applied. Regions meeting requirements mentioned above need to fulfill one of the three classification steps. It should be noted that regions fulfilling one of three requirements go into recognition process and others are discarded. Application sequence of the methods is decided in accordance with the consumed time by each method. The least time consuming method is applied first. By this means, the overall process time is minimized.

As the first process, it is controlled that whether there is a yellow sign below the traffic sign or not. In order to execute this process, previously obtained 20x15 sized yellow color map is used. As it is also stated before, no exact position information of yellow signs is required since, only existence of yellow traffic sign below the interested region is important. For this operation, coordinates of interested region is divided by four and corresponding coordinates in the yellow color map are found. After new coordinates are obtained, search region below the candidate sign is determined. Mapped coordinates of candidate sign in X axis are used as they are. In Y axis, ending coordinate of candidate sign is taken as the starting point of search region. The end point of the search region is determined by adding two to starting coordinate. Values of the yellow map in this area are read by the system and if any valid region is found, candidate sign is labeled as traffic sign. Selection of search region in the blue color map and detection of valid area in the yellow color map is given in the figure 2.17.

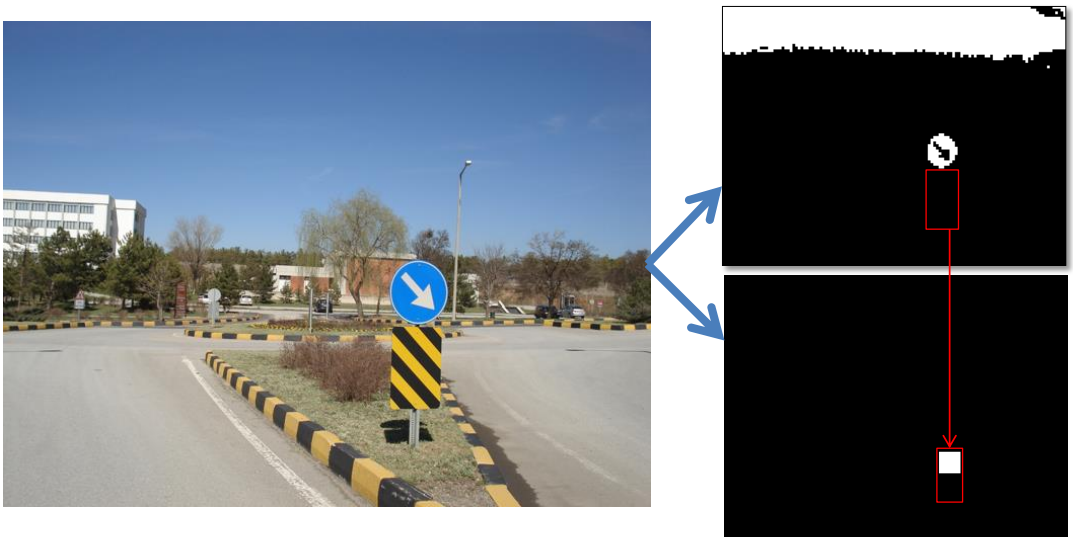

Figure 2.17: Detection of yellow sign below the informative signs.

As the second process, case of alignment of multiple signs over the same pole is checked. For this process, center position of the sign being controlled is calculated. Next, it is first shifted below by the height of the sign being controlled, then shifted above. After each shift, it is controlled that shifted center point is whether enclosed by another sign or not. If it is enclosed by another sign, this region is labeled as traffic sign. This process is given in the figure 2.18. In this figure, the point of the sign at the top shifted below as much as the height of the sign and this is shown with the red arrow. End point of the red arrow is enclosed by another sign candidate, therefore this region is accepted.

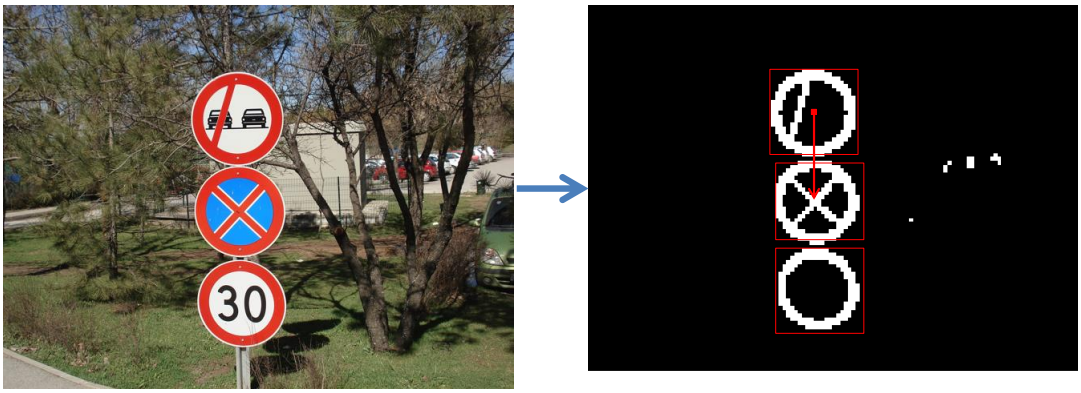

Figure 2.18: Detection of multiple regions in the same vertical line

Finally, if a region is not fulfilling one of the two conditions stated above, it is assumed that this region belongs to a single sign on a pole. In this condition, a pole can be detected in the area below the candidate sign. To decide whether there is a pole or not below the candidate sign, first coordinates of a 60x60 sized region is decided. While making this decision, starting point of this region in Y axis is acquired by adding 15 (pixels) to end point of candidate sign in the same axis. Midpoint of the candidate sign in X axis is taken as midpoint of this region. In the initial MATLAB simulations, size of this region is decided according to size of the candidate sign; however, hardware implementation of a variable sized filter is not efficient. Therefore, considering the largest sign which can be accepted, the size of pole search region is set to 60x60. It should be noted that extracted region is not processed in colored domain. Pole search region is obtained from green component of the image via calculated coordinates. After that point edge detection algorithm is applied to extracted 60x60 sized region. Extraction of a pole search region and output of edge detection process are given in the figure 2.19. Finally, a histogram in X axis is generated from the output of the edge detection algorithm. If at least two vertical lines, whose lengths are at least 70 % of height of the extracted region, are detected in the histogram, it is decided that there is a pole below the candidate sign. As a result, candidate sign is labeled as a traffic sign and its coordinates transferred to recognition module. Flow diagram of the overall classification process is given in the figure 2.20.

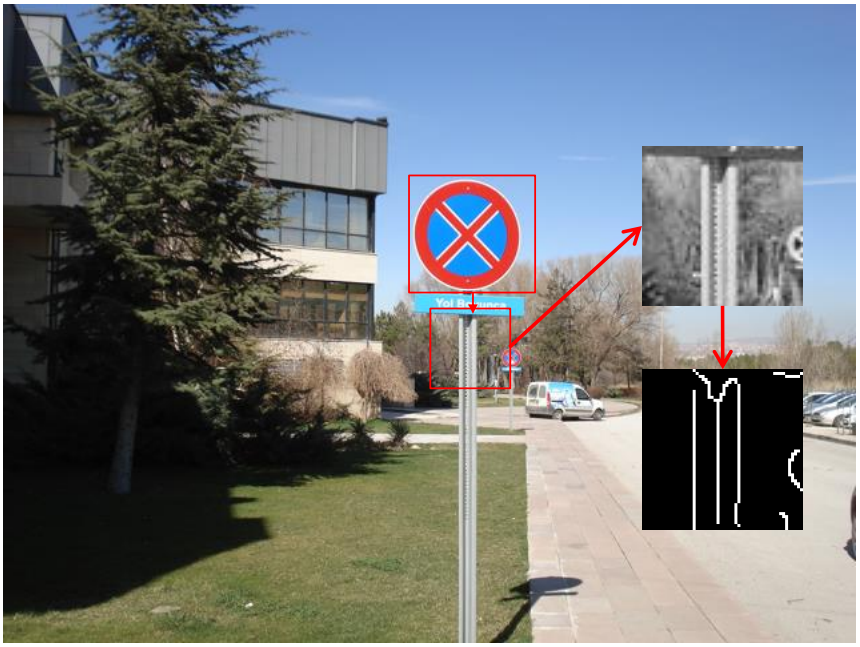

Figure 2.19: Extraction of a pole search region and detection of edges

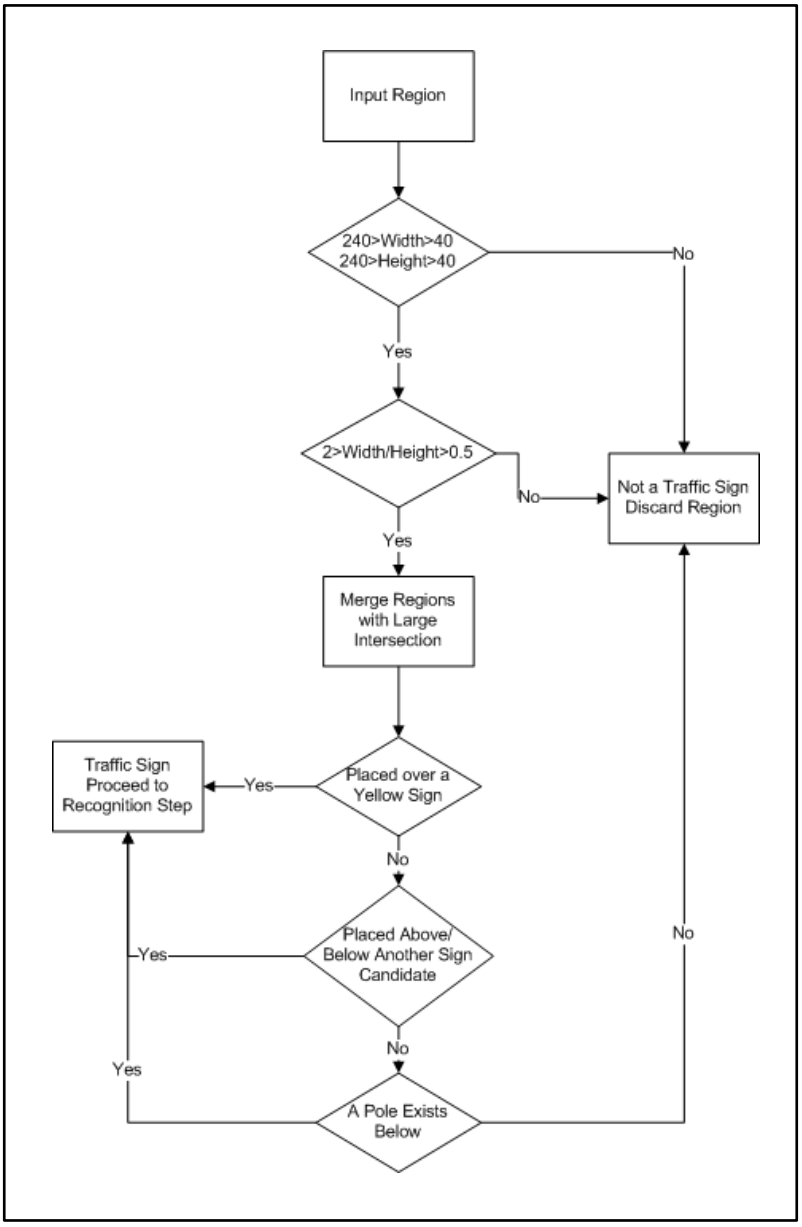

Figure 2.20: Flow diagram of the region classification process

CHAPTER 3

RECOGNITION OF TRAFFIC SIGN

Recognition part of the system is based on Irmak's recognition system [34]. In his work, proposed IPP algorithm has high recognition rate and it is suitable for embedded implementation. Furthermore, in his work, required procedures to apply IPP algorithm is presented clearly. By this means, outputs of Irmak's thesis are utilized to build recognition system.

3.1 Interpolation of ROIs

When detected regions are classified as ROI, they need to be scaled to a common size. This operation is not applied for Informative Pixel Percentage (IPP) method. IPP method use normalized values; therefore it is scale invariant, which is explained in Chapter 3.2 in details. However, exact contour information is needed for proper operation of IPP method. Therefore, Hough Transform is applied to each ROI to get information about traffic sign contour. Hough Transform is a time and memory consuming algorithm. Moreover, as application area increase, computation time and memory requirements also increase. Besides maximum time consumption, size of the smallest ROI accepted by the system is 40x40 and the largest one is 240x240. This difference causes too much variance in computation time and a constant time for processing a frame cannot be attained.

In the light of these reasons, all ROIs are scaled to a common size (60x60). However, information in the ROI should be maintained to some extend for successful operation of IPP method. Moreover, aspect ratio of initial ROI has crucial importance and it must be kept constant through scaling. Therefore, coordinates of ROIs are updated to equalize height and width of ROI.

Gonzalez and Woods states that bicubic interpolation usually gives better results than bilinear and nearest neighbor techniques in terms of preservation of details[35]. Therefore, bicubic interpolation is employed through algorithm development process in MATLAB. Efficiency of the recognition algorithm and implementation bugs can be examined efficiently by conserving details as much as possible in the scaling process.

Main disadvantage of bicubic interpolation is its execution speed. For calculating a point, information of sixteen neighbor points is used as given in equation 11; therefore, scaling operation takes too much time.

$$v(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij} x^i y^j \tag{11}$$

Therefore, after recognition system is finalized, the nearest neighbor interpolation method is also tried instead of bicubic interpolation. In Figure 3.1, results of bicubic and the nearest neighbor methods are given. In this figure, a 110x110 pixel sized input image (Figure 3.1(a)) is scaled to 60x60 with bicubic (Figure 3.1(b)) and the nearest neighbor (Figure 3.1(c)) interpolation methods. Furthermore, 60x60 images are again enlarged to details can be seen clearly.

(a) (b) (c)

Figure 3.1: Scaling results of a ROI (a) Input image (110x110) (b) Scaling to 60x60 with bicubic interpolation (c) Scaling to 60x60 with the nearest neighbor interpolation

In the given results, it is clear that bicubic interpolation gives smoother result than the nearest neighbor interpolation. However, it is obvious that enough information for the rest of operations is conserved. Moreover, conservation of information is also tested in the system and a decrease in the recognition performance is not observed. After performance of recognition system and time consumptions of interpolation algorithms are considered, the nearest neighbor interpolation is preferred over bicubic interpolation.

3.2 Edge Detection and Shape Classification

3.2.1 Edge Extraction

In 60x60 region, exact contour of traffic sign must be determined to deliver correct data to recognition module. For this purpose, line extraction and Hough Transform is applied to the input regions.

It is stated in Chapter 2 that color information is used to detect candidate signs. Therefore, it is known that input regions include red or blue colored shapes. In the light of this information, color information is also used in this part to eliminate noisy structures and reduce computation complexity.

Segmentation method (eqn. 10) used in Chapter 2, is utilized also in this part to segment colors. On the other hand, color information of the sign in the 60x60 region is known from the detection process. Purpose of segmentation in this part is not to detect whether there is a sign or not. Its main reason is to get rid of irrelevant structures; therefore, threshold values in the segmentation equation are loosened. Segmentation results of blue and red signs are given in the Figure 3.2 and Figure 3.3, respectively.

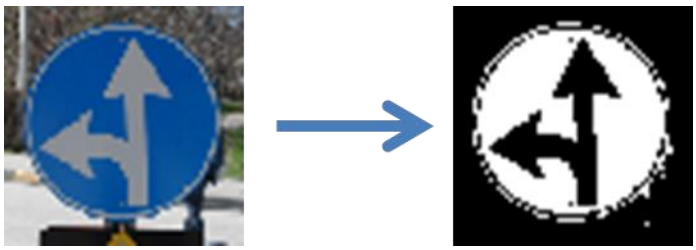

Figure 3.2: Segmentation of a blue traffic sign

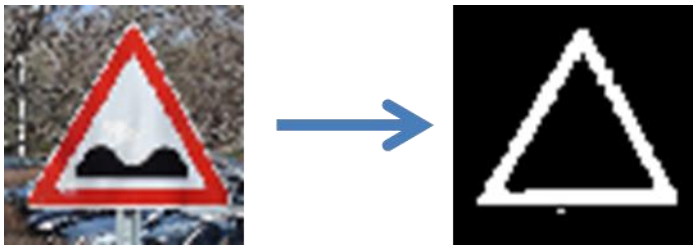

Figure 3.3: Segmentation of a red traffic sign

It should be noted that color segmentation in this stage reduces complexity of edge detection algorithm. Use of horizontal or vertical edge detection algorithms have no meaning because of circular and triangular shapes. Therefore, use of a 2-D edge detection algorithm is an obligation. On the other hand, use of a 2-D edge detection algorithm without color segmentation generates too much noisy structures as given in Figure 3.4. Because of these reasons, edge detection algorithm is applied to segmented image.

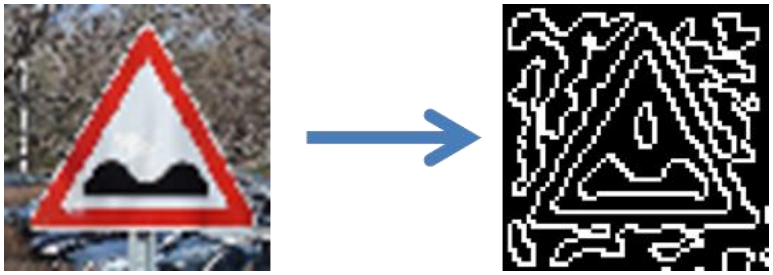

Figure 3.4: Edge detection on ROI not segmented

Irmak, suggest the use of Laplacian of Gaussian (LoG) filter for edge detection[34]. This is a very reasonable choice. Laplacian filtering is application of second order derivative image and it can successfully highlight rapid intensity changes[36]. Furthermore, Gaussian operation can compress noisy structures to some extent. Function of this filter is presented in (12).

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{12}$$

Using (12) and taking σ=1.4, a 9x9 digital filter kernel, presented in Figure 3.5, can be obtained.

0	1	1	2	2	2	1	1	0
1	2	4	5	5	5	4	2	1
1	4	5	3	0	3	5	4	1
2	5	3	-12	-24	-12	3	5	2
2	5	0	-24	-40	-24	0	5	2
2	5	3	-12	-24	-12	3	5	2
1	4	5	3	0	3	5	4	1
1	2	4	5	5	5	4	2	1
0	1	1	2	2	2	1	1	0

Figure 3.5: 9x9 LoG filter kernel

When LoG filter applied to an image, zero-crossing detection must also be applied to detect edges. Edge information of a traffic sign is obtained after application of these operations as presented in Figure 3.6.

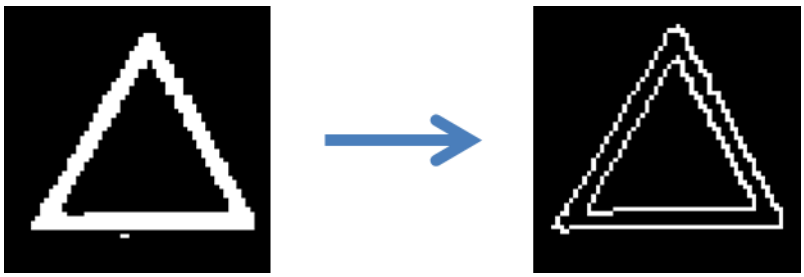

Figure 3.6: Extraction of edge information

LoG filtering and zero-crossing detection show a significant performance in edge extraction operation and give reliable results.

In edge detection, color segmentation is an obligation and LoG filter is applied to a binary image. Moreover, due to structure of its kernel, zero crossings must be detected to obtain a binary edge image. In this case, computation complexity can be decreased with some modifications. Instead of applying LoG filter and detecting zero-crossings, a modified zero-crossing detection algorithm can be applied directly to segmented region.

For this purpose, ROI is scanned with a simple difference detector. This difference detector kernel can be described with Figure 3.7

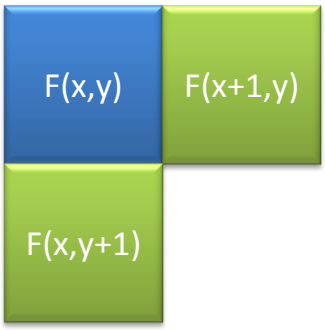

Figure 3.7: Simple difference detector kernel

If there is a difference between blue and green regions, kernel gives positive output. In other words, non-uniform regions in horizontal or vertical direction cause positive output. In case of any other conditions, a zero output is returned by the kernel. Its operation can be summarized with equation 13. In this function f(x,y) and g(x,y) are used for segmented and edge images, respectively.

$$g(x,y) = [f(x,y) \oplus f(x+1,y)] \vee [f(x,y) \oplus f(x,y+1)] \quad (13)$$

Result of this operation is presented in Figure 3.8. It can be seen that this operation is a successful approximation of optimal solution. Moreover, edge extraction for a blue circular traffic sign is presented in Figure 3.9.

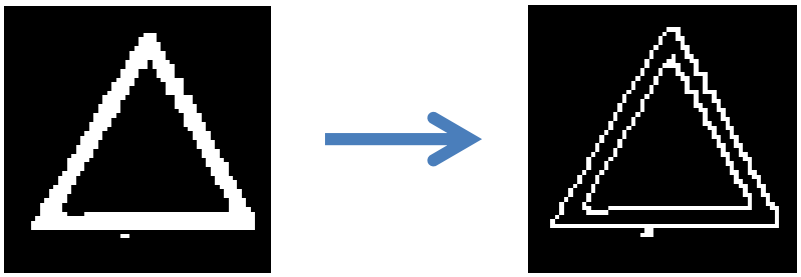

Figure 3.8: Edge extraction with simple difference detection kernel

Figure 3.9: Edge extraction for a blue circular traffic sign

For the rest of recognition process, detailed information should be obtained from extracted edges. Orientation, position and length of lines have significant importance for the rest of process. Therefore, Hough Transform is used to extract required information. It is a highly accurate and robust method [37]. Although, it has high computational complexity, this complexity must be endured for the success of recognition. Furthermore, different types of Hough Transforms must be employed to obtain parameters of different types of shapes.

Linear Hough Transform is used for detection of triangle signs. Circular signs are detected with Circular Hough Transform. Linear Hough Transform can also be used to detect rectangular signs; however, rectangular shapes enable us to use less complex algorithms. Therefore, a histogram based detection method is used for rectangular signs, which is given in the following part.

3.2.2 Triangular Shape Detection

Linear Hough Transform is used to detect lines in a region. Therefore, it can be explained by starting equation of a line. Equation of a line can be written as (14).

$$r = x\cos\theta + y\sin\theta \tag{14}$$

In this equation, r is the distance from origin to the line and θ is the angle of the vector, which connects origin the closest point of the line. If a coordinate plane is generated with r and θ axes, each point in Cartesian coordinate generates a sinusoidal in this plane with respect to (14). Furthermore, if two points are on the same line, defined by r' and θ', sinusoidal generated by these points intersect on r' and θ' in the r-θ plane. This case is presented in Figure 3.10.

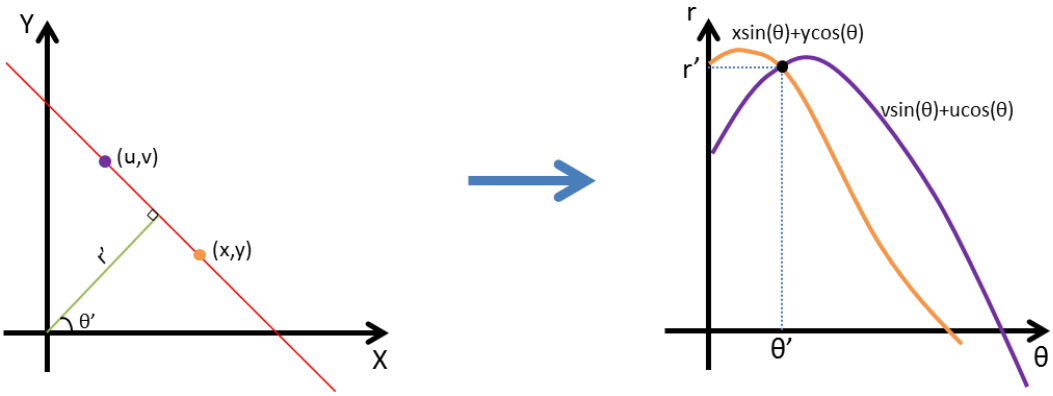

Figure 3.10: Mapping of two point on a line to r-$\boldsymbol{\theta}$ plane

To detect lines in the edge image, r values should be calculated for each θ value. However, edges of triangles are aimed in this process and searching edges for every possible θ is a waste of time. If there is a triangle in the edge image, three edges, oriented about 0, 60 and 120 degrees, exist. Therefore, instead of calculating r with respect to all θ values, only 0, 60 and 120 degrees are used with a margin of ± 5 degrees. Two edges are searched at each angle, because triangular signs have both inner and outer edges. At the end of detection of

six lines, outer ones are discarded, because only the region enclosed by inner lines is used for recognition. Detection of lines on the edge image is given in the Figure 3.11.

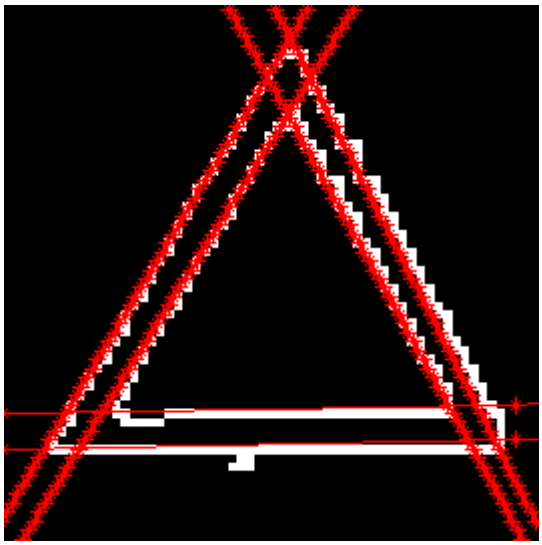

Figure 3.11: Detection of lines on a edge image

3.2.3 Circular Shape Detection

Detection of circles requires a more exhausting computation, which is circular Hough Transform. To describe a circle, three parameters are needed (15). These parameters are center points of circle in x axis (x_c) and y axis (y_c) and the radius of circle (r).

$$(x - x_c)^2 + (y - y_c)^2 = r^2 \tag{15}$$

If all of three parameters are unknown, a point generates a cone in hough domain as given in the Figure 3.12. Position of the point corresponds to coordinate of tip point of the cone. Furthermore, coordinate of each point on the surface of this cone gives the parameters of a circle, which can include this point. Therefore, intersection point of the cones generated by the points of a circle gives parameters of that circle.

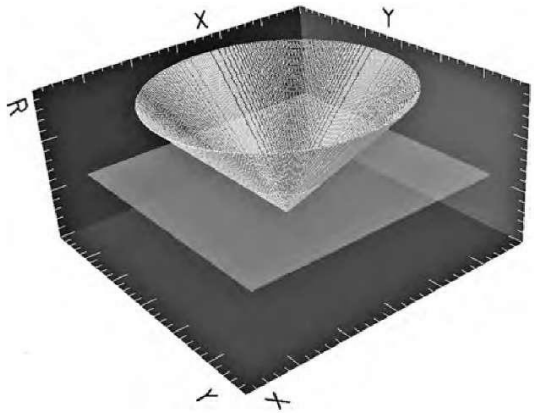

Figure 3.12: Cone in the Hough domain generated by a point [38]

It is previously stated that all traffic signs are affected from perspective distortion. Therefore, using only parameters of a circle is not an efficient method to get precise parameters. In (16), the equation that describes an ellipse is presented. In this equation, "a" and "b" are the radius of major and minor axes, respectively. Moreover, center point of the ellipse is described with (x_c, y_c).

$$\frac{(x - x_c)^2}{a^2} + \frac{(y - y_c)^2}{b^2} = 1 \tag{16}$$

$$(x - x_c)^2 + k^2.(y - y_c)^2 = r^2 \tag{17}$$

However, difficulty of using division operation is stated above. Therefore, (16) can be rearranged as (17) to get rid of burden of division. In (17), k is the ratio of major axis radius to minor axis radius and r is the major axis radius. In this case, there are four parameters to detect and this is a heavy computational burden. Some restrictions must be taken to decrease computation time to a reasonable extent.

First of all, k parameter should be considered. After experiments, it is decided that using k parameter in range of (0.5-2) is enough; because k values out of this range means too much perspective distortion and this much nonlinear distortion makes IPP give false results. Moreover, this distortion ratio is also used in the detection state and this selection makes system consistent.

Besides limiting values of k, r parameter should also be limited. In the detection step, regions smaller than 40 pixel are already discarded. Therefore, there is no meaning of taking r in the range of (0-30). Minimum search radius is taken as 10 pixel to be on the safe side. Result of the ellipse detection is presented in Figure 3.13.

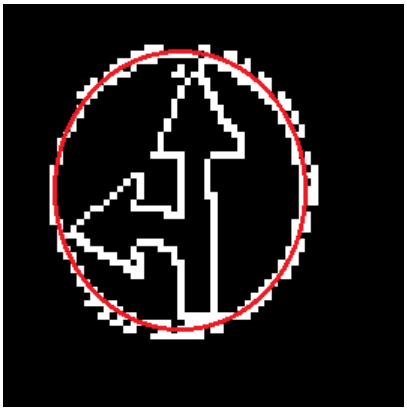

Figure 3.13: Detection of ellipse on the edge image

3.2.4 Rectangular Shape Detection

Detection of edges of rectangle is the easiest of all. Lines of rectangles can be detected by Linear Hough Transform; however, all lines are horizontally or vertically oriented. Therefore, histograms can be generated instead of applying Hough Transform. When evaluating output of the histogram, one edge is searched in the first half of the image and the other one is searched in the second half of the image. This process is repeated in both vertical and horizontal axes. In Figure 3.14, reflection of histograms to the edge image is presented.

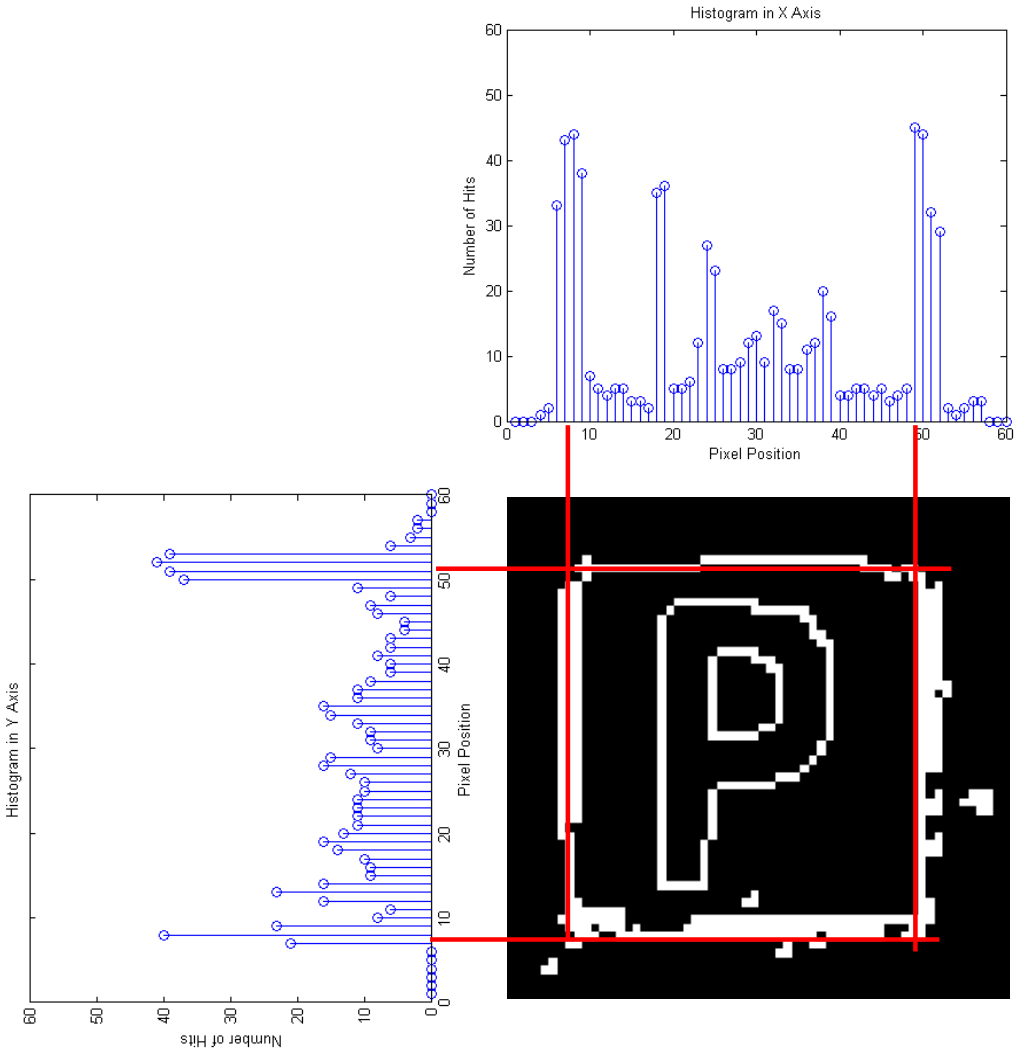

Figure 3.14: Detection of edges of a rectangle with histograms

3.2.5 Shape Classification

Application of the methods stated above gives all information about the shape; however, at the first instance, shape information is not known. Only information about the input regions is the color of the shapes. Shape colors are detected at the region detection stage and color information is also delivered to the recognition stage with the region coordinates. Moreover, for each color, there are two shape possibilities, as given in the figure 3.15.

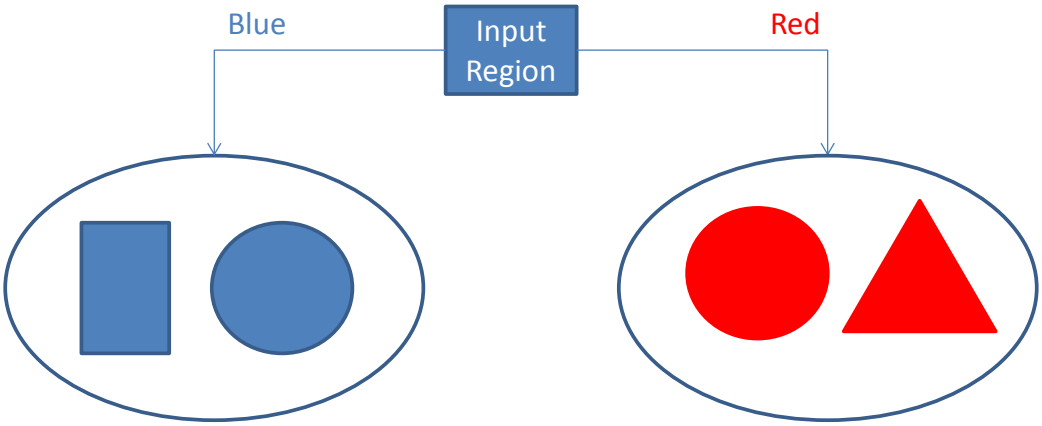

Figure 3.15: Possible shape types with respect to sign color

In the light of this information, two shape detection algorithms are applied. Circle and triangle detection algorithms are applied to red shapes. On the other hand, rectangle and circle detection algorithms are applied to blue shapes. It should be noted that all information of edges in the shape is obtained with detection algorithm. Moreover, if circular Hough Transform is applied to triangular or rectangular shapes, algorithm gives inconsistent outputs. In this case, detected circles by circular Hough Transform are the ellipses fitted in the rectangle or triangle as given in the Figure 3.16.

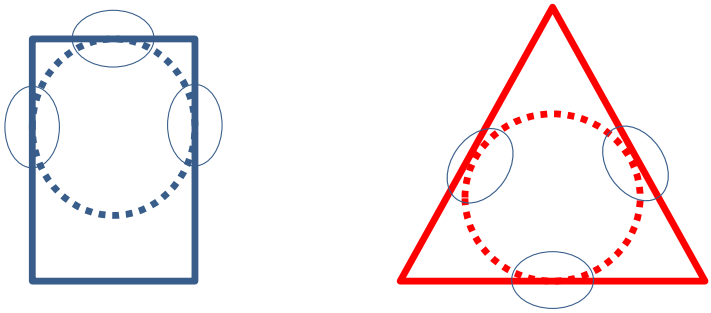

Figure 3.16: Ellipse fitting to triangular and rectangular shapes

It can be seen that number of points used to obtain parameters of circle are too few. Therefore, this number and detected radius generates an inconsistency. Furthermore, points on the edges of the triangle and rectangle are also obtained via detection methods. If the shape is red, number of points on the three edges is controlled and it is expected to have

similar values for a triangle. In the light of stated evaluations, shape is classified and sign recognition process begins.

3.3 Sign Recognition

Informative Pixel Percentage (IPP) is used for recognition of traffic signs, which is suggested by Irmak [34] .Although this method has a very low computation complexity; it shows very successful results on sign recognition. Moreover, a recognition ratio can also be obtained with the result, which is a very important feature to give idea about quality of recognition.

IPP is a template based method. It divides a sign into regions and compares normalized informative pixels in each region with the database. Data training does not needed by this method; however a clean interior area is need to be supplied to algorithm. Because of this reason, a considerably high amount of effort is given to detection of exact positions of edges. In the recognition step, first interior area of the sign is extracted. For this operation, edge information obtained from shape detection process is used and interior region of the sign is binarized. In the binarization operation, instead of using color image, usage of only green component is preferred. There are two reasons of usage of green component of the image. First of all, no computation is needed for obtaining this component. If gray scale image is preferred in this process, it is need to be calculated for all pixels. Secondly, no green sign is recognized in this system; therefore, green color space is highly discriminative for interested traffic signs. When binarization is applied to image, threshold is selected as the mean value of the pixels. Result of this process is presented in the Figure 3.17.

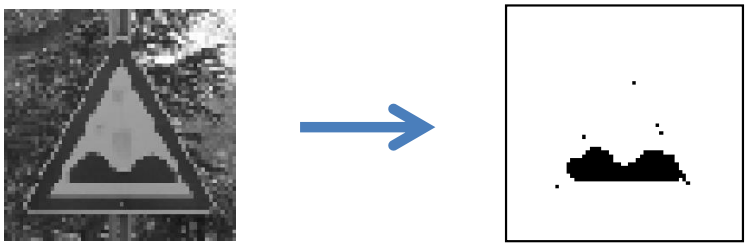

Figure 3.17: Binarization of the traffic sign

In order to increase recognition success, one more operation is applied to binary image. It seems in Figure 3.17 that some pixels can violate the border of inner region. Actually, this situation is expected; because, borders are detected with respect to majority of the pixels

and extracted lines sometimes cannot cover all pixels. Therefore, an extra precaution is taken with opening operation.

Opening, a combination of erosion and dilation, is a morphological operation. It generally smoothes shapes and breaks thin connections [35]. In our case, it can eliminate noisy pixels in the sign region with minimum distortion over the original sign. For this, operation first erosion operation is applied to image and single pixels are removed. After that dilation operation is applied and eroded original sign is expanded for minimum information loss. Result, given in Figure 19, is obtained with the application of kernels presented in Figure 3.18.

1	1
1	1

1	0
0	0

Figure 3.18: Kernels used for erosion and dilation operations

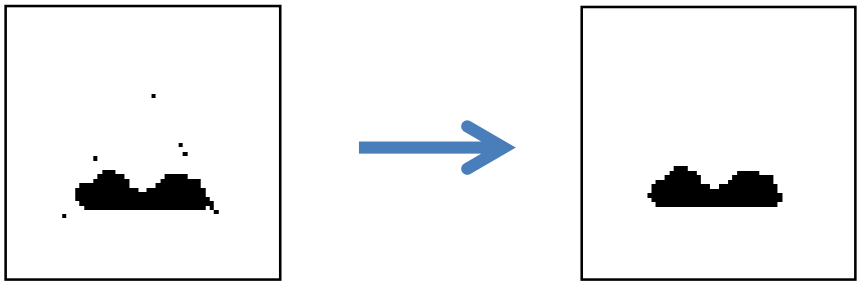

Figure 3.19: Result of opening operation

After the opening, traffic sign is ready for IPP operation. As it is stated above, in IPP method whole image information is not used for template matching. Instead of that, image is divided into sub-regions according to shape of traffic sign. Division methods of signs differ in order to maximize efficiency of recognition. In this aspect, triangular signs are divided in six equal regions. Circular signs are divided in 9 regions, however 5 regions include considerable information and these regions are used. Finally, rectangular signs are divided in 4 equal regions. Divisions of traffic signs are illustrated in Figure 3.20.

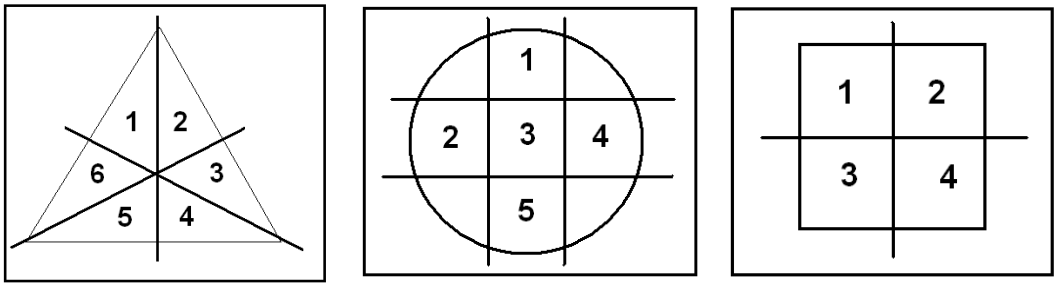

Figure 3.20: Division of traffic signs into sub-regions for IPP operation

After division operation, pixels describing sign in each region are counted. Then, these counts are normalized with total pixel count. After normalization process, obtained values are compared with the values in database. Comparison of the sign and database is based on Sum of Absolute Difference (SAD) method. Obtained minimum SAD value corresponds to best match and best matching sign is accepted as result. Moreover, it should be noted that a perfect match gives a SAD value of zero and this correspond to 100% recognition rate. As the value of SAD increases, it is known that quality of recognition decreases.

CHAPTER 4

HARDWARE IMPLEMENTATION

As it is presented in Chapter 2 and Chapter 3, several algorithms are applied to an image in order to detect and recognize traffic signs. Furthermore, in some cases, same algorithm is repeated for each color. This case causes high computation load. There are two options to overcome this problem. The first one is to use a sequential platform (CPU, DSP, etc.) and use as higher frequency as you can. The second solution is to divide operations in parallel lanes and pipeline them. The first one is the most straightforward solution and designed systems can be implemented in any computer. However, operating frequency (processor fabrication technology) is the limiting factor and there is not much to do to speed up implementation. On the other hand, if a parallel implementation platform is preferred; algorithms can be divided into several lanes and pipelined. Therefore, whole operation can be speed up several times.

GPUs, FPGAs and ASICs are the most common parallel processing platforms. ASICs are expensive to manufacture and not flexible devices. Therefore, they can be used for completely verified high volume productions. GPUs and FPGAs are flexible devices with high data processing bandwidth. This makes them suitable for advanced signal processing. In the proposed system, FPGA becomes a better choice because of its memory architecture.

ML507 Evaluation Board is preferred for the implementation of proposed work. In this board, there are several peripherals, required for an image processing system and a Xilinx Virtex-V FX70T FPGA. Moreover, Virtex-V FX series FPGAs have embedded hard PPC440 processors. Therefore, iterative and repetitive processes (inefficient to implement in HDL) can be implemented in this processor. Besides FPGA, this board includes an SSRAM, a DDR2 SDRAM, RS232 UART and USB interfaces etc. An image of this board is presented in Figure 4.1.

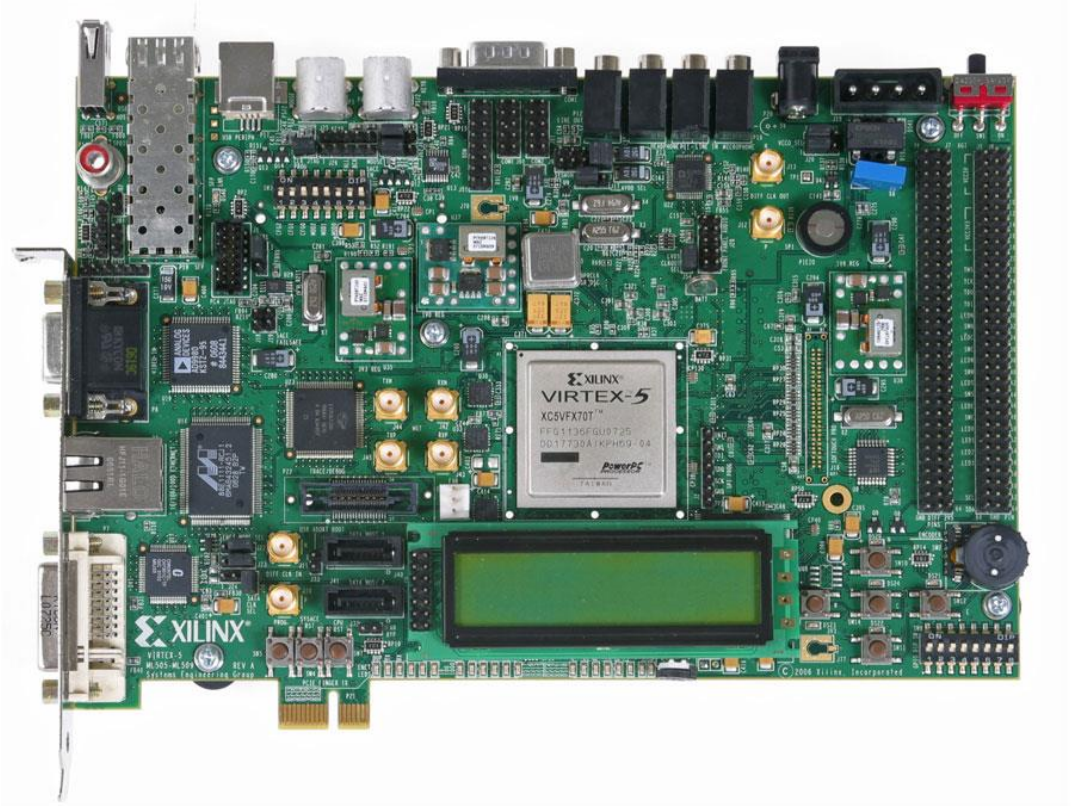

Figure 4.1: An image of Xilinx ML507 evaluation board[39]

4.1 System Overview

The proposed system is an embedded solution, which is fully implemented in ML507 platform. However, data and commands are needed to be feed to ML507 board; therefore, a computer is connected to the system over RS232 link for this purpose. RS232 is preferred due to its simple protocol. There are several advantages of this protocol in the development process. It is not error-prone due to its simplicity; therefore, it is known that any malfunction is not related to data transfer step. It is also easy to modify; therefore, in any unexpected case, parameters can be monitored on PC.

In the development and verification steps, detection and recognition processes are desired to be controlled to detect and correct odd cases and bugs. Therefore, frames are sent one by one and frame data are written in the SRAM, first. Then detection and recognition steps are initiated with a command given to processor. When the process is initiated, rest of the steps is same with a system operating with streaming data. Therefore, this system can be used with streaming data just changing its data interface architecture.

The system is fed with 640x480 pixels images. These images are sent from PC to the system according to progressive scan. In this way, starting from the first row, all rows of the image are sent sequentially. Incoming data enter to interface module, first. In this module, data is decoded according to used protocol. In this case, it is RS232 UART. After this part, decoded data enter to processor for controlled process. There are command and hand shaking procedures between PC and processor for each command. With the initiation of command of write frame data, processor transfers all of the frame data to memory control module and these data are written to SRAM. After all of the frame data are written in the SRAM, with the initiation process data command, memory control module is set to transfer frame data from SRAM to color segmentation module. During the transfer process, data are transferred to segmentation module with the same order they are written to SRAM. By this means, simultaneous buffering and segmentation of frame data can be achieved with little change in system. In the segmentation module, 640x480 sized frame data are converted to 80x60 sized red and blue and 20x15 sized yellow color maps.

Blue and red maps are read by Detection and Classification module. In this module, blobs in the red and blue color maps are detected. Width and height values of the blobs are detected. Moreover, these values and their ratio are compared with predetermined constants and each region is labeled as suitable or not. When operation is done, end of operation signal is asserted and this initiates operation of the processor.

In the processor part of the system, PPC440 system generated by Irmak [34] is utilized. This architecture is preferred because his C codes for Hough Transform and IPP are utilized in the edge detection and sign recognition parts. To achieve the same recognition rate and timing results, this procedure is chosen.

Processor starts its process by reading coordinates and color information of the regions labeled as suitable. After that, these regions are merged according to algorithm given in chapter 2.2. Second port of the yellow color map is directly connected to processor. When merging operation is completed, processor reads data from the yellow color map and checks whether there is a yellow sign under the region or not. After that, positions of the signs are compared to check whether they are on the same vertical line. Finally, if a region does not fulfill any other conditions, area under that region is searched for a sign pole. For this operation, processor transfers pixels of searched area to edge detection filter implemented in hardware. When filtering operation is done, processor reads edge data and searches for vertical lines. When lines are detected, related regions are marked as ROI and rest of the regions are discarded.

ROIs are read from SRAM and transferred to DDR2 SDRAM. After this point, it should be noted that hardware part of the system is ready for another frame. By this means, a new frame can be processed for region detection and buffered in SRAM. However, this case is applicable if system is fed with streaming data and it is not applicable for this thesis. Each region is read from DRAM and segmented, first. After that, edges are extracted and with the application of Hough Transform, parameters of the edges are obtained. Finally, these parameters are evaluated and IPP method is applied for recognition of the signs.

High level diagram of the designed system for implementation of operations stated above is presented in the Figure 4.2. In the following parts, implementation details of the detection modules and recognition algorithms are presented.

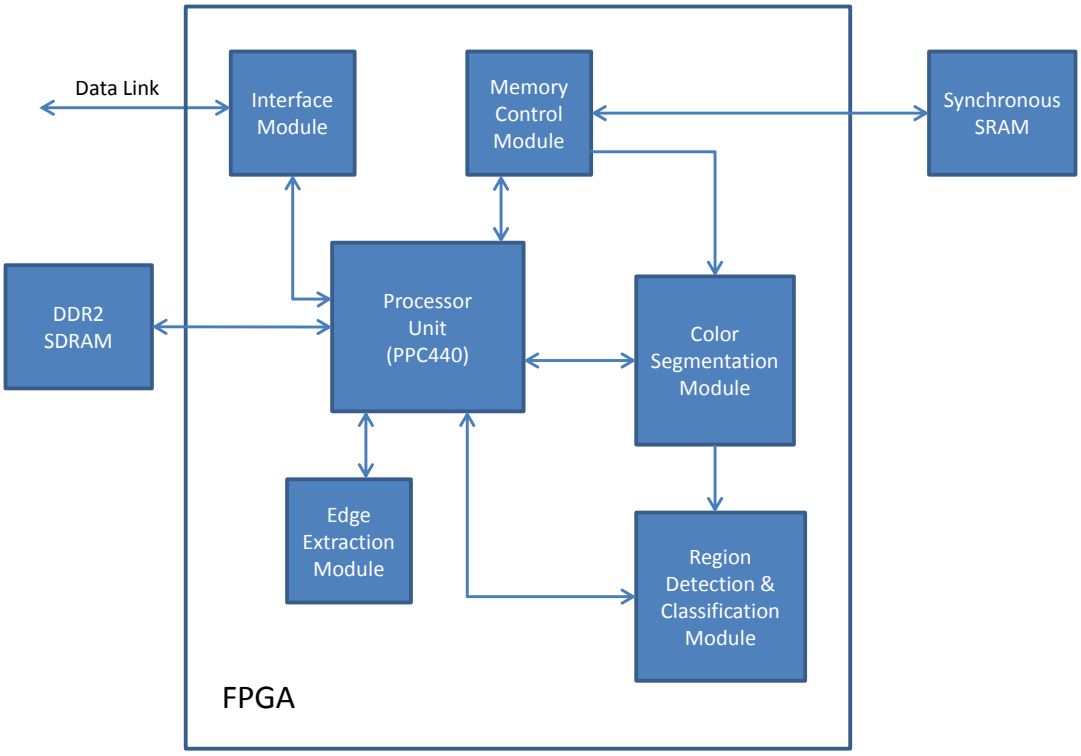

Figure 4.2: High level diagram of the traffic sign detection and recognition system

4.2 Details of the Detection System

Detection of the signs in a frame starts with the color segmentation process. As it is stated above, data are sent to color segmentation module according to progressive scan. This detail is very important; because, at each step position of the processed pixel must be known to establish position relations of the pixels properly. Hardware architecture presented Figure 4.3 is used for color segmentation and downsampling operation.

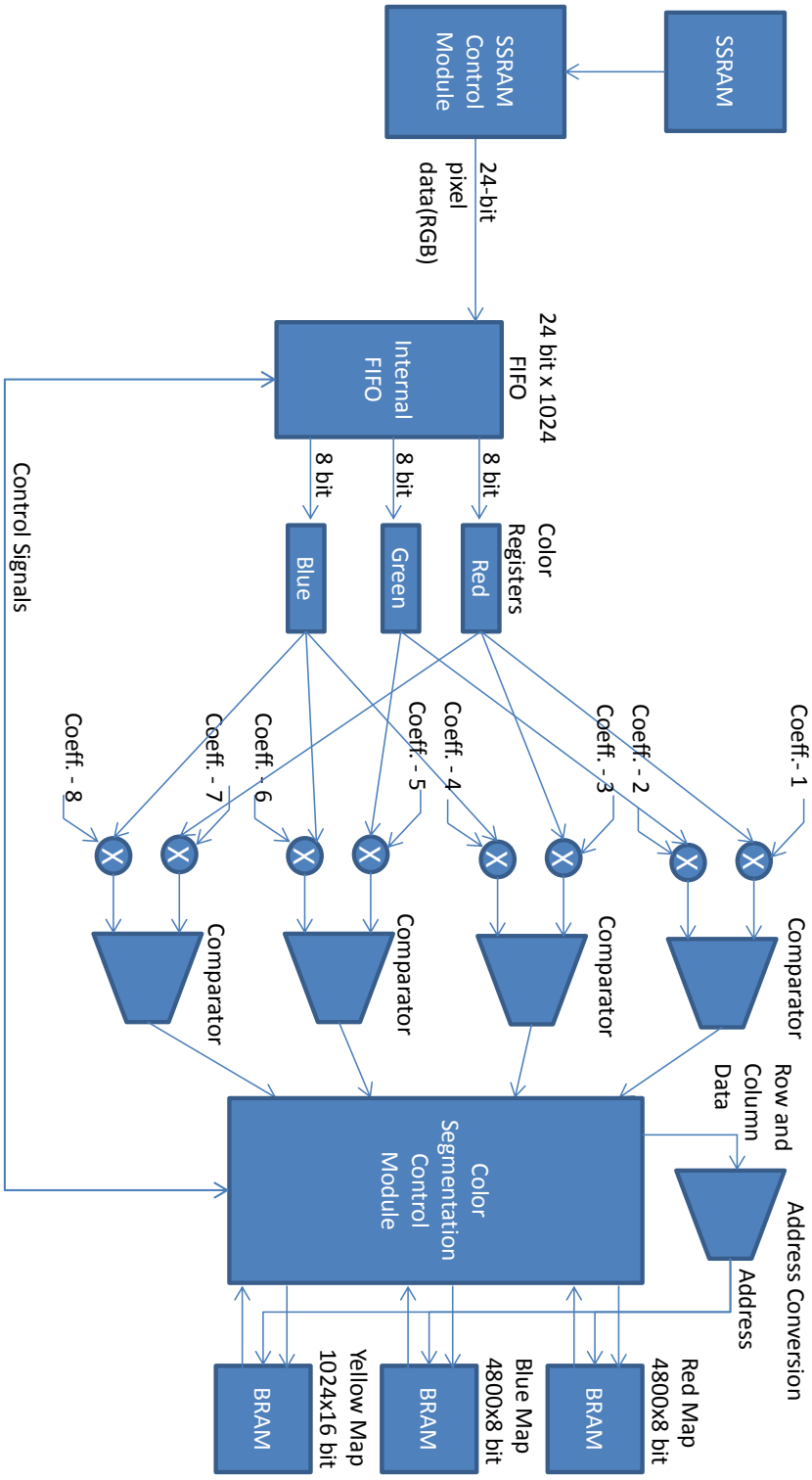

Figure 4.3: Hardware architecture used for color segmentation and downsampling

With initiation of the detection process, memory control module starts to transfer frame data (900 kB) to detection module. There is a FIFO (24-bit x 1024) at the beginning of the detection state. In the hardware design, FIFOs are usually preferred for connecting two asynchronous domains; because, they can use different clock signals at read and write domain as presented in Figure 4.4.

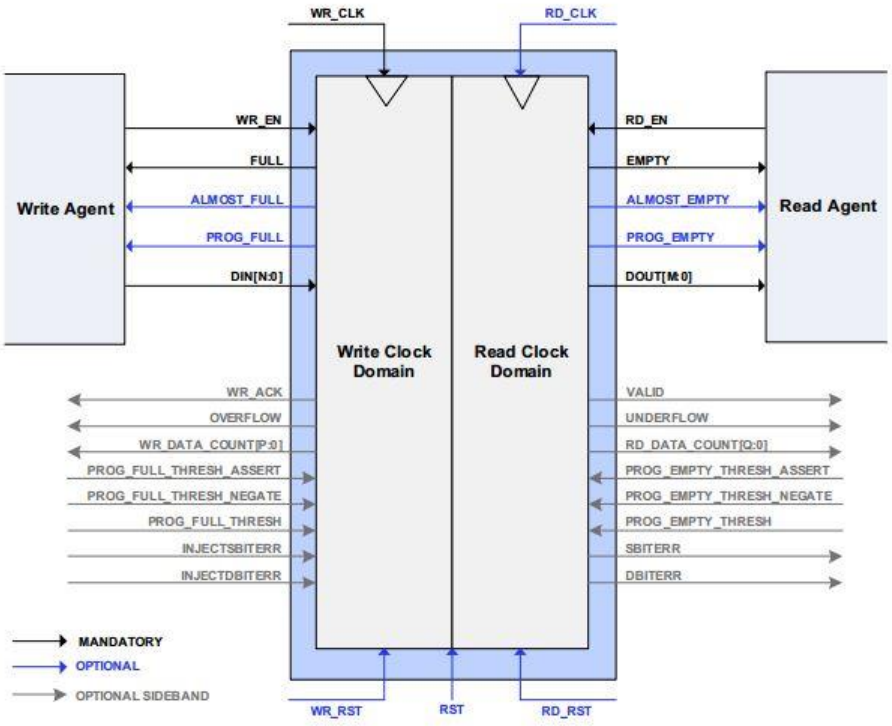

Figure 4.4: Architecture of a FIFO[40]

As SRAM controller module transfers data to FIFO via write domain, "EMPTY" signal is de-asserted. Segmentation control module checks "EMPTY" signal and as long as it is de-asserted, control module reads data until frame data is finished. Frame data are written to FIFO and read from it in 24-bit format (8-bit for each color component of a pixel). After pixel is read from the FIFO, each color component is written to a different 8-bit register. Values in these registers are used to segment color of the pixel according to equation 10. However, there are division operations in the segmentation equation and they need to be converted in a suitable form for hardware implementation. Multiplication operation is more suitable for hardware operations. Therefore, when red to green ratio is required to be checked whether over 1.5 or not, twice of red value can be compared with three times of

green one. By this means, equation 18 can be rearranged as equation 19, where v and t are integers.

$$\frac{\mathrm{G(x,y)}}{\mathrm{R(x,y)}} \leq \mathrm{G'_b} \;\&\; \mathrm{G'_b} = \frac{v}{t} \tag{18}$$

$$t * \mathrm{G(x,y)} \leq v * \mathrm{R(x,y)} \tag{19}$$

Therefore, to find out whether ratios of the colors are over the thresholds or not, color values are multiplied with coefficients and then fed into comparators. Outputs of the comparators are delivered to segmentation control module. In this module, results are placed in segmentation equations and the pixel is labeled as either one of the three colors or irrelevant color.

With the color segmentation of the pixels, frame should be downsampled. For this operation each 8x8 region is converted to one pixel for red and blue color maps. For yellow color map, each 32x32 region converted to one pixel information. Downsampling operations are done at the same time with the conversion operation. It is stated above that row and column information of each processed pixel is known. This is done by generating a row and a column counters. Both of the counters are set to zero at the beginning of each segmentation process. With the read of each pixel from the FIFO, column counter is increased by one until 639. When column counter is 639 and a new pixel is read, column counter is set to zero and row counter is increased by one. This process is continued until the column counter is 639 and row counter is 479. In order to generate column and row counters 10-bit and 9-bit registers are required. These registers are combined in a way that every pixel in the same 8x8 area corresponds to a single address for blue and red color maps. For this purpose, the least significant three bits are discarded and rest of the registers is concatenated as presented in Figure 4.5. When yellow map is generated, the least significant five bits are discarded to obtain a combined address.

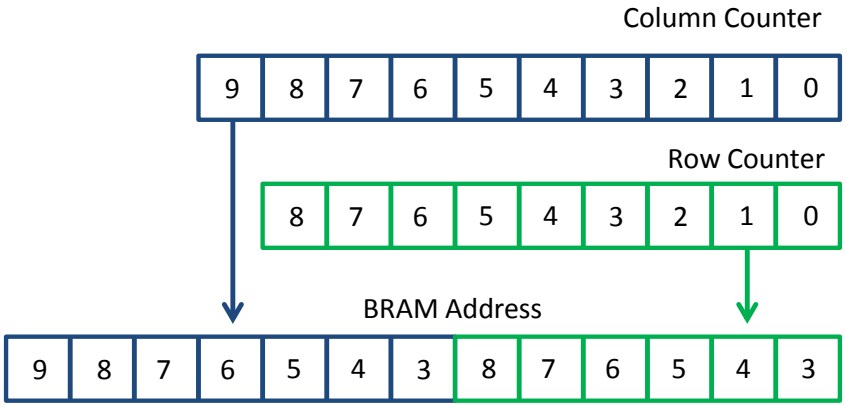

Figure 4.5: Concatenation of row and column counters for blue and red color maps

Obtained data as a result of concatenation is used for feeding address port of Block RAM (BRAM). BRAMs are build-in structures of FPGAs. They are small in size; however, they have very low latency and high operation speed. Moreover, BRAMs have dual control ports and they can be reached by two different modules without needing any arbitration. Therefore, they are highly suitable for keeping data of color maps.

At the beginning of the each process, data in the every address of BRAMs are set to zero. Every time a pixel gives positive result for one of the three searched colors, value at the corresponding address of the corresponding color map is increased by one. By this means, number of pixels with a certain color at each region is obtained as given in the Figure 4.6.

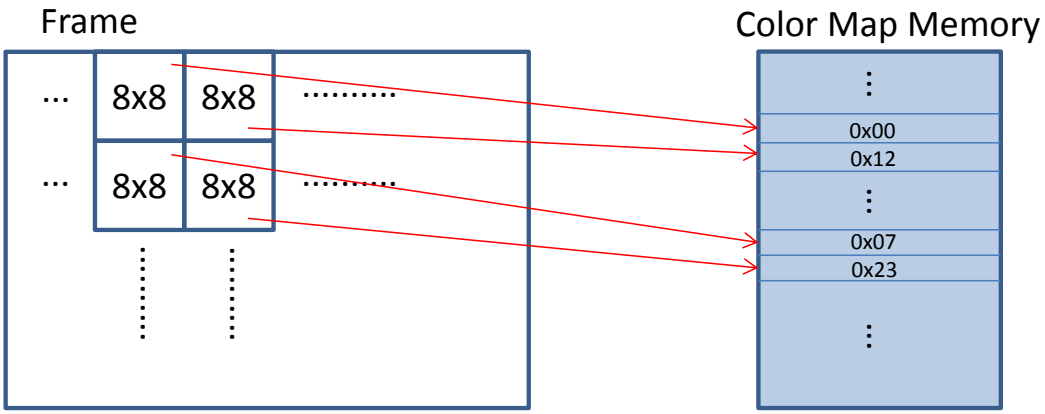

Figure 4.6: Mapping of 8x8 regions to an address of the BRAM

When color maps are generated, parallel architecture of FPGA is used, intensively. Color segmentation operations for interested colors are conducted simultaneously. Furthermore, all color maps are updated simultaneously and completed a few clock cycles after from the point the last pixel of the frame is sent to system. This saves us from spending too much time for repeating same operation for different colors. This approach is also employed for detection of blobs. When segmentation operation is completed, “end of segmentation” signal is asserted and this signal is connected to detection and classification module.

Detection and classification module starts its operation by reading blue and red map data from the BRAMs. In the beginning of read operation column and row addresses are set to zero and these values are increased according to progressive scan. After that, generated addresses are converted to BRAM address in accordance with write order. Each time an address of BRAM is read, this address contains a value between the ranges of 0-64. This value is fed into a comparator and compared with the threshold value (30 % of the block). If the value is less than the threshold, it is discarded and module reads the next address of the BRAM. Otherwise, address of the pixel is used to detect blobs via algorithm presented

in chapter 2.2 and results are written to ROI Array. After the read operation, content of the address is set to zero to eliminate initialization process of BRAMs. General architecture of this module is presented in Figure 4.7.

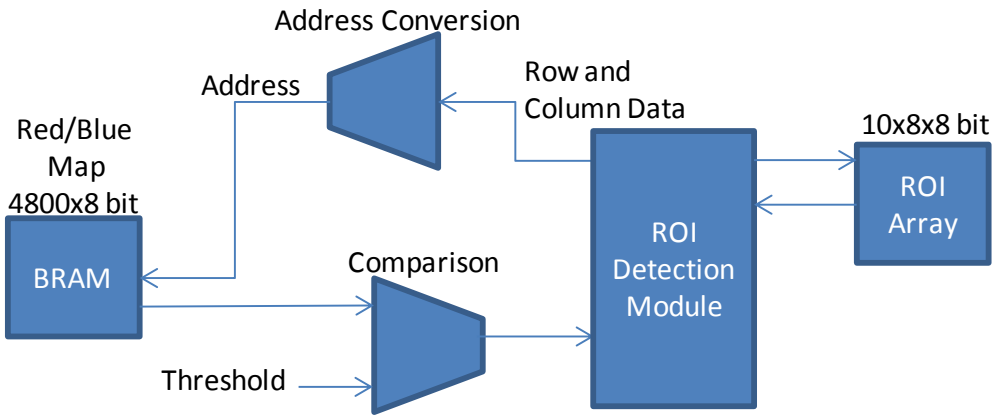

Figure 4.7: Architecture of ROI detection module

The most important thing in this operation is to evaluate neighboring relations in a parallel manner. If loop is defined for this operation and operations are conducted in a sequential manner, capabilities of an FPGA are not fully utilized. Therefore, the following architecture is utilized. First of all, if BRAMs are used as ROI arrays, only one entry can be reached at each cycle. For full parallelism, it is desired to reach all entries at once. Therefore, ROI array is built with registers. After many trials ten entries is seen to be enough. Every entry is composed of eight registers as presented in Figure 4.8.

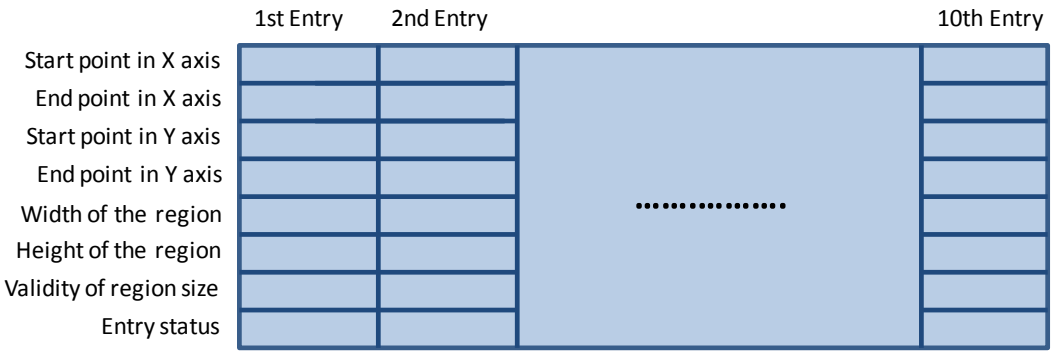

Figure 4.8: Structure of a ROI array

Two arrays are generated for region detection process. One of the arrays is used for red regions (red color map) and the other is used for blue ones. Names of most of the registers describe their purposes. However, purposes of validity of region size and entry status registers are need to be elaborated. Validity of region size register defines whether size of the region in the predefined borders or not. Implementation details of this register are presented below.

Entry status register shows whether the entry is active or not. At the beginning of the process, status registers of all entries are set to zero, which shows all entries are inactive. Furthermore, an index for each array is generated and this index show the entry number, which will be activated in case of necessity. When a valid pixel is read, coordinates of the pixel is compared with the coordinates of the active entries. If neighboring relation is found between the pixel and an active entry, relation found flag for that entry is set and coordinates of the entry are updated. At the next clock cycle, logical operation is applied to relation found flags. If result is zero, then valid pixel is neighbor to previously detected regions and entry shown by the index signal is activated. Furthermore, index value is increased by one.

By means of presented parallel architecture, in five clock cycles evaluation of neighboring relation of a valid pixel is completed and values in the array are updated as required. Moreover, processes for blue and red color maps are conducted, in a parallel manner. By this means, detection algorithm is speeded up several times.

Several requirements for validity of a detected region are presented in chapter 2.2. First part of these requirements is about the shape of the region and these requirements can be observed using coordinates of the shape. At every clock cycle, coordinates of regions are used to update width and height of the shape as given in the Figure

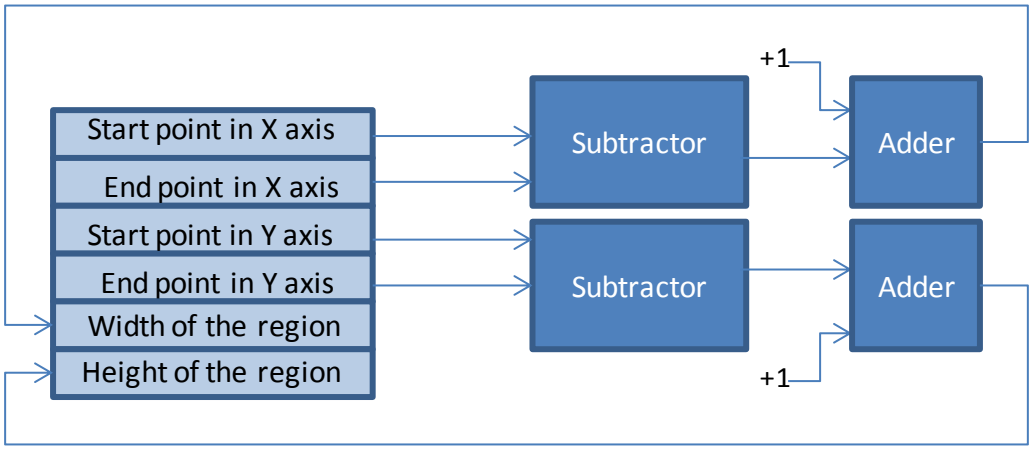

Figure 4.9: Calculation of width and height of each entry

After calculation of width and height parameters, it should be checked whether they are appropriate to determined limits. For this operation, ratio and length of the width and height parameters need to be controlled. Height to width ratio should be between the range of (0.5-2); therefore height and width registers are copied to other registers with right shift operation and half of the size values are obtained in these registers. To detect height to width ratio, width value is compared with the half of the height value and vice versa. Furthermore, width and height values are compared with threshold values to detect whether they are in the range of (40-240) pixels. Finally all of the comparison results are used to feed an "AND" gate, and result of this operation is transferred to "validity of region size" register. Implementation of this structure is presented in Figure 4.10. Moreover, it should be noted that presented architectures are implemented separately for each entry. Therefore, they do not inhibit operation of each other. Furthermore, validity of region size register is updated whenever coordinates of the region changes and update operation takes five clock cycles. By this means, validity of the regions is finalized at most five cycles after the end of detection operation.

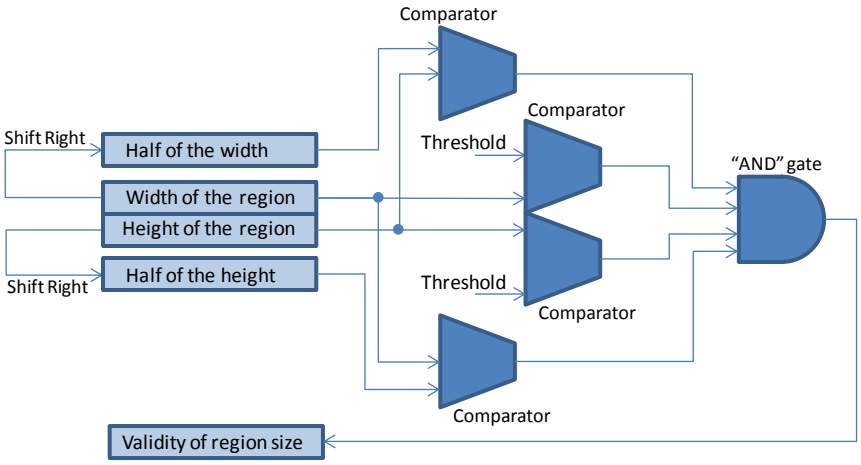

Figure 4.10: Implemented structure to assign "Validity of region size" register

After proper sized regions are labeled as valid, coordinates and color information of these shapes are transferred to PPC440 processor for rest of the operations in the classification step. When transfer operation is completed, merging operation is applied to these regions. For this operation coordinates of all regions is compared with each other. If intersection area of two regions is larger than 70 % of the smaller region, two regions are combined as given in Figure 4.11. In combination operation, smaller region is labeled to be discarded in further operations and coordinates of the larger region is updated to cover both regions.

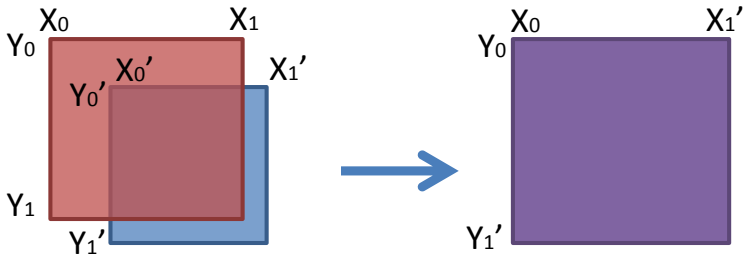

Figure 4.11: Merging operation of two regions

After merging operation, yellow signs are searched under the ROI. Yellow map is generated by hardware in the segmentation step. One port of the BRAM, which includes yellow map, is connected to segmentation module, and the other one is connected to processor. To search a yellow map, coordinates of the area below the ROI is detected and data in this area are read one by one. If one of the read data is greater than the threshold value, it is decided that there is a yellow sign below the ROI. Therefore, this region is marked as a traffic sign and no more classification operation is applied to this region.

Next step in the classification process is to evaluate positions of the ROIs. For this operation, coordinate of each ROI is compared with every region. In the comparison process, first center of the ROI is calculated. Then this point shifted up and down by the length of the controlled ROI. If the shifted point is enclosed by a region, then both regions are marked as traffic sign.

The last classification step is detection of a pole under the sign. In order to achieve this, edge detection operation is applied to a 60x60 area below the ROI. For the edge detection operation, LoG filtering is preferred due its superior results to other filters in MATLAB trials. Moreover, this filter is implemented in hardware to reduce computation load over processor. For filtering operation, a 9x9 kernel stored in a BRAM and interested region is transferred to another BRAM by processor. Then processor initiates filtering operation. When the operation is initiated, filtering module starts to read data from both coefficient RAM and image RAM. These coefficients are multiplied via dedicated multipliers in the FPGA and the results of multiplications are accumulated to obtain the result of a pixel. This operation is applied to all pixels with the architecture presented in Figure 4.12.

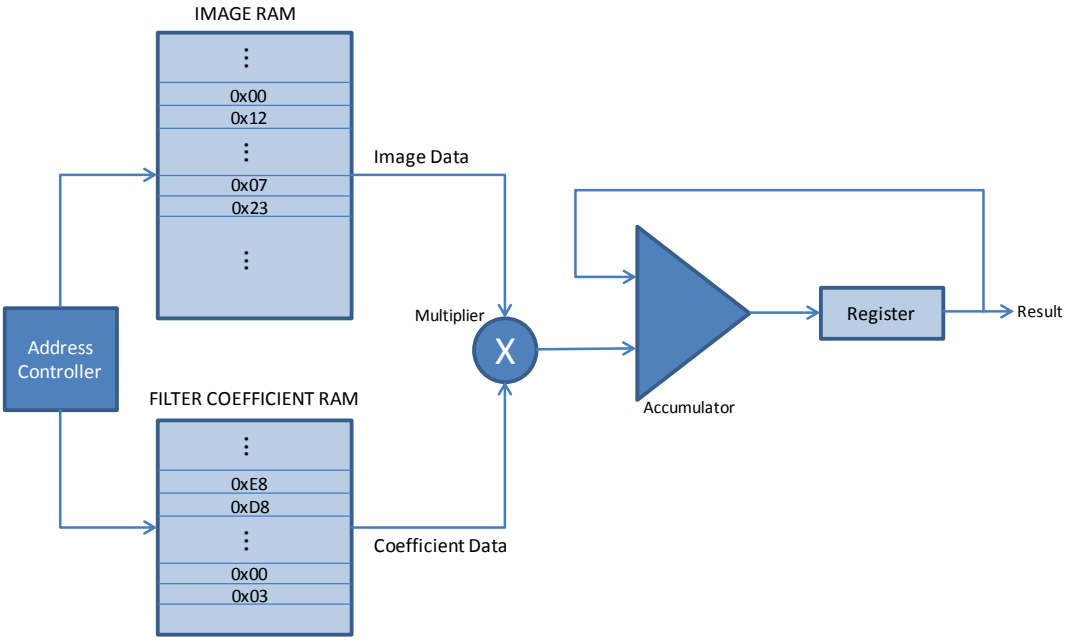

Figure 4.12: Hardware implementation of the filter

Result of filtering operation should be refined. For this purpose, results are fed through a comparator, first. Comparator reduces 8-bit results to 1-bit according to their values being less or larger than zero. Moreover, to detect zero crossings, equation 10 is implemented in hardware as given in the Figure 4.13.

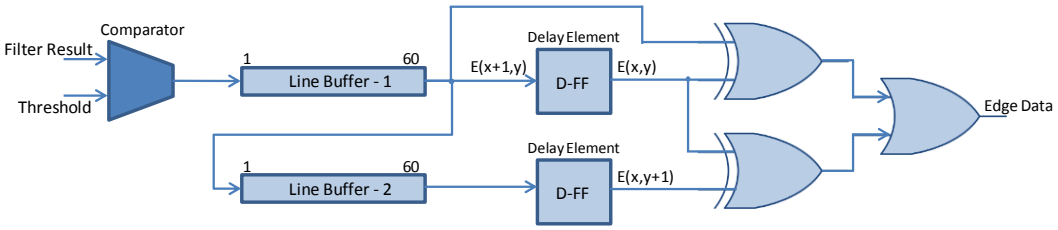

Figure 4.13: Hardware implementation of zero crossing detector

Obtained edge information is transferred back to processor. From this information, a histogram in X axis is generated. After that, it is counted that how many vertical lines lie along at least 75 % of the image. If the count of these lines exceeds 2 (there exist 2 parallel lines), ROI is marked as traffic sign.

Pole detection operation is the last operation of the classification process. After this operation, recognition operation is applied to regions marked as traffic sign and the other regions are discarded.

4.3 Details of the Recognition System

First operation in the recognition process is interpolation of the images to a common size. It is stated in chapter 3.1 that the nearest neighbor is the optimum solution for this system. There is no need to complex calculations for this algorithm. Furthermore, this algorithm can be applied during data are read from the SRAM.

First of all, coordinates of the shape are extended from all sides by 8 pixels. This is required, because during detection of the shapes, border of the shapes can be trimmed due to downsampling process. Then coordinates of the sign are updated to equalize width and height of the region. After that, width and height is increased to the nearest multiple of ten. Therefore, possible width and length values become a multiple of ten between the range of (40-240). By this means, sign width and height can become one of the 21 values (40, 50… 240) and a previously prepared mapping array is installed to the system. This array is filled such that when width of the region is divided by ten, result indicates which entry of the array should be used. Offset values are placed in the entries of this array. In the data reading process from the SRAM, if 2nd pixel is retrieved, value in the second position of the entry is added to the starting position and data in the obtained address is read. To make the structure of the array clear, 12th and 13th entry of the array is given in the Figure 4.14. After the interpolation of the image, edge image is obtained with the implementation of equation 10 and 13 in the C code.

	1	2	3	4	5	6	…
12th Entry	0	2	4	6	8	10	…
13th Entry	0	2	4	7	9	11	…

Figure 4.14: 12th and 13th entries of the mapping array

After extraction of edges, positions and other features are obtained. Three different methods are used for this purpose. These are Linear Hough Transform, Circular Hough Transform and horizontal/vertical histogram extraction.

In the implementation of the Linear Hough Transform an accumulator array is generated. X axis of this array is defined by the angle parameter (Theta, θ) and Y axis is defined by the distance parameter (Rho, r). It is stated in chapter 3.2.2 that lines only around the

angles of 0, 60 and 120 are searched. Therefore, to make voting operation, “for loops” are generated with angle parameters and equation 13 is calculated with position of valid pixel and angle information. Distance variable is obtained as a result of this computation. Finally, angle and calculated distance variables are used to decide which entry of the array is increased. This calculation and voting process is repeated for all valid pixels in the edge image and accumulator array is completed. Plot of an accumulator array generated for a triangular sign is presented in Figure 4.15. After the generation of the array, the largest hits are searched around the desired angles and parameters of the three lines are obtained.

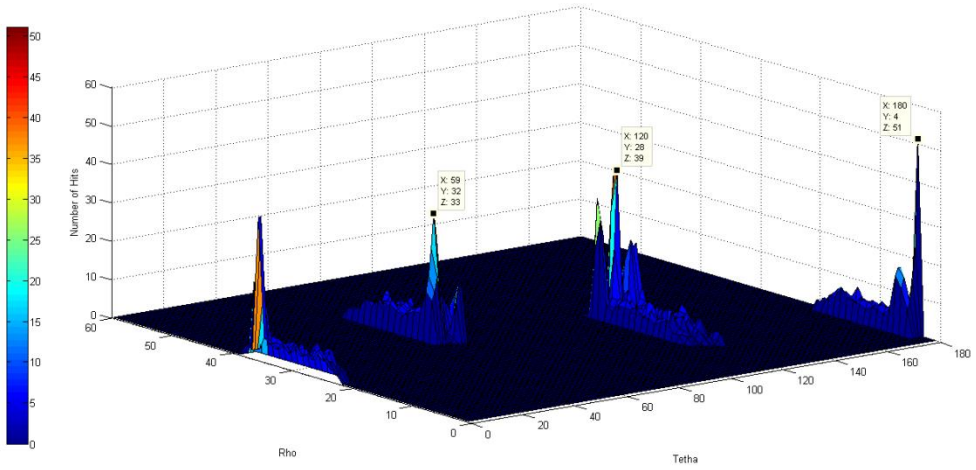

Figure 4.15: Plot of an accumulator array generated by application of linear Hough Transform to a triangular sign

Implementation of circular Hough Transform takes the calculation complexity one step further. For a circular shape, equation 17 is calculated and there are four unknowns in this case. Therefore, three nested “for loops” are generated for this process and the r parameter is calculated for different values of x_c, y_c and k. Calculation of r value is repeated for all valid pixels in the edge image and another accumulator array is generated. It should be noted that three nested loops cause too much computation and some restrictions are applied to decrease computation load. Center coordinate of the circular sign is limited to 20x20 area at the center of the image. Moreover, ten different values are considered for k parameter. As a result of parameter selection, size of generated accumulator array becomes 30x4000 and plot of this array generated by application of circular Hough Transform to a circular sign is presented in Figure 4.16.

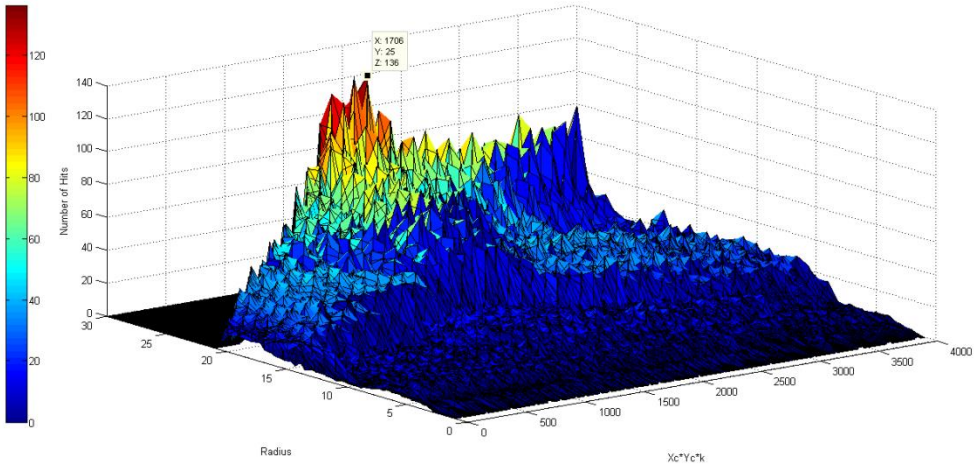

Figure 4.16: Plot of an accumulator array generated by application of circular Hough Transform to a circular sign

Similar to linear Hough Transform, maximum hit for the array is searched after the generation of the accumulator. Index in the Y axis of the maximum entry is the radius of the circular sign and x_c, y_c and k parameters are determined from the maximum entry in the X axis.

Furthermore, in chapter 3.2.5, consistency of the result of Hough Transform is stated. In Figure 4.16, radius of the detected circle is 25, in this case its circumference is expected to be around 150. By this means, 136 hits give us a very reasonable result. On the other hand, accumulator array of the application of circular Hough Transform to a rectangular sign is presented in the Figure4.17. Radius of detected circle is 16 in this case; therefore a number of hit around 100 is expected. However, 43 hits cause inconsistency between hits and detected radius.

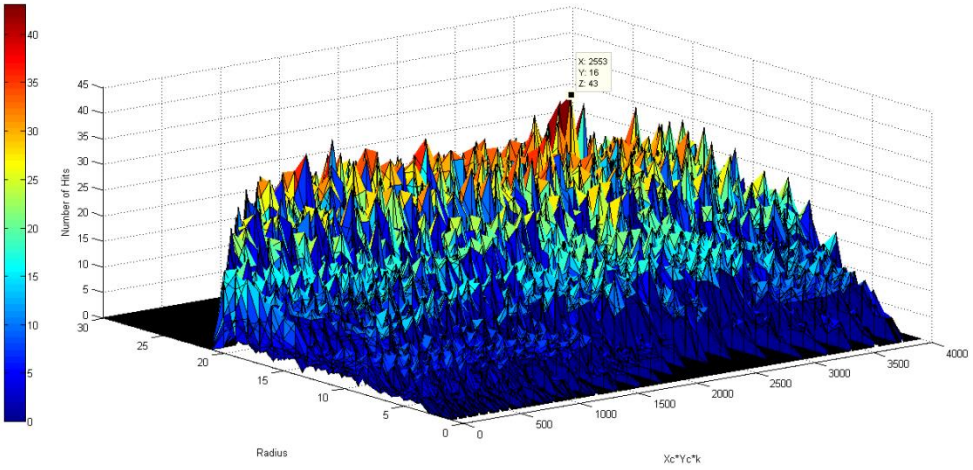

Figure 4.17: Plot of an accumulator array generated by application of circular Hough Transform to a triangular sign

Last detection operation is to detect borders of the rectangular signs. Borders of rectangular signs aligned with only horizontal and vertical axes. Therefore, histograms are extracted in the horizontal and vertical axes and borders are detected from maximum hits.

After detection operations, border information is obtained and inner regions of the traffic signs are extracted. Opening filter is applied to these regions to eliminate noisy structures. After that obtained shape is divided into regions for IPP operation. In the IPP operation, a counter is generated for each region. Each counter is increased by one with the detection of a pixel in the search area. After generation of all counters are finished, counter values are normalized with total pixel number.

These normalized values are extracted from corresponding region values of the template shapes and absolute value of the differences are added for each template. After this operation a sum is obtained for each sign template and the template giving minimum sum is accepted as result.

CHAPTER 5

EXPERIMENTAL RESULTS

FPGAs are powerful platforms to process data; however they are not the most suitable and the fastest platforms to develop algorithm. Generation of programming file of a complete system like our system takes almost one hour. Because of this reason, observing result of any change in the system consumes too much time. Therefore, all algorithms are implemented on MATLAB without using high level MATLAB commands. By this means, generated algorithm can be mapped to FPGA with minimum change. Test results are obtained in the both platforms and it is observed that they show consistency. Although, MATLAB and FPGA systems give consistent results, objective of the MATLAB implementation is only development of the algorithm faster. Therefore, only results obtained from the FPGA system are presented in this chapter.

5.1 Test Results and Explanations

A computer interface is needed to test the detection and the recognition system. This computer interface is required to send image data and commands and display the results sent by the system. For this purpose, GUI presented in Figure 5.1 is implemented in MATLAB.

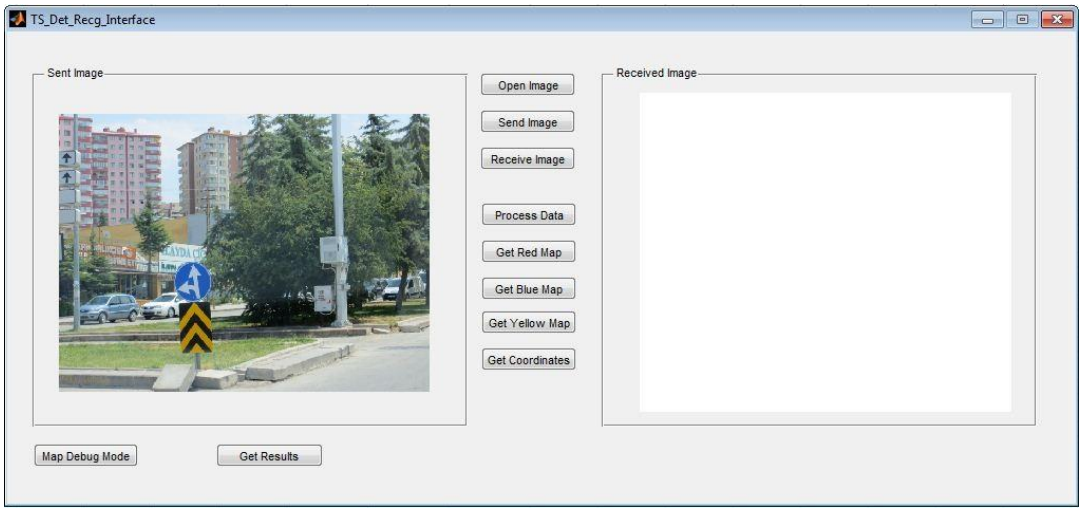

Figure 5.1: Computer interface used to test the system

First of all, an image is selected in this interface and sent to FPGA board. Then process is initiated and results are obtained. When get results command is initiated results appear on the command window of the MATLAB.

Using the interface, 137 images, which include 162 traffic signs, are sent to the detection and recognition system. These traffic signs are selected between the signs included in the database, in which there are 5 blue circular, 5 blue rectangular, 9 red circular and 14 red triangular signs. Test images are captured with Sony DSC-W30 and Canon SX260 cameras. These images are taken either in a moving car or on still points at the different hours of the daytime. Therefore, effects of the illumination variations can also be observed. Detection and recognition rates of the system are introduced separately in Table 5.1 and Table 5.2.

Table 5.1: Detection rate of the system

	Triangular Signs (Red)	Circular Sign (Red)	Circular Signs (Blue)	Rectangular Signs (Blue)
Number of Signs	43	66	39	14
Number of Detected Signs	38	59	36	13
Detection Rate	88.4 %	89.4 %	92.4 %	92.8 %
Number of False Detections	0	0	0	1

Table 5.2: Recognition rate of the system

	Triangular Signs (Red)	Circular Sign (Red)	Circular Signs (Blue)	Rectangular Signs (Blue)
Number of Signs	38	59	36	13
Number of Recognized Signs	34	50	35	13
Recognition Rate	89.4 %	84.7 %	97.2 %	100 %

In the experiments, it is observed that 146 of the 162 traffic signs are detected. Moreover, 132 of the detected signs are recognized. These results correspond to 90.1 % detection and 90.4 % recognition rates. Therefore, overall success of the system becomes 81.4 %. On the other hand, the system shows remarkable results on false detection rate. In 137 images, false detection is occurred only in one. This corresponds to less than 1 % in the false detection rate. This false detection rate is achieved by means of exhaustive region classification methods. In Figure 5.2, images which include deceptive regions are presented. These images are given to the system and deceptive regions are eliminated successfully.

Figure 5.2: Images processed successfully despite of the existence of deceptive objects

Besides eliminating deceptive object, vibration in the camera also can be tolerated to some extent. Due to system use segmented images in detection and the recognition stages, images similar to presented images in Figure 5.3 can be detected and recognized without any problem. Moreover, detection and recognition systems can tolerate perspective distortions to some extend as presented in Figure 5.4.

Figure 5.3: Detected and recognized signs although images are distorted due to vibration

Figure 5.4: Detected and recognized signs although images are affected from perspective distortion

On the other hand, preferred methods have some disadvantages. Searching poles under the signs decrease false detection rate. However, if there is no pole below the sign or it is occluded, traffic signs are discarded as irrelevant regions. Possible reasons which prevent detection of poles are presented in Figure 5.5.

Figure 5.5: Possible cases which prevents detection of traffic signs

Furthermore, detection of the traffic signs is also affected by color segmentation stage. Using relative values of the RGB components increase immunity of the system against illumination changes. This is tested with the images taken different hours of daytime. However, it is also observed that reflection of the light from the traffic sign, existence of too little light and other factors that cause change in light of the traffic sign deteriorate success of the color segmentation. Furthermore, effects of the illumination changes cause more significant impacts on segmentation of the red color than blue color. It is also

observed that color of the worn out red signs turn to tones of the orange and this prevents detection of these signs with color segmentation. In Figure 5.6, traffic signs cannot detected due to color changes are presented.

Figure 5.6: Traffic signs which cannot be detected due to color change and illumination

Besides stated problems above, sign detection stage cannot discriminate vertically overlapped signs. After color segmentation, these signs generate a united region in the color map. Therefore, detection module treat this signs as one region and if their length becomes more than twice of their width, they are discarded. If this region meets aspect ratio specification, signs are handled as if it is a single sign and recognition stage gives wrong result. A sample image of this case is presented in Figure 5.7.

Figure 5.7: Vertically overlapped signs that prevents detection

Besides detection stage, recognition stage also affects overall success of the system. In the IPP algorithm, triangular, circular and rectangular signs are divided into 6, 5 and 4 parts, respectively. These low numbers of regions make calculation of the IPP parameters easier and faster. However, some signs give almost same IPP parameters for each region. Therefore, for healthy operation some signs cannot exist in the same database. "Fifty Kilometers Speed Limit" and "No Overtaking" signs are an example to this situation. Although they do not have similar appearance, they have same IPP parameters and cannot exist in the same database.

Moreover, low number of regions cause wrong recognition results as number of the signs in the database increases. This case can be observed in Table 5.2. Blue signs have lower number of templates in the database than red signs; therefore, their recognition rate becomes much better then red ones.

5.2 Resource Utilization and Execution Time

In the FPGA architecture the basic resources are logic and memory elements. Virtex-V FPGAs group logic elements in a specific manner and name each group as slice. Therefore, occupied slice and BRAM ratio gives idea about resource utilization of the FPGA.

Presented FPGA system is implemented via Xilinx ISE 14.4 design software. It is observed that presented system consumes 40 % of the slices and 25 % of the BRAMs in the FPGA. Detailed resource utilization of the system is given in the Table 5.3.

Table 5.3: Resource utilization of the traffic sign detection and recognition system

Device Utilization Summary			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	7,681	44,800	17%
Number used as Flip Flops	7,681		
Number of Slice LUTs	8,959	44,800	19%
Number used as logic	8,667	44,800	19%
Number using O6 output only	8,055		
Number using O5 output only	146		
Number using O5 and O6	466		
Number used as Memory	270	13,120	2%
Number used as Dual Port RAM	122		
Number using O6 output only	6		

Table 5.3: Resource utilization of the traffic sign detection and recognition system(Cont'd)

Number using O5 output only	32		
Number using O5 and O6	84		
Number used as Single Port RAM	4		
Number using O6 output only	4		
Number used as Shift Register	144		
Number using O6 output only	144		
Number used as exclusive route-thru	22		
Number of route-thrus	193		
Number using O6 output only	163		
Number using O5 output only	28		
Number using O5 and O6	2		
Number of occupied Slices	4,540	11,200	40%
Number of LUT Flip Flop pairs used	12,140		
Number with an unused Flip Flop	4,459	12,140	36%
Number with an unused LUT	3,181	12,140	26%
Number of fully used LUT-FF pairs	4,500	12,140	37%
Number of unique control sets	827		
Number of slice register sites lost to control set restrictions	1,793	44,800	4%
Number of bonded IOBs	201	640	31%
Number of LOCed IOBs	199	201	99%
IOB Flip Flops	341		
Number of BlockRAM/FIFO	38	148	25%
Number using BlockRAM only	35		
Number using FIFO only	3		
Number of 36k BlockRAM used	32		
Number of 18k BlockRAM used	3		
Number of 36k FIFO used	3		
Total Memory used (KB)	1,314	5,328	24%
Number of BUFG/BUFGCTRLs	8	32	25%
Number used as BUFGs	8		

Table 5.3: Resource utilization of the traffic sign detection and recognition system(Cont'd)

Number of IDELAYCTRLs	3	22	13%
Number of BUFIOs	8	80	10%
Number of DCM_ADVs	1	12	8%
Number of DSP48Es	4	128	3%
Number of PLL_ADVs	1	6	16%
Number of PPC440s	1	1	100%
Average Fanout of Non-Clock Nets	3.78		

This system is operating at 125 MHz clock frequency at the hardware (logic) side and 200 MHz at the processor side. Execution time of the hardware side is independent of the number of traffic signs in the image. It is only dependent to clock frequency of the system and the image size. On the other hand, processor side is highly dependent to content of the 60x60 region after color segmentation. Number of valid pixels affects computation time of the Hough Transforms. In this system, size of the sign in the 60x60 region is dependent to several variables. Therefore, the worst case scenarios cannot be generated by our will. To obtain, worst case computation times of the Hough Transform and IPP calculations Irmak's worst case measurements are used [34]. This usage causes no problem because same system and C codes are used and different time for the same operation cannot be expected. Measured and calculated execution times of the system for a single sign are presented in Table 5.4.

Table 5.4: Execution time of the system for a single sign

Type of Sign / Operation	Red Signs	Blue Signs
Hardware Color Segmentation and Downsampling	17.203 ms	
Region Detection and Size Classification	0.230 ms	
Classification Operations	0.150 ms	
Pole Detection	4.578 ms	

Table 5.4: Execution time of the system for a single sign (Cont'd)

Interpolation with Memory Read	0.288 ms	
Color Segmentation of 60x60 Region	0.549 ms	
Edge Extraction	0.323 ms	
Edge Detection and Sign Recognition	75.765	56.310
Data Transfer and Miscellaneous Operations	11.000 ms	
Total Hardware Time	17.433 ms	
Total Processor Time	92.658 ms	73.203 ms
Total Latency	**110.091 ms**	**90.636 ms**

Maximum latency of the system becomes almost 110 ms. A car with 70 km/h speed can go 2.2 m in 110 ms. Capturing a new image in every 2.2 m is highly convenient and minimizes the probability of missing a sign. Furthermore, hardware and processor systems can operate in a parallel manner. Therefore, if system architecture is changed accordingly, a new frame can be processed at every 92.658 ms.

CHAPTER 6

CONCLUSION & FUTURE WORK

A study over traffic sign detection and recognition system is presented in this thesis. This system is designed for FPGA implementation with a combination of several FPGA implementation and image processing techniques. Implemented system is tested with 640x480 sized outdoor images.

In this system, search regions in the images are reduced with color properties of traffic signs. Therefore, first, color segmentation is applied to input image. As a result of this operation, red, blue and yellow color maps are obtained. Blue and red color maps are divided into 8x8 blocks and yellow map is divided into 32x32 blocks. To mark a block as valid, it is required that at least 30 % of the pixels in that block are segmented as the required color. After this operation each block is treated as a pixel. Therefore, blue and red maps are downsampled to 80x60 and yellow map is downsampled to 20x15.

After these operations, blue and red maps are used for blob detection. After blobs are detected, they are classified according to their shapes. Suitable regions need to be meet one of the three requirements. These requirements are having a yellow sign below, being vertically neighbor to another sign and having a pole below. When regions meet one of these requirements they are treated as traffic signs.

Obtained sign regions are interpolated into a 60x60 region. To extract edges of the sign color segmentation is applied and crossing points are labeled as edge points. Hough Transform is applied to the edge points and contour information is obtained. Finally, IPP operation is applied inner region of the traffic sign and sign is recognized [34].

In the FPGA implementation process of the algorithm, system is generated as a combination of hardware (logic) system and process system. In the segmentation, region detection and filtering processes, parallel processing algorithms can be used efficiently. Therefore, these processes are implemented in the FPGA hardware. On the other hand, other algorithms require implementation of several loops; therefore, processor system is used for execution of these algorithms.

To test succession of the system, 137 images with 162 signs are processed with the FPGA system. At the end of the test procedure, 90.1 % detection and 90.4 % recognition rates are achieved with a processing speed around 10 fps. After examining test images and results robust and weak points of the system are discovered. The robust features of the system can be listed as follows:

- The system can detect and recognize multiple signs in a single image.
- Colors of the signs can be segmented successfully under the variable illumination of day time.
- Signs can be detected with a high detection rate; although, false detection rate is less than 1 %.
- IPP algorithm can handle perspective distortion to a certain extent.
- Distortion caused by vibration of the camera can be handled successfully.

Observed weak points of the system enumerated as follows:

- Red color segmentation algorithm is sensitive to reflection and color changes of worn out signs.
- Overlapped signs cannot be discriminated
- Signs are discarded if sign poles cannot be detected

Implemented system is highly suitable for improvements. High compatibility between previously implemented algorithms by Irmak [34] and rest of the system is achieved by using ML507 platform. However, Virtex-5 FPGA family is introduced to market in 2006 [41] and more advanced products, like Zynq family of Xilinx, are unveiled. In Zynq family, FPGA platform includes dual Cortex A9 processors with more logic elements. Moreover, Cortex A9 processor can operate at 667 MHz.

In the future work, proposed system is planned to transfer to Zynq platform. By this means, proposed system can be executed faster and additional algorithms can be included to eliminate weak points of the system.

Moreover, in the next version of the system, it is planned to work with video instead of images.

REFERENCES

[1] "Automobile Safety." [Online]. Available: https://en.wikipedia.org/wiki/Automobile_safety. [Accessed: 22-Jul-2013].

[2] "Bosch Automotive Technology." [Online]. Available: http://www.bosch-automotivetechnology.com/en/com/driving_safety_com/driving_safety_systems_for_passenger_cars_com/driving_safety_systems_for_passenger_cars_com.html.

[3] "Trafik kazalari özeti 2012" , from "http://www.kgm.gov.tr/Sayfalar/KGM/SiteTr/Yayinlar/Yayinlar.aspx", retrieved on Aug 2013.

[4] J. Park, J. Kwon, J. Oh, S. Lee, J. Kim, and H. Yoo, "A 92-mW Real-Time Traffic Sign Recognition System With Robust Illumination Adaptation and Support Vector Machine," vol. 47, no. 11, pp. 2711–2723, 2012.

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

[6] Z. Chen, J. Yang, and B. Kong, "A Robust Traffic Sign Recognition System for Intelligent Vehicles," *2011 Sixth International Conference on Image and Graphics*, pp. 975–980, Aug. 2011.

[7] S. Waite and E. Oruklu, "FPGA-Based Traffic Sign Recognition for Advanced Driver Assistance Systems," *Journal of Transportation Technologies*, vol. 2012, 2013.

[8] A. Broggi, P. Cerri, and P. Medici, "Real time road signs recognition," *Intelligent Vehicles …*, no. section III, pp. 981–986, 2007.

[9] L. Chen, Q. Li, M. Li, and Q. Mao, "Traffic sign detection and recognition for intelligent vehicle," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, no. Iv, pp. 908–913.

[10] K. Ganapathi, V. Madumbu, R. Rajendran, and S. David, "Design and implementation of an automatic traffic sign recognition system on TI OMAP-L138," *2013 IEEE International Conference on Industrial Technology (ICIT)*, pp. 1104–1109, Feb. 2013.

[11] J. F. Khan, S. M. a. Bhuiyan, and R. R. Adhami, “Image Segmentation and Shape Analysis for Road-Sign Detection,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 1, pp. 83–96, Mar. 2011.

[12] A. Ruta, F. Porikli, S. Watanabe, and Y. Li, “In-vehicle camera traffic sign detection and recognition,” *Machine Vision and Applications*, vol. 22, no. 2, pp. 359–375, Dec. 2009.

[13] T. Harasthy and J. Turan, “Optical correlator based Traffic Signs Recognition,” *Systems, Signals and …*, no. April, pp. 11–13, 2012.

[14] F. Zaklouta and B. Stanciulescu, “Real-time traffic sign recognition in three stages,” *Robotics and Autonomous Systems*, Aug. 2012.

[15] W. Liu, J. Lv, H. Gao, B. Duan, H. Yuan, and H. Zhao, “An efficient real-time speed limit signs recognition based on rotation invariant feature,” *2011 IEEE Intelligent Vehicles Symposium (IV)*, no. Iv, pp. 1000–1005, Jun. 2011.

[16] M. Garcia-Garrido, “Fast traffic sign detection and recognition under changing lighting conditions,” *… , 2006. ITSC’06. IEEE*, pp. 811–816, 2006.

[17] C. Keller and C. Sprunk, “Real-time recognition of US speed signs,” *Intelligent Vehicles …*, pp. 518–523, 2008.

[18] N. Barnes, a. Zelinsky, and L. S. Fletcher, “Real-Time Speed Sign Detection Using the Radial Symmetry Detector,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 2, pp. 322–332, Jun. 2008.

[19] F. Moutarde and A. Bargeton, “Robust on-vehicle real-time visual detection of American and European speed limit signs, with a modular Traffic Signs Recognition system,” *Intelligent Vehicles …*, vol. 51, no. 33, pp. 1122–1126, 2007.

[20] “Trafik İşaretleri Elkitabı I”, from "http://www.kgm.gov.tr/Sayfalar/KGM/SiteTr/Yayinlar/Yayinlar.aspx", retrieved on Aug 2013.

[21] M. S. Prieto and A. R. Allen, “Using self-organising maps in the detection and recognition of road signs,” *Image and Vision Computing*, vol. 27, no. 6, pp. 673–683, May 2009.

[22] a. de la Escalera, L. E. Moreno, M. a. Salichs, and J. M. Armingol, “Road traffic sign detection and classification,” *IEEE Transactions on Industrial Electronics*, vol. 44, no. 6, pp. 848–859, 1997.

[23] M. Benallal and J. Meunier, “Real-time color segmentation of road signs,” *CCECE 2003 - Canadian Conference on Electrical and Computer Engineering. Toward a Caring and Humane Technology (Cat. No.03CH37436)*, vol. 3, pp. 1823–1826, 2003.

[24] M. a. Souki, L. Boussaid, and M. Abid, “An embedded system for real-time traffic sign recognizing,” *2008 3rd International Design and Test Workshop*, pp. 273–276, Dec. 2008.

[25] U. L. Jau, “A comparison of RGB and HSI color segmentation in real - time video images: A preliminary study on road sign detection,” *2008 International Symposium on Information Technology*, pp. 1–6, 2008.

[26] A. Fog, “Optimization Manuals, Instruction Tables,” 2012. [Online]. Available: http://www.agner.org/optimize/instruction_tables.pdf. [Accessed: 31-Jul-2013].

[27] T. P. Cao and G. Deng, “Real-Time Vision-Based Stop Sign Detection System on FPGA,” *2008 Digital Image Computing: Techniques and Applications*, pp. 465–471, 2008.

[28] B. Hoferlin and K. Zimmermann, “Towards reliable traffic sign recognition,” *2009 IEEE Intelligent Vehicles Symposium*, pp. 324–329, Jun. 2009.

[29] A. Ruta, Y. Li, and X. Liu, “Real-time traffic sign recognition from video by class-specific discriminative features,” *Pattern Recognition*, vol. 43, no. 1, pp. 416–430, Jan. 2010.

[30] M. Shi, H. Wu, and H. Fleyeh, “A Robust Model for Traffic Signs Recognition Based on Support Vector Machines,” *2008 Congress on Image and Signal Processing*, pp. 516–524, 2008.

[31] S. Maldonado-bascón, S. Lafuente-arroyo, P. Gil-jiménez, H. Gómez-moreno, and F. López-ferreras, “Road-sign detection and recognition based on support vector machines,” *Intelligent …*, vol. 8, no. 2, pp. 264–278, 2007.

[32] C. Fang, S. Chen, and C. Fuh, “Road-sign detection and tracking,” *Vehicular Technology, IEEE …*, vol. 52, no. 5, pp. 1329–1341, 2003.

[33] Z. Zheng and H. Zhang, “Robust traffic sign recognition and tracking for Advanced Driver Assistance Systems,” *… Systems (ITSC), 2012 15th …*, pp. 704–709, 2012.

[34] H. Irmak, “REAL TIME TRAFFIC SIGN RECOGNITION SYSTEM ON FPGA,” no. September, 2010.

[35] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Third. Pearson Education, 2008.

[36] R. Fisher, S. Perkins, A. Walker, and E. Wolfart, "Laplacian/Laplacian of Gaussian," 2003. [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm. [Accessed: 09-Aug-2013].

[37] J. Illingworth and J. Kittler, "The adaptive Hough transform," *Pattern Analysis and Machine …*, no. 5, pp. 690–698, 1987.

[38] "Detection of Circles and Ellipses With The Hough Transform (Biomedical Image Analysis)." [Online]. Available: http://what-when-how.com/biomedical-image-analysis/detection-of-circles-and-ellipses-with-the-hough-transform-biomedical-image-analysis/. [Accessed: 13-Aug-2013].

[39] "Virtex-5 FXT FPGA ML507 Evaluation Platform." [Online]. Available: http://www.xilinx.com/products/boards-and-kits/HW-V5-ML507-UNI-G.htm. [Accessed: 15-Aug-2013].

[40] Xilinx, "LogiCORE IP FIFO Generator v9.2 Product Guide," 2012. [Online]. Available: http://www.xilinx.com/support/documentation/ip_documentation/fifo_generator/v9_2/pg057-fifo-generator.pdf.

[41] "Xilinx Unveils 65nm Virtex-5 Family - Industry's Highest Performance Platform FPGAs." [Online]. Available: http://www.xilinx.com/prs_rls/2006/silicon_vir/0657v5family.htm.