

REAL TIME TRAFFIC SIGN RECOGNITION SYSTEM ON FPGA

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

BY

HASAN IRMAK

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

SEPTEMBER 2010

Approval of the thesis

REAL TIME TRAFFIC SIGN RECOGNITION SYSTEM ON FPGA

submitted by **Hasan IRMAK** 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. İsmet Erkmén
Head of Department, **Electrical and Electronics Engineering** _____

Assoc. Prof. Dr. Mehmet Mete Bulut
Supervisor, **Electrical and Electronics Engineering Dept., METU** _____

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

Examining Committee Members:

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

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

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

Assist. Prof. Dr. İlkay Ulusoy
Electrical and Electronics Engineering Dept., METU _____

Emre Ulay
MGEO, ASELSAN _____

Date: 02.09.2010

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 : Hasan Irmak

Signature :

ABSTRACT

REAL TIME TRAFFIC SIGN RECOGNITION SYSTEM ON FPGA

Irmak, Hasan

M. Sc., Department of Electrical and Electronics Engineering

Supervisor: Assoc. Prof. Dr. Mehmet Mete Bulut

Co-Supervisor: Prof. Dr. Gözde Bozdağı Akar

September 2010, 78 pages

In this thesis, a new algorithm is proposed for the recognition of triangular, circular and rectangular traffic signs and it is implemented on an FPGA platform. The system can recognize 32 different traffic signs with high recognition accuracy.

In the proposed method, first the image is segmented into red and blue regions, and according to the area of the each segment, the dominant color is decided. Then, Laplacian of Gaussian (LoG) based edge detection is applied to the segmented image which is followed by Hough Transform for shape extraction. Then, recognition based on Informative Pixel Percentage (IPP) matching is executed on the extracted shapes.

The Traffic Sign Recognition (TSR) system is implemented on Virtex 5 FX70T FPGA, which has an embedded PPC440 processor. Some modules of TSR

algorithm are designed in the FPGA logic while remaining modules are designed in the PPC440 processor. Work division between FPGA and PPC440 is carried out considering their capabilities and shortcomings of FPGA and processor. Benefits of using an FPGA with an embedded processor are exploited to optimize the system.

Keywords: Color Segmentation, Shape Extraction, Traffic Sign Recognition, FPGA, Embedded Processor

ÖZ

FPGA ÜZERİNDE GERÇEK ZAMANLI TRAFİK İŞARETİ TANIMA SİSTEMİ

İrmak, Hasan

Yüksek Lisans, Elektrik Elektronik Mühendisliği Bölümü

Tez Yöneticisi : Doç. Dr. Mehmet Mete Bulut

Ortak Tez Yöneticisi: Prof. Dr. Gözde Bozdağı Akar

Eylül 2010,78 sayfa

Bu tezde, üçgen, çember ve dikdörtgen trafik işaretlerinin tanınması üzerine yeni bir algoritma önerilmiş ve FPGA ortamında gerçekleştirilmiştir. Bu sistem 32 farklı trafik işaretini yüksek tanıma yüzdesi ile tanıyabilmektedir.

Önerilen yöntemde, önce resim kırmızı ve mavi renklere bölünmüş ve her bölünme sürecindeki bölünme alanına göre, baskın renge karar verilmiştir. Sonra, bölünmüş resme “Laplacian of Gaussian” kenar tanıma algoritması uygulanmış ve Hough Dönüştürücü şekil çıkarma sürecinde kullanılmıştır. Daha sonra, Bilgilendirici Piksel Yüzdesi (BPY) eşleştirmeye dayalı tanıma çıkarılan şekillere uygulanmıştır.

Trafik İşareti Tanıma (TİT) Sistemi, içinde gömülü PPC440 işlemci olan Virtex 5 FX70T FPGA’inde tasarlanmıştır. Bazı TİT modülleri FPGA mantık kapıları kullanılarak geriye kalan modüller ise PPC440 işlemcisi kullanılarak tasarlanmıştır. FPGA ve PPC440 arasındaki görev dağılımı bunların yetenek ve eksiklikleri göz önünde bulundurularak yapılmıştır. Sistemi optimize etmek için, FPGA ve gömülü işlemciyi beraber kullanmanın sağladığı faydalardan yararlanılmıştır.

Anahtar Kelimeler: Renk Bölümleme, Şekil Çıkarımı, Trafik İşareti Tanıma, FPGA, Gömülü İşlemci

To My Family

ACKNOWLEDGEMENTS

First of all, I wish to express my deepest thanksgiving to my supervisors Assoc. Prof. Dr. Mehmet Mete Bulut and Prof. Dr. Gzde Bozdađı 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 Meltem Savař, Erkan Okuyan, Hseyin Irmak and all my friends for their support and fellowship.

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

I would like to express my special appreciation to my family for their continuous support and encouragements.

TABLE OF CONTENTS

ABSTRACT	IV
ÖZ.....	VI
ACKNOWLEDGEMENTS	IX
TABLE OF CONTENTS	X
LIST OF TABLES	XIII
LIST OF FIGURES	XIV
LIST OF ABBREVIATIONS	XVI
CHAPTERS	
1. INTRODUCTION.....	1
1.1 SCOPE OF THESIS	3
1.2 THESIS OUTLINE	3
2. BACKGROUND ON TRAFFIC SIGN RECOGNITION	5
2.1 INTRODUCTION.....	5
2.2 RECOGNITION OF TRAFFIC SIGNS	5
2.2.1 <i>COLOR SEGMENTATION</i>	6
2.2.2 <i>SHAPE EXTRACTION</i>	10
2.2.3 <i>RECOGNITION</i>	12
3. SHAPE EXTRACTION AND RECOGNITION	16
3.1 SHAPE EXTRACTION	17
3.1.1 <i>COLOR SEGMENTATION</i>	17
3.1.2 <i>EDGE DETECTION</i>	19
3.1.2.1 LAPLACIAN OF GAUSSIAN FILTERING	19

3.1.2.2	ZERO CROSSING DETECTION	21
3.1.3	<i>SHAPE BASED DETECTION</i>	22
3.1.3.1	TRIANGULAR SIGN DETECTION	23
3.1.3.2	CIRCULAR SIGN DETECTION	27
3.1.3.3	RECTANGULAR SIGN DETECTION	30
3.2	RECOGNITION	31
3.2.1	<i>BINARY THRESHOLDING</i>	32
3.2.2	<i>MORPHOLOGICAL OPERATIONS</i>	33
3.2.3	<i>RECOGNITION USING INFORMATIVE PIXEL PERCENTAGE</i> <i>FINDING AND MATCHING</i>	34
3.2.3.1	TRIANGULAR SIGN RECOGNITION	35
3.2.3.2	CIRCULAR SIGN RECOGNITION	36
3.2.3.3	RECTANGULAR SIGN RECOGNITION	37
4.	RECOGNITION OF TRAFFIC SIGNS USING FPGA HARDWARE	39
4.1	HARDWARE ARCHITECTURE	39
4.1.1	<i>XILINX ML507 DEMO BOARD</i>	40
4.1.2	<i>USED COMPONENTS AND FUNCTIONS</i>	41
4.1.2.1	VIRTEX5-FX70T FPGA	41
4.1.2.2	XILINX PLATFORM FLASH PROM	41
4.1.2.3	LINEAR FLASH CHIP	42
4.1.2.4	DDR2 SDRAM	43
4.1.2.5	16X2 CHARACTER LCD	43
4.1.3	<i>SYSTEM OVERVIEW</i>	44
4.2	FPGA ARCHITECTURE	46
4.2.1	<i>INTRODUCTION</i>	46
4.2.2	<i>COLOR SEGMENTATION</i>	48
4.2.3	<i>EDGE DETECTION</i>	48
4.2.3.1	LOG FILTERING	48
4.2.3.2	ZERO CROSSING DETECTOR	50
4.2.4	<i>SHAPE EXTRACTION</i>	52
4.2.4.1	TRIANGULAR SHAPE EXTRACTION	52

4.2.4.2 CIRCULAR SHAPE EXTRACTION	55
4.2.4.3 RECTANGULAR SHAPE EXTRACTION.....	57
4.2.5 <i>BINARY THRESHOLDING</i>	58
4.2.6 <i>MORPHOLOGICAL OPERATIONS</i>	58
4.2.6.1 EROSION	59
4.2.6.2 DILATION	59
4.2.7 <i>IPP FINDING AND MATCHING</i>	60
4.2.8 <i>LCD INTERFACE</i>	61
5. EXPERIMENTAL RESULTS.....	62
5.1 INTRODUCTION	62
5.2 TEST RESULTS	62
5.3 RESOURCE USAGE.....	65
5.4 EXECUTION TIMES	68
6. CONCLUSIONS AND FUTURE WORK.....	70
6.1 CONCLUSIONS	70
6.2 FUTURE WORK	72
REFERENCES.....	73
APPENDICES	
A. ML507 DEVELOPMENT BOARD	76
B. GUI OF TSR SYSTEM.....	78

LIST OF TABLES

TABLES

Table 2-1 Features of traffic signs	5
Table 2-2 Color enhancement values (normalized to 255)	9
Table 4-1 Truth Table used to find the edges in x direction	51
Table 4-2 Truth Table used to find the edges in y direction	51
Table 4-3 Truth Table used to find the edges in the image	51
Table 5-1 Recognition Results of Set-1 Images	64
Table 5-2 Recognition Results of Set-2 Images	64
Table 5-3 Wrong Recognitions	65
Table 5-4 Resource Utilization of the Traffic Sign Recognition System	65
Table 5-5 Execution Times of the TSR System.....	69

LIST OF FIGURES

FIGURES

Figure 2-1 Structure of Single SVM.....	6
Figure 2-2 M-SVMs structure.....	7
Figure 2-3 Angular Spacing of 4 bin HoG.....	15
Figure 3-1 The Steps of the Proposed System	16
Figure 3-2 Traffic Sign and Red Segmented Image	18
Figure 3-3 Traffic Sign and Blue Segmented Image	19
Figure 3-4 Laplacian Kernel	20
Figure 3-5 Laplacian of Gaussian Function.....	20
Figure 3-6 Laplacian of Gaussian Kernel	21
Figure 3-7 Edge Image without Color Segmentation	22
Figure 3-8 Edge Image with Color Segmentation	22
Figure 3-9 Representation of a line with (r, θ) parameters	24
Figure 3-10 Mapping from Cartesian Coordinates to Polar Coordinates	25
Figure 3-11 Edge Image of a Triangular Traffic Sign	26
Figure 3-12 Detected Lines after Applying Hough Transform	26
Figure 3-13 Detected Circle After Applying CHT	27
Figure 3-14 Effect of the "k" parameter.....	28
Figure 3-15 Detected Ellipse after Applying Ellipse Detection	29
Figure 3-16 Rectangular Sign and Edge Image	30
Figure 3-17 x and y Projections of the Edge Image.....	31
Figure 3-18 Output of Binarization Process	33
Figure 3-19 Structuring Elements used in Dilation and Erosion	34
Figure 3-20 Binary Image before and after Opening	34
Figure 3-21 Divided Regions of Triangular Sign	35
Figure 3-22 Classification of Circular Signs.....	36

Figure 3-23 Divided Regions of Circular Sign	37
Figure 3-24 Divided Regions of Rectangular Sign	38
Figure 4-1 ML507 FPGA Development Board	40
Figure 4-2 Flash Interface Timing Diagram	42
Figure 4-3 Timing Diagram of the Character LCD	43
Figure 4-4 Hardware Architecture of the System	44
Figure 4-5 Flowchart of the Power Up Initialization Sequence.....	45
Figure 4-6 Block Diagram of Traffic Sign Recognition System	47
Figure 4-7 Kernel of LoG Edge Detector	49
Figure 4-8 Hardware Block Diagram of LoG Filter Block.....	50
Figure 4-9 Hardware Block Diagram of Zero Crossing Detector.....	52
Figure 4-10 Block Diagram of the LCD Interface	61
Figure 5-1 An example Traffic Sign from Set-1	63
Figure 5-2 An example Traffic Sign from Set-2.....	63
Figure A-1 Block Diagram of ML507 Board	76
Figure B-1 TSR GUI.....	78

LIST OF ABBREVIATIONS

ADAS	: Advanced Driver Assistance Systems
CHT	: Circular Hough Transform
CSS	: Curvature Scale Space
DDR	: Double Data Rate
FPGA	: Field Programmable Gate Array
HIS	: Hue Saturation Intensity
HLS	: Hue Luminance Saturation
HOG	: Histogram of Oriented Gradients
IPP	: Informative Pixel Percentage
LDA	: Linear Discriminant Analysis
LOG	: Laplacian Of Gaussian
LUT	: Look Up Table
RGB	: Red Green Blue
ROI	: Region of Interest
SAD	: Sum of Absolute Difference
SDRAM	: Synchronous Dynamic Random Access Memory
SVM	: Support Vector Machine
TSD	: Traffic Sign Detection
TSR	: Traffic Sign Recognition

CHAPTER 1

INTRODUCTION

In the new century, automotive industry has been showing great improvements in the area of car electronics. Electronic systems are being used extensively and they equip cars with more intelligence. The term Advanced Driver Assistance Systems (ADAS) is used for these high technology intelligent systems. ADAS helps the driver to drive the car and leads to a better awareness on the road. Thus, ADAS makes cars and roads safer for both drivers and pedestrians [1]. Some examples of these systems are [2],

- Lane Departure Warning
- Night Vision
- Automatic Parking
- Blind Spot Detection
- Traffic Sign Recognition

The last example, Traffic Sign Recognition System (TSR), is a computer vision application and it supports drivers to follow the restrictions and obey the regulations via utilization of image processing techniques. The system recognizes the traffic signs and warns the driver about the sign [3].

First generation TSR Systems appeared in the late 2008 in luxury cars such as BMW 7 Series. These systems could only recognize speed limits. In second-generation systems, some restriction signs are added to the systems. It is announced that 2011 Volkswagen Phaeton can also detect overtaking restriction signs [4].

Current studies and trends show that, traffic sign recognition systems will probably be capable of recognizing all the traffic signs in the future [2].

Traffic Signs have standards in shapes and colors that are defined by the governments and they remain unchanged within country. Drivers easily recognize them because colors and shapes of the signs are very different from the natural environment. The dominant color in urban environments is green whereas the used colors are blue and red in the traffic signs. Moreover, the shapes of the signs are triangle, circle or rectangle, which are difficult to see in the nature. These two unique features are used to build efficient detection and recognition algorithms. In the published works, the general approach consists of two main parts. First, the traffic sign is localized in a smaller region with the help of color and shape features of the sign. This stage is called detection. In the second stage, recognition stage, using these shape and color information, the sign is recognized. TSR Systems use these beneficial features of the signs to recognize them. However, there are some problems while processing the images. Some of the main problems are the followings,

- Illumination affects the color analysis.
- Occlusion affects the shape analysis.
- Weather conditions such as rain, snow, or fog, affect the shape extraction.
- Physically damaged or changed surface metal of traffic signs affects the recognition.

Various techniques are developed in order to eliminate these problems and make the traffic sign recognition system more robust [5][6][7][8]. However, adding robustness means the increase in complexity which does not allow the systems run in real time. Therefore, efficient algorithms are required for reliable TSR systems.

1.1 SCOPE OF THESIS

In this thesis, recognition of the traffic signs in 60x60 image patches is studied and implemented on FPGA.

First, the algorithms for the TSR have been searched and analyzed. Then, a new algorithm is proposed which has similar performance with the available algorithms but computationally more efficient. Finally, the algorithm is implemented on Virtex 5 FX70T FPGA.

The developed algorithm consists of two main parts, namely shape extraction and recognition. In the first part, using the shape information, Region of Interest (ROI) is extracted. The extraction of ROI is based on color segmentation, edge detection and border detection. After color segmented image is found, edge detection algorithm is applied. Using Hough Transform, triangle/circle/rectangle is found in the edge image. Then, ROI is extracted using triangle/circle/rectangle shape information. In the second part, ROI is divided into regions and the IPP is calculated for these regions. Candidate sign is compared with all template signs using Sum of Absolute Differences (SAD) of IPPs. Template sign which has the minimum SAD with candidate sign becomes the recognition result.

Algorithm is developed and verified using MATLAB before the FPGA implementation carried out. After a satisfactory performance is achieved, it is implemented on Virtex 5 FX70T FPGA. In order to test the TSR system, a Graphical User Interface (GUI) is designed to load 60x60 images and send the shape information. This GUI works as a Traffic Sign Detection (TSD) module. The detailed description of the developed GUI is given in Appendix B.

1.2 THESIS OUTLINE

In Chapter 2, literature review for color segmentation, shape extraction and sign recognition is given.

In Chapter 3, the developed algorithm for shape extraction and traffic sign recognition is explained in detail.

In Chapter 4, the FPGA implementation of the algorithm is explained from both hardware and software point of view. The division of the workload and communication between hardware and software in the FPGA is given in this chapter. Hardware implemented modules are explained using block diagrams and software implemented modules are explained using pseudo codes.

In Chapter 5, the test results of the FPGA implementation are given. MATLAB implementation results are not presented since FPGA and MATLAB implementations have almost identical results except minor differences. In this chapter, the execution time of each process used in the algorithm is also given.

Chapter 6 gives the conclusion and the future work.

CHAPTER 2

BACKGROUND ON TRAFFIC SIGN RECOGNITION

2.1 INTRODUCTION

The shapes and color features of the traffic signs attract the drivers' attention easily. These features make the traffic signs more distinguishable from the other objects. There are international standards regulating the traffic signs. The discriminations of the traffic signs are shown in Table 2-1.

Table 2-1 Features of traffic signs

Sign Type	Possible (Border) Colors	Sign Shape
Restricting & Warning	Red, Blue, Black	Triangle, Rectangle, Octagon, Circle
Information	Blue, Red	Rectangle
Highway Information	Green	Rectangle

2.2 RECOGNITION OF TRAFFIC SIGNS

Recognition of traffic signs consists of three main steps. First, the image is color segmented. Second, in order to localize the sign, shape is extracted. In the last step, the traffic sign is recognized with processing the local region. In the literature,

various techniques are proposed for all these three steps. In the following sections, related previous works are given in each of the steps.

2.2.1 COLOR SEGMENTATION

Pacheco et al. [9] use HSI color space for segmentation of traffic signs because of the robustness of HSI color space for the luminosity changes. They convert the digital RGB data to HSI using their own RGB to HSI converter module. This module has three static memories storing HSI information and works as LUTs. The address inputs of the LUTs are the RGB values of the image, and each color component has 5 bit resolution. The outputs of the LUTs are the 5 bit Hue, Saturation and Intensity components of the image. Finally, Red, Green, Blue and Yellow pixels are segmented according to the Hue and Saturation values. Intensity component is not used in the segmentation process.

Zhu and Liu [10] apply a color standardization approach for the segmentation. 24 bit, 16,777,216 kinds of color are mapped to the five colors which are red, yellow, blue, white and black. This mapping is implemented using the Support Vector Machines (SVM) for its good generalization and over fitting avoidance performance. SVM is a binary classifier and the structure is shown in Figure 2-1. Using SVM, only binary classification can be done. Therefore, it is extended to multi-category SVM (M-SVM) for the segmentation into five colors. The structure of the M-SVM can be seen in Figure 2-2.

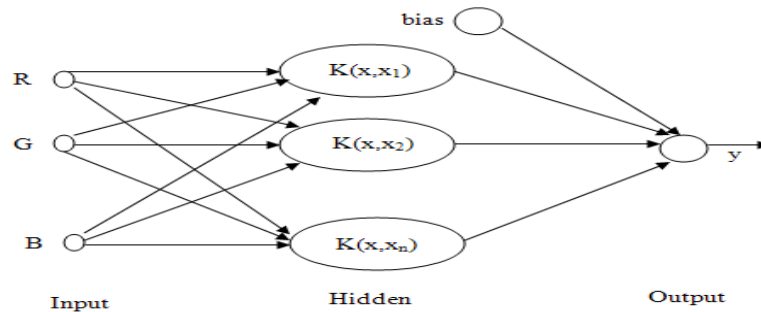


Figure 2-1 Structure of Single SVM

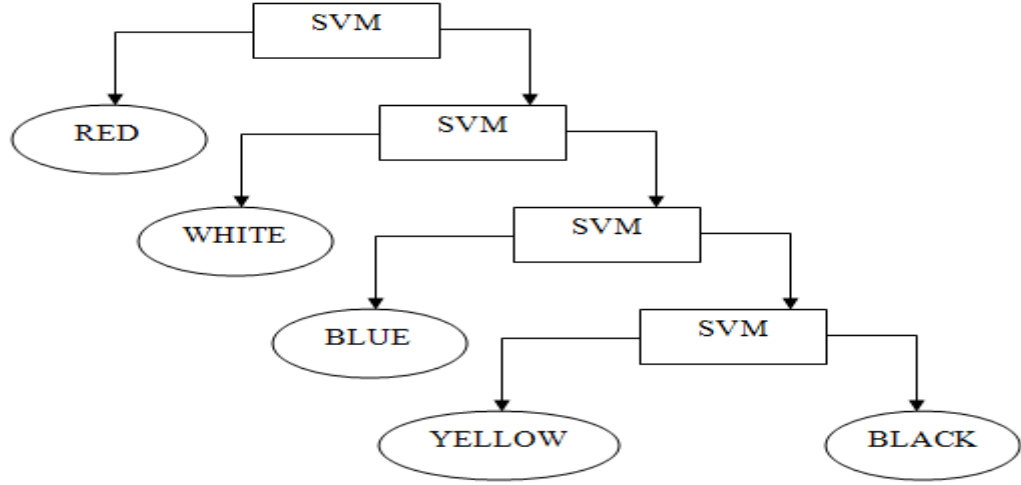


Figure 2-2 M-SVMs structure

Fleyeh [11] uses an improved version of HLS color space which was introduced by Hanburry and Serra [12]. It is later called as Improved HLS (IHLS) which avoids the inconveniences of the other color spaces designed for computer graphics rather than image processing [12]. The formulas for conversion from RGB to IHLS are the (2.1), (2.2), (2.3), (2.4).

For the Hue component,

$$H = \begin{cases} \theta & B \leq G \\ \theta - 360 & B > G \end{cases} \quad (2.1)$$

Where:

$$\theta = \cos^{-1} \left(\frac{\left[R - \frac{G}{2} - \frac{B}{2} \right]}{\sqrt{R^2 + G^2 + B^2 - RG - RB - GB}} \right) \quad (2.2)$$

For the Saturation Component,

$$S = \max(R, G, B) - \min(R, G, B) \quad (2.3)$$

For the Luminance Component,

$$L = 0.212R + 0.715G + 0.072B \quad (2.4)$$

Three methods are introduced for the color segmentation process after converting the RGB color space to the IHLS color space.

In the first method, the reference color and the unknown color are represented by the vectors and the Euclidian distance of these two colors are calculated using the (2.5),

$$d = \{(S_2 \cos H_2 - S_1 \cos H_1)^2 + (S_2 \sin H_2 - S_1 \sin H_1)^2\}^{1/2} \quad (2.5)$$

If the Euclidian distance is less than a threshold which is calculated using the mean of the luminance image for a pixel, then that pixel considered as the object pixel, otherwise it is considered as background. This method allows controlling the relation between reference pixel and unknown pixel using the luminance image.

The second method is based on the region growing. In the first step, a range of hue angles are specified as the probable candidates. After that, saturation image is divided into 16x16 pixels sub-regions, and if there is enough number of probable candidates in this sub-region, a seed is set at the center of it. Using these seeds, another binary image is generated which shows the candidate objects of road signs.

The last method is using transfer functions for the segmentation for each color component. The Saturation transfer function is calculated by (2.6);

$$S_{out} = \begin{cases} 0 & 0 \leq S_{in} \leq S_{min} \\ S_{in} & S_{min} < S_{in} \leq S_{max} \\ 255 & S_{max} < S_{in} \leq 255 \end{cases} \quad (2.6)$$

Similarly, the Hue transfer function is calculated by (2.7);

$$H_{out} = \begin{cases} 255 & H_{min} \leq H_{in} \leq H_{max} \\ 0 & otherwise \end{cases} \quad (2.7)$$

Finally, a logical AND operation is applied to generate the segmented image. In the method, S_{\min} and S_{\max} are chosen as 51 and 170 respectively and the Hue parameters for each color component are given in Table 2-2. For the red color, Hue Min is greater than the Hue Max, this means that the pixels are considered as red, if the Hue is between 245 and 255, also 0 and 10.

Table 2-2 Color enhancement values (normalized to 255)

	Hue Min.	Hue Max.
Red Color	245	10
Blue Color	160	180
Green Color	75	95

Kantawong [13] introduces that although RGB color space is very sensitive to the lighting changes, it is computationally efficient as compared to the HSI color space which has non linear formulas and requires special hardware with high computational cost. Color thresholding is applied using (2.8), (2.9), (2.10);

$$p(x, y) = 255 - \frac{\sum(R, G, B)}{3} \quad (2.8)$$

$$p(x, y)_{avg} = \frac{\{10 \cdot p(x, y) + \sum_1^8 g(x, y)\}}{9} \quad (2.9)$$

$$p(x, y) = \begin{cases} black & \text{if } p(x, y)_{avg} \geq 3 \cdot P(x, y)_{avg} \\ white & \text{if } p(x, y)_{avg} < 3 \cdot P(x, y)_{avg} \end{cases} \quad (2.10)$$

Where R, G, B are red, green and blue color components of the selected pixel, $g(x, y)$ is the 8-neighboring pixels of the selected pixel, $P(x, y)_{avg}$ is the average color value of all pixels in the image.

Varun et al. [14] compares the sum of the green and blue pixel components with the 1.5 times the red pixel component for the color segmentation. If pixel has relatively higher red component, it is determined as the feature pixels.

Andrey et al. [15] also works in the RGB color space; he uses two criteria for segmentation. He states that first criterion shows good result in good lighting conditions. In this criterion, a pixel belongs to red sign if it satisfies (2.11),

$$R_{i,j} > 50 \text{ and } R_{i,j} - B_{i,j} > 15 \text{ and } R_{i,j} - G_{i,j} > 15 \quad (2.11)$$

Second criterion is more robust in bad lighting conditions. In this method, a pixel is red if it satisfies (2.16),

$$k = \frac{255}{\max(R_{i,j}, G_{i,j}, B_{i,j})} \quad (2.12)$$

$$R'_{i,j} = R_{i,j} \cdot k \quad (2.13)$$

$$G'_{i,j} = G_{i,j} \cdot k \quad (2.14)$$

$$B'_{i,j} = B_{i,j} \cdot k \quad (2.15)$$

$$R'_{i,j} - G'_{i,j} > 10 \text{ and } R'_{i,j} - B'_{i,j} > 10 \quad (2.16)$$

2.2.2 SHAPE EXTRACTION

The used shapes for traffic signs are given in Table 2-1 Features of traffic signsTable 2-1. Shape Extraction is the most important part in the traffic sign recognition system because ROI is determined from the extracted shape. In other words, traffic signs cannot be recognized satisfactorily unless shape is extracted accurately.

Generally, shape extraction is applied to the feature image after the color segmentation process. There are several algorithms developed for the shape extraction.

Miura et al. [16] states that if an edge is a part of a circle, the center of the circle and the edge point create a line and this line has the same direction with the gradient of the edge. For each edge, a line is created using this statement and voted in the search area. If there exists a peak in the area, it is considered as the circle. For the extraction of rectangles, they detect the four line boundaries using the x and y histograms of the edge image. Detected peaks in the histograms are the boundary lines of the rectangle.

Arlicot et al. [17] uses the parametric equation of the ellipses to detect the circle in the image. The reason of using parametric equation of ellipse is the perspective deformation of circular signs. The used equation of ellipse is given in (2.17);

$$ax^2 + 2bxy + cy^2 = 1 \quad (2.17)$$

In (2.17), there are only three unknowns. Using the three edge points, the model of the ellipse (i.e. a, b, c parameters) is found. Finally, it is searched how many edge points fit this model. If the number of fitted points is large, the model is accepted and the location of the circle is found on the image.

Liu et al. [18] explores a circle pattern in the image to extract the circles in the image. A prepared circle standard pattern is used to perform pattern matching.

Hatzidimos [19] proposes a method for the shape extraction based on Hough Transform. Using Hough Transform, the lines are detected with the angle data. Triangular signs are equilateral and each angle is accepted within a range rather than a single value due to the perspective distortion. If there are three lines with the

angles $[-10, 10]$, $[50, 70]$ or $[-70, -50]$ degrees, then the shape is accepted as triangle. Using the line equations, the coordinates of the center of gravity and the apexes are calculated. For the detection of circles, Randomized Hough Transform is applied. Using three random points from the pixels of interest, an ellipse is created and the number of pixels belongs to this ellipse is calculated. If there is enough number of pixels in the ellipse, the ellipse is accepted.

Damavandi and Mohammadi [20] utilize Hierarchical Hough Transform for circle detection in order to decrease the computational complexity. Hierarchical Hough Transform described as an iterative version of Circular Hough Transform (CHT). In each iteration, the quantization factor decreased and the accuracy increased. In other words, first large quantization factor is used and unreliable center and radii are found. This step decreases the search region and in the next steps more accurate Hough Transform is applied to the decreased search region for good circle detection. They experienced that two or three iterations are enough for good results.

2.2.3 RECOGNITION

After detection of the ROI, the recognition procedure starts. The main goal of the recognition procedure is to identify the specific sign within its class. Since the number of the traffic signs is huge and also there can be distortions and occlusions on the signs, the recognition procedure should be cost effective and reliable.

For the recognition of the traffic signs, many algorithms have been developed. These algorithms can be classified in three main classes. These are template based algorithms, histogram based algorithms, and neural network based algorithms.

Andrey et al. [15] use the template matching to recognize the traffic signs. Template size is 50x50 pixels. For more robust results, matching is processed relative to the centroid coordinates of template and candidate. Centroids are found from the detection phase. Result of the recognition process is the template sign which is best

matched with the candidate. They reported a 91.3% recognition success rate for their experiment.

Hatzidimos [19] uses cross correlation to match the ROI and specific template images. Before cross correlation is carried out, the ROI is scaled and rotated, so both ROI and template have the same coordinate system. After transformation process, for every ROI pixel, the cross correlation coefficient is calculated. The template with the largest cross correlation corresponds to the sign searched.

Escalera et al. [21] use neural networks with multilayer perceptron. Two neural networks are used for recognition one for triangular and another for circular signs. They present the 30x30 input images as the input pattern. The output layer is ten for both shapes. One of the output shows that the sign is not recognized. Therefore, totally nine different signs can be recognized for triangular and circular traffic signs.

Paulo et al. [22] utilize Pictogram Contours to recognize the traffic signs. The proposed recognition algorithm analyses the pictogram outer contours. If pictogram consists of several disconnected regions, a snake is used to create an outer contour. To represent the outer contour a Curvature Scale Space (CSS) is used. CSS is a multi scale representation, which describes the object shapes by analyzing the second derivative zero crossing points of the outer contour. After creating the CSS image of the curve, traffic sign recognition is done by matching CSS image of the candidate with all the database CSS images of the template signs.

Wu et al. [23] use horizontal and vertical axis projections to classify the road signs. The projections are divided into three parts and the peaks of these two projections are used as the features. Therefore, for all the signs with the same shape totally nine different classes can be created. In others words, there may be two or more traffic signs that have the same features. In this case, it is necessary to use a further classification technique to identify the sign. However, in this paper, a further classification technique is not used.

Bahlman et al. [24] classifies the traffic signs using Bayesian generative modeling with unimodal Gaussian Probability densities. Before the modeling, Linear Discriminant Analysis (LDA) feature transform is applied. The database consists of totally 23 signs and the error rate is 6%.

The adaptive approach of Zheng et al. [25] is utilizing a distance weighted k-nearest Neighbor classifier.

In [26], a real TSR system is introduced using a soft core processor in Cyclone 2 FPGA. After the detection step, ROI is resized to 80x80. The correlation is used to compare the candidate and the template images in the database. After comparing task, the highest similarity with the candidate shows the identification result. Although, it is stated that their system is a real time recognition system, the total computation time of all tasks is about fifteen seconds.

Braun et al. [27] implement a TSR system using Leon Processors and Virtex-4 LX100 FPGA. In the recognition step, the system can recognize different speed limitations and the prohibition signs. These signs are classified using template matching based on the SVM. Total computation time is about 0.5 seconds.

In [28], stop sign detection system on FPGA is developed. System uses Histogram of Gradients (HoG) for the stop sign detection. In order to calculate HoG, gradient angles are required. Normally, the gradient angle is calculated using (2.18),

$$g = \tan^{-1}\left(\frac{g_y}{g_x}\right) \quad (2.18)$$

Where g is the pixel's gradient angle; g_x and g_y are pixels' gradients in the horizontal and vertical directions, respectively.

Division and inverse trigonometric functions are not suitable for hardware implementations. Therefore, rather than using (2.18), each angular bin is quantized to 4 bins directly using the g_x and g_y ranges as shown in Figure 2-3.

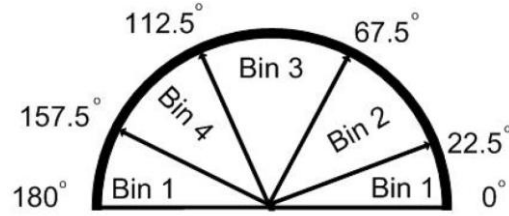


Figure 2-3 Angular Spacing of 4 bin HoG

After calculating HoG, for each angular bin Integral Map (IMap) is constructed and stored in the dual port RAMs. At every pixel location, HoG features are extracted using the IMaps, and each HoG feature is compared to its appropriate threshold in a parallel manner to determine the output [28]. In the last step, the verification of the outputs is done. A candidate is detected as a stop sign if it is detected in at least 6 out of 10 consecutive frames and the location difference between consecutive detected signs are at most 4 pixels from each other. System detection rate is 81.25% with 1 false positive in 7200 frames.

CHAPTER 3

SHAPE EXTRACTION AND RECOGNITION

The input image and shape information of the traffic sign are given to the TSR system. In the shape extraction, exact location of the sign is found. First the image is segmented and dominant color is decided in order to decrease the complexity. For the accurate and robust shape detection, Hough Transform is utilized. Before the Hough Transform, edge detection is applied. LoG Edge Detector is preferred because of its good edge detection capability. In the recognition step, first, binary threshold is applied to decrease the complexity and morphological opening is used to eliminate the unwanted pixels. Finally, recognition is utilized to identify the sign. Steps of the proposed system are given in Figure 3-1. In this chapter; each process will be explained in detail.

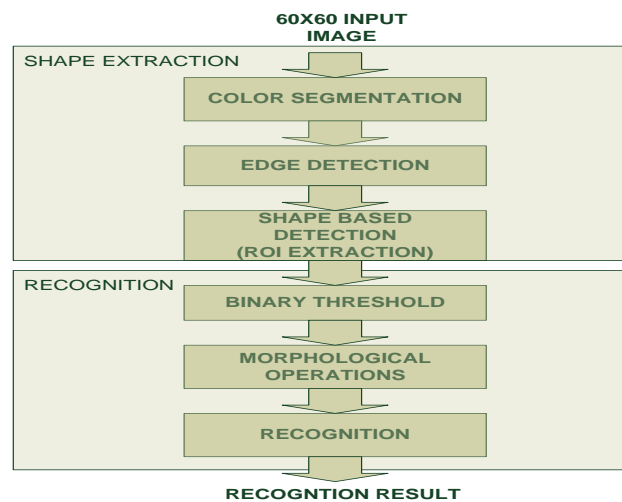


Figure 3-1 The Steps of the Proposed System

3.1 SHAPE EXTRACTION

3.1.1 COLOR SEGMENTATION

The colors of the traffic sign borders are red, blue and green. In this thesis, red and blue traffic signs are studied. The red bordered signs are used as a warning or a restricting signs and blue bordered signs are information signs. Since these colors are very discriminative as compared to background, color segmentation can be applied to these colors. The output image of the segmentation process can be used in edge detection process. The advantages of color segmentation before the edge detection are as follows;

- Using color segmentation eliminates undesired colors, thus the number of edge pixels in the edge detection process decreases. In other words, the complexity decreases since only edge pixels are processed.
- With the help of color segmentation, fault detections decrease in the detection process. Since circular, rectangular or triangular shaped objects with colors other than blue or red are eliminated by the segmentation.
- Color segmentation gives information about the border color and the inner color of the sign. This information is further used in the recognition process.

For the traffic signs considered in this thesis, two color segments are possible, one for red signs and another for blue signs. Since only shape information exists and the color of the sign is not known from the pre-detection process, the first thing that should be done is to determine the color of the borders and inner part of the sign. The procedure is given in (3.1),

$$\begin{aligned} & \sum_{i,j} B_{i,j} > T_b \text{ and } \sum_{i,j} R_{i,j} > T_r, \text{ then sign has red border and blue color} \\ \text{if } & \sum_{i,j} B_{i,j} \leq T_b \text{ and } \sum_{i,j} R_{i,j} > T_r, \text{ then sign has red border} \\ & \sum_{i,j} B_{i,j} > T_b \text{ and } \sum_{i,j} R_{i,j} \leq T_r, \text{ then sign has blue border} \end{aligned} \quad (3.1)$$

Where T_b and T_r are the blue and red threshold parameters, B and R are the blue and red segmented images respectively.

Although HSV color space is invariant to changing lighting conditions, because of its computational complexity RGB color space is used in segmentation. In RGB color space, a pixel belongs to red region if it satisfies (3.2),

$$R_{i,j} > k.B_{i,j} \text{ or } R_{i,j} > k.G_{i,j} \quad (3.2)$$

Similarly, a pixel belongs to blue region if it satisfies (3.3),

$$B_{i,j} > k.R_{i,j} \text{ or } B_{i,j} > k.G_{i,j} \quad (3.3)$$

Where $R_{i,j}, G_{i,j}, B_{i,j}$ is the color components of the pixel of image with coordinates (i, j) , and k is the coefficient parameter.

Figure 3-2 and Figure 3-3 show the original pictures from which red and blue color segments are extracted and red and blue color segmented images for $k=2$.

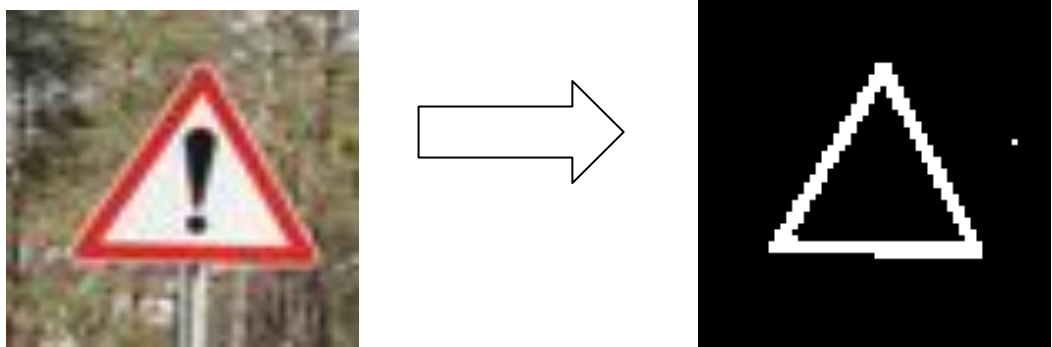


Figure 3-2 Traffic Sign and Red Segmented Image

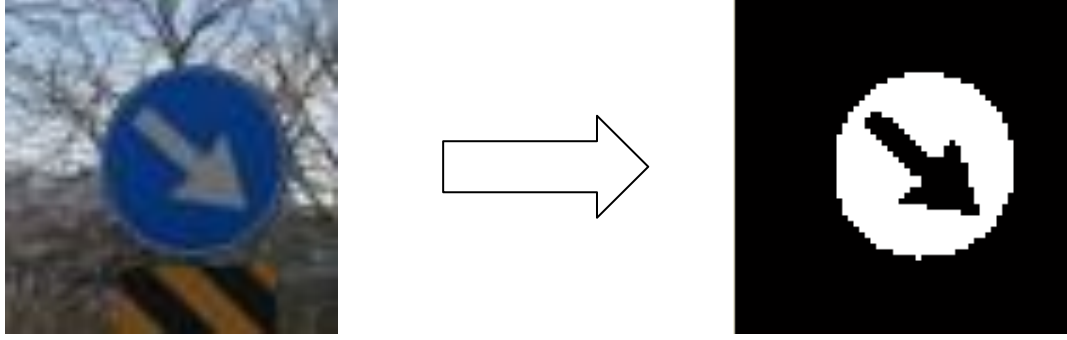


Figure 3-3 Traffic Sign and Blue Segmented Image

3.1.2 EDGE DETECTION

After color segmentation process, edge detection is performed. Edge detection process consists of two steps. First the segmented image is filtered by Laplacian of Gaussian Filter (LoG). LoG is used to highlight regions of rapid intensity changes and reduce the sensitivity to noise [29]. This filter has higher performance than the first derivative operators such as Robert, Prewitt and Sobel operators and its edge detection performance is very close to the Canny Edge Detector which is known as the optimal edge detector [29]. After filtering operation, zero crossing points of the filter output are detected. These points are the edge pixels in the image.

3.1.2.1 LAPLACIAN OF GAUSSIAN FILTERING

The Laplacian $L(x,y)$ of an image is given in (3.4):

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \quad (3.4)$$

Where $I(x,y)$ is the intensity value of pixel (x,y) .

The discrete convolution kernel that approximate the second derivative function x is shown in Figure 3-4.

-1	-1	-1
-1	8	-1
-1	-1	-1

Figure 3-4 Laplacian Kernel

Since the second derivative approximation is very sensitive to noise, a pre-processing is required. To decrease the high frequency components, before applying Laplacian filtering, image is smoothed with Gaussian Filter. Since convolution operation is associative, a single function can be used. In other words, instead of using two separate filters, a pre-calculated kernel is used for Gaussian Smoothing and Laplacian Filtering. This kernel is called Laplacian of Gaussian (LoG) Kernel. The LoG function is given in (3.5) and the plot of the function is shown in Figure 3-5. The kernel is shown in Figure 3-6.

$$LoG(x, y) = -\frac{1}{\pi\sigma^4} \left[1 - \frac{x^2 + y^2}{2\sigma^2} \right] e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (3.5)$$

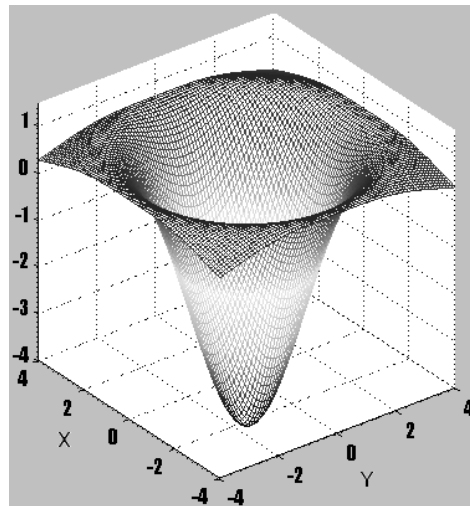


Figure 3-5 Laplacian of Gaussian Function

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 3-6 Laplacian of Gaussian Kernel

3.1.2.2 ZERO CROSSING DETECTION

After image is filtered, to find the edges in the image, zero crossing points in the filtered image are detected. Zero crossing points are the edge pixels in the image. A pixel is zero crossing pixel if filtered output pixel is changing its sign in the 4 neighborhood pixels. These zero crossing pixels are the edges in the image. It is important to note that the filtered output is zero for smooth regions in the image. There is no sign change in these regions. Thus, they are directly discarded in the zero crossing detection.

One of the main advantages of using LoG edge detector is that unlike first derivative operators such as Sobel, there is no threshold parameter adjustment in the edge detection. Moreover, using color segmentation before the edge detection decreases the number of edge pixels and the edge image contains less number of unrelated edges. An example traffic sign and the detected edges of it using LoG edge detector can be seen in Figure 3-7. The effect of the applying red color segmentation before the edge detection process can be seen in Figure 3-8.

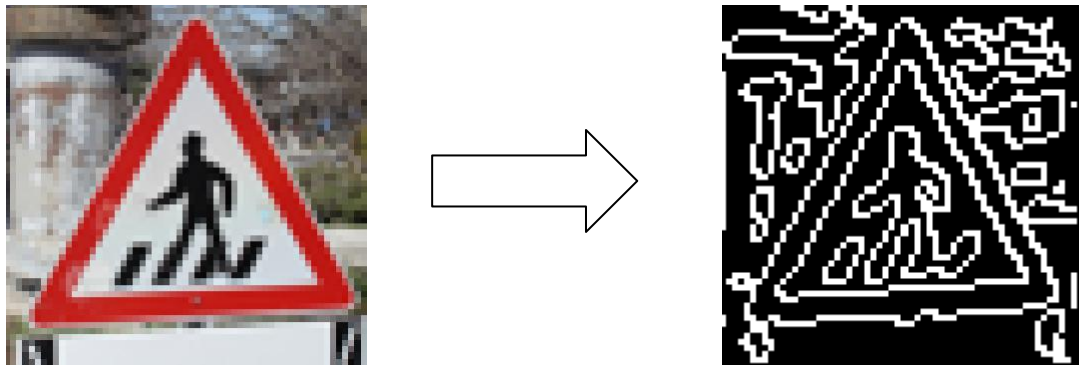


Figure 3-7 Edge Image without Color Segmentation

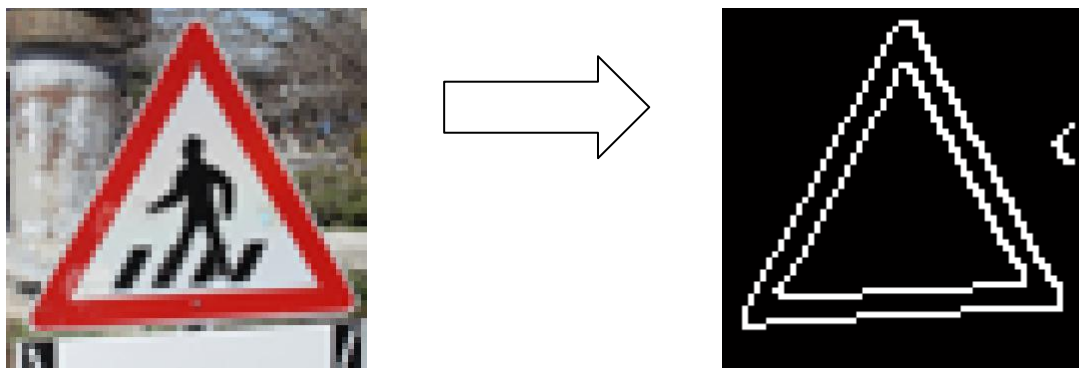


Figure 3-8 Edge Image with Color Segmentation

3.1.3 SHAPE BASED DETECTION

Triangle, circle and rectangle are the most widely used shapes in traffic signs. Shape information is known from the detection stage. In order to increase the recognition rate of the system, detection process should be as accurate as possible. Accurate means that ROI of the image must be well isolated from the background, no background pixel should be in the ROI and the center of the traffic sign should be detected as close as possible. For the accurate detection of the traffic signs, Hough Transform is used. The main advantage of the Hough Transform is

robustness and high positional accuracy in detecting lines and circles. However, the main disadvantage of the Hough transform is the complexity. There is a tradeoff between positional accuracy and the complexity of the algorithm. Increasing the accuracy would increase the complexity. To overcome this, some inherent features of traffic signs have been used to decrease the complexity without affecting the positional accuracy performance. Since the angle of the lines in triangle and rectangle traffic signs are known, there is no need to search the lines for the other angles. However, there are some problems which also increase the complexity. Geometric distortion of the traffic signs because of viewing angle is a good example for this situation. Especially in circular traffic signs, CHT should be extended to the ellipses because of the geometric distortion of circles. In the following sections, each shape detection algorithm is explained in detail.

3.1.3.1 TRIANGULAR SIGN DETECTION

Hough Transform gives us a method to detect lines in an image. First, let's consider the slope-intercept equation of a line;

$$y = a.x + b \quad (3.6)$$

In (3.6), “a” is the slope of the line and “b” is a constant intercept parameter. To detect a line in an image, enough number of points (x,y) in the image should lie on the same line. In other words, for a set of (x,y) pair, there is a corresponding (a,b) pair, which satisfies (3.6). Similarly, each different line in (x,y) domain has different (a,b) pair, and all these (a,b) pairs create a line in (a,b) domain [31].

However, there is a problem in (a,b) domain while detecting vertical lines that give rise to the unbounded values to the a and b. Since for vertical lines, slope goes to infinity. Because of these computational reasons, it is better to use Polar Coordinates instead of Cartesian Coordinates. In other words, a line can be represented with the distance (r) between the line and origin, and an angle (θ) of the

vector from origin to the closest point to the line. The (r, θ) coordinate system is shown in Figure 3-9.

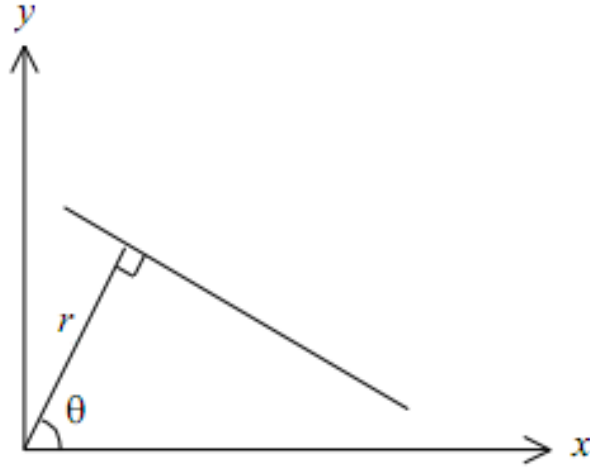


Figure 3-9 Representation of a line with (r, θ) parameters

The corresponding line equation is of Figure 3-9 is given in (3.7):

$$x \cdot \cos \theta + y \cdot \sin \theta = r \quad (3.7)$$

In (3.7), if one takes (x,y) as constants, and (r,θ) as the variables, then a point in (x,y) domain has a curve in (r,θ) domain because a few lines are passing from a single (x,y) point and each line has different (r,θ) values. In Figure 3-10, point (x,y) maps to the solid curve in (r,θ) domain, and point (u,v) maps to the dashed curve in (r,θ) domain. If there is a line which is passing through these two points, then there should be an intersection point in the (r,θ) domain. In Figure 3-10, this intersection point is (r^*,θ^*) . This single point in (r^*,θ^*) domain maps to a line in (x,y) domain with the line equation given in (3.8) :

$$x \cdot \cos \theta^* + y \cdot \sin \theta^* = r^* \quad (3.8)$$

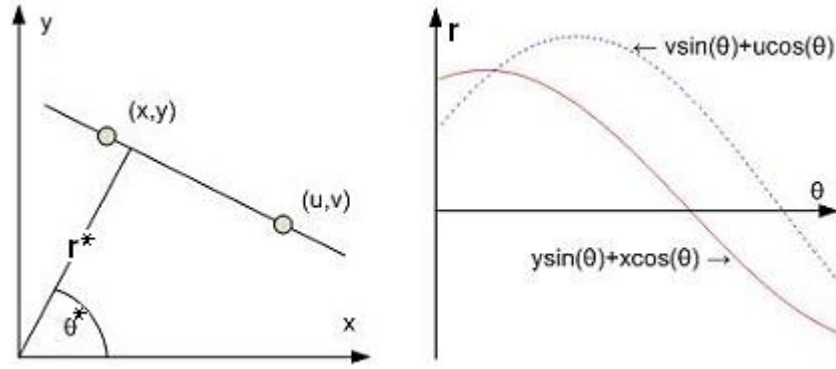


Figure 3-10 Mapping from Cartesian Coordinates to Polar Coordinates

Now, in order to find the lines in the image, for every point in the image we find the corresponding (r, θ) curve and then count the intersection points. Actually, this is a voting procedure. Each intersection point is a vote for the corresponding (r, θ) value, and the one with takes the highest vote in voting is the equation of the line in the image [30]. In other words, the (r, θ) point which has the highest number of intersection points is the equation of line in the image. When there is more than one line in the image, the procedure is the same. If one knows the number of lines in the image, the corresponding (r, θ) parameters of these lines are the points which take the highest votes in the voting procedure.

Since triangles consist of three lines, in order to detect triangles, three lines should be detected. To decrease the complexity, only 0, 60, 120 degree and ± 6 degree around of these angles are searched for the lines. In each degree, two lines are searched, one for the inner border and the other for the outer border. After detecting the two lines in a searched angle, the inner one is selected as the detected line and the outer one is discarded.

An example edge image of a traffic sign is shown in Figure 3-11 and the result is shown in Figure 3-12, the blue crosses show the edge image. The corresponding detected six lines (two lines for each angle) are shown as red circles in Figure 3-12.

After detecting these six lines, one can easily extract the inner part of the image. The area between inner lines is the inner part and called ROI.

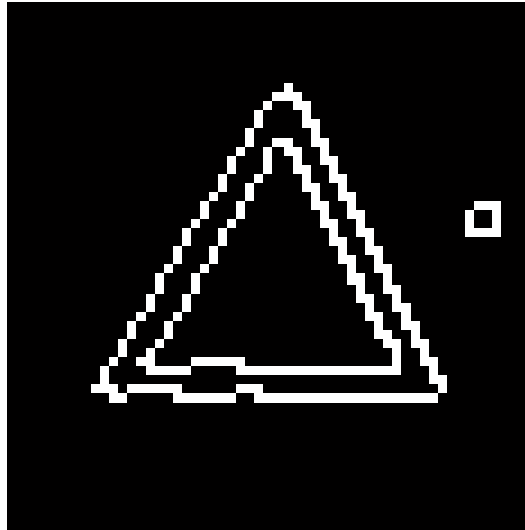


Figure 3-11 Edge Image of a Triangular Traffic Sign

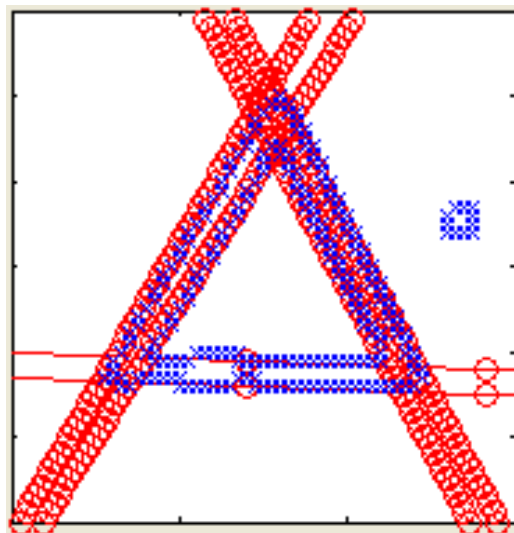


Figure 3-12 Detected Lines after Applying Hough Transform

3.1.3.2 CIRCULAR SIGN DETECTION

In order to detect circles, CHT is used. In CHT, after determining edge image for each possible radius, an accumulator space representing center votes for each pixel on the image is constructed. Therefore, radius and the center coordinates of the circle are the elements of the parameter space. In other words, the parameter space is three dimensional (x_c, y_c, r) . The parametric equation of a circle of radius r with center (x_c, y_c) is given in(3.9);

$$(x - x_c)^2 + (y - y_c)^2 = r^2 \quad (3.9)$$

However, due to the perspective distortion, circular traffic signs may appear as elliptical objects. Using CHT in these elliptical objects cannot give an accurate detection of ROI and center of the traffic sign. In Figure 3-13, a circular traffic sign is seen as elliptical. After applying CHT, the resulting detected circle and the center of the sign is shown in red color in the sign.



Figure 3-13 Detected Circle After Applying CHT

CHT is not capable of detecting ellipses. Therefore, it should be extended in order to detect ellipses. The parametric equation of an ellipse is given in (3.10):

$$(x - x_c)^2 + k.(y - y_c)^2 = r^2 \quad (3.10)$$

The only difference between parametric equation of circle and ellipse is the “k” parameter. When k is equal to one, ellipse becomes a circle. If k is smaller than one, then the y diameter of the ellipse is greater than the x diameter, if k is larger than one, then x diameter of the ellipse is greater than the y diameter. The effect of the “k” parameter can be seen in Figure 3-14.

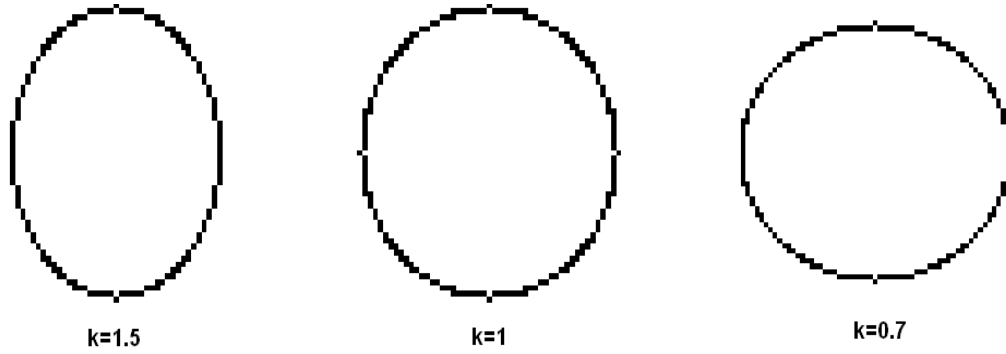


Figure 3-14 Effect of the "k" parameter

Since the parametric equation of the ellipse is known, extending the CHT to detect ellipse is straightforward procedure. The main disadvantage of the ellipse detection as compared to the circle detection is the number of parameters. In circle detection, parameter space is three dimensional (x_c, y_c, r) where in ellipse detection, parameter space is 4 dimensional (x_c, y_c, k, r) . In Hough Transform, increasing the number of parameters directly increases the complexity. For the line detection, $O(n)=2$, and similarly for circle detection $O(n)=3$, and for the ellipse detection $O(n)=4$.

In order to decrease the process time in ellipse detection some assumptions are made. These are;

- The radius of the traffic sign is not smaller than 10 pixels in the 60x60 pixel image patch.
- The ratio of the long radius to short radius of the ellipse is not greater than two.

These two assumptions are reasonable. Since when a car is getting closer to a traffic sign, a reliable detection can be done if the traffic sign is as close as possible and if traffic sign is detected when its radius is smaller than 10 pixels, probably it will be detected again when its radius is greater than 10 pixels. In addition, for the second assumption, in the traffic it is nearly impossible to see a circle as an ellipse with long radius to short radius ratio greater than two due to the perspective distortion.

After applying ellipse detection to the image in Figure 3-13, the resulting detected ROI and center of the sign is shown in Figure 3-15. With the help of the “k” parameter in ellipse detection, the ROI and center of the sign is detected more accurate as compared the CHT.



Figure 3-15 Detected Ellipse after Applying Ellipse Detection

3.1.3.3 RECTANGULAR SIGN DETECTION

In rectangular sign detection, totally four lines should be detected. Since the angles of the lines are 0 and 90 degree and around, line detection with Hough Transform is simplified as a peak histogram finding problem using x and y projections. In the x and y projections, two peaks should be observed because of the edges in the border of the rectangular traffic sign. An example can be seen in Figure 3-16. In this figure, in both projections of image, there are two peaks which are the borders of the traffic sign. After finding the borders, center coordinates of the rectangle can be detected taking the average of the border coordinates. The detected border lines can be seen in Figure 3-17.

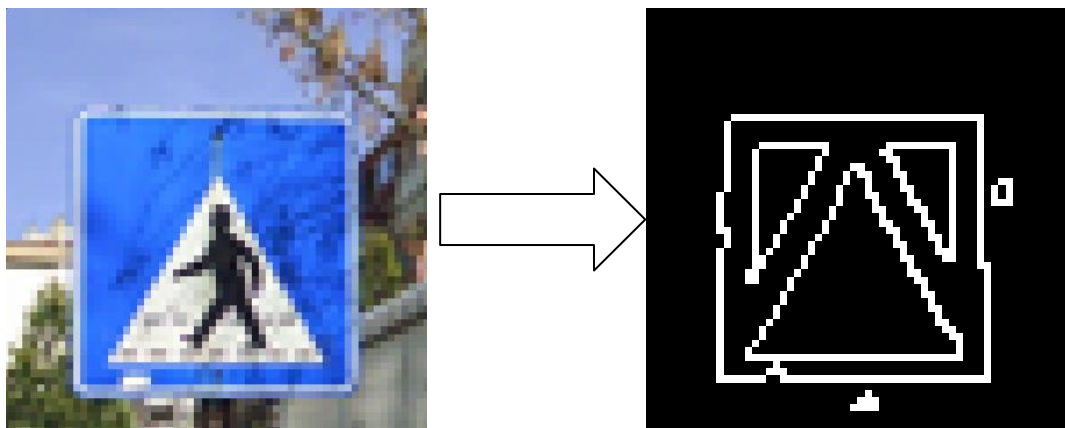


Figure 3-16 Rectangular Sign and Edge Image

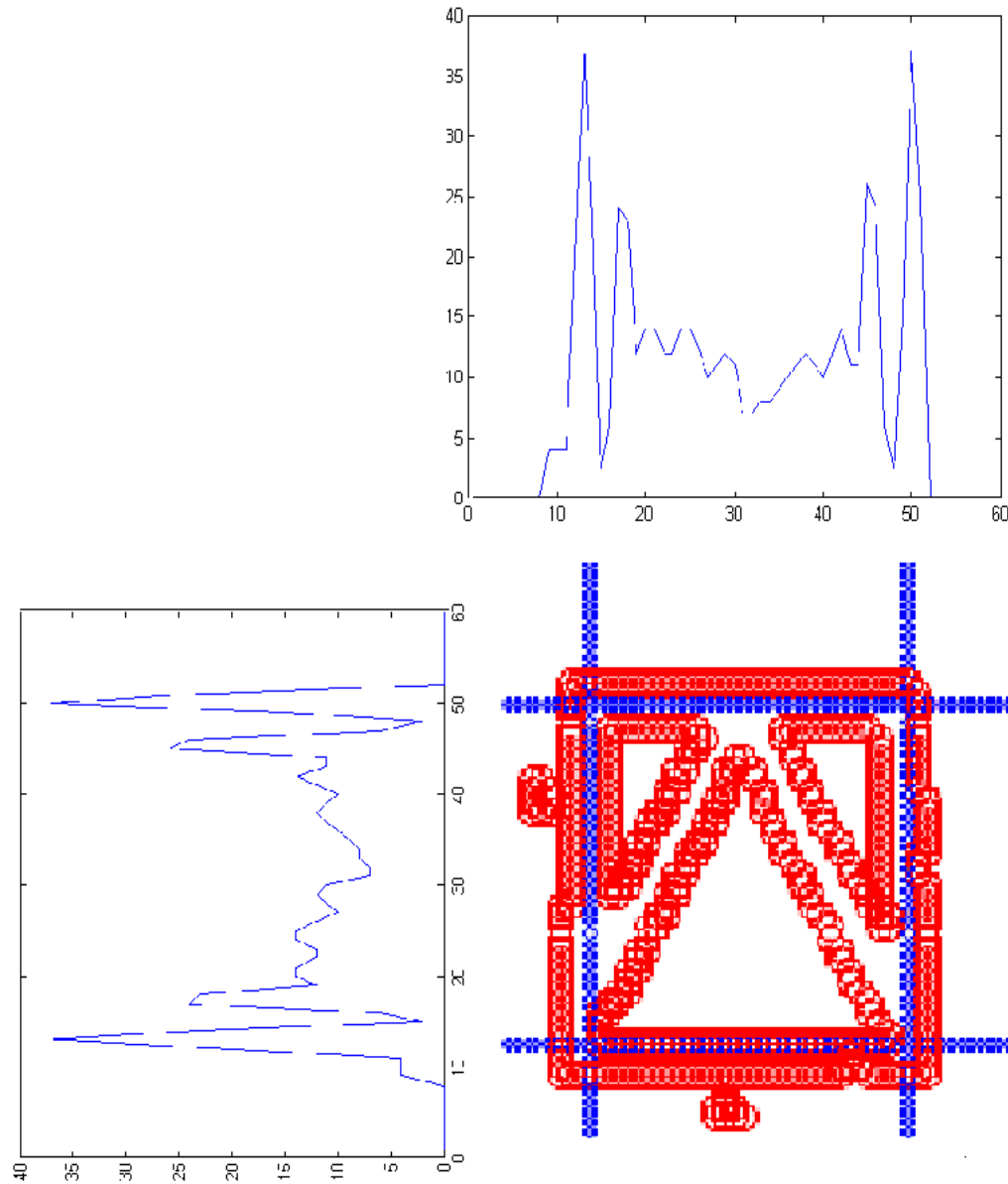


Figure 3-17 x and y Projections of the Edge Image

3.2 RECOGNITION

Recognition consists of three steps. In this first step, a binary image is generated using ROI of the image, since one bit is enough to extract the foreground from the background. In the second step, morphological operations are applied to the binary image in order to remove the unwanted pixels. In the last step, sign is recognized.

For the recognition of signs, IPP matching is used for its lower complexity. In IPP matching, each shape is divided into smaller regions and the informative pixels of each region are calculated. These values are normalized dividing by the total number of informative pixels in ROI. These normalized values are called IPPs. After that, these IPPs are matched with the templates' IPPs in the database. Best matching template sign is the result of the recognition process. Best matching means the SAD between the template IPPs and the image IPPs is minimum. In addition to the IPPs matching, some additional information known from the segmentation and detection processes are also used to increase the recognition rate. The number of divided regions depends on the shape of the traffic sign. Using more regions makes the recognition more accurate but also more complexes. Using IPP matching has advantages in implementation. These are;

- Since instead of the number of pixels, the percentages are used, the method becomes scale invariant. The percentage of the informative pixels is not affected from scaling.
- For each template image, only few numbers (percentages of each region) are used to model the templates. There is no need to use the whole image. Therefore, there is no memory requirement to store the templates as compared to template matching.
- There is no initialization or calibration sequence as compared to the neural network based methods. In addition, inserting a new template only means inserting few numbers to the database. There is no need to train the system.
- Calculating the percentages is less complex as compared the other methods such as template matching, or neural network.

3.2.1 BINARY THRESHOLDING

In the detection process, three lines of the triangle are determined. ROI is the inner part of these three lines which is the informative part of the image. It consists of only two different colors. One is the informative color of ROI and the other is the background color. Therefore, using only one bit is enough to identify the traffic

sign. A binary threshold function which is the average pixel value of ROI is used for the binarization process. In the ROI, if green component pixel value is greater than the average, then it is “1”, otherwise it is “0”. The aim of using the green component in the binarization process is that green component has the best extraction of foreground from background for traffic signs. Since for the blue bordered signs, the foreground is white and the background is blue, so the difference between background and foreground is high in the green component image. Similarly, for the red bordered signs, the difference between red border and white background in the green component of the image is high which eliminates the red borders directly in the binarization process. Thus, green component is used for the binarization. The result of the binarization process applied to the image is given Figure 3-18.

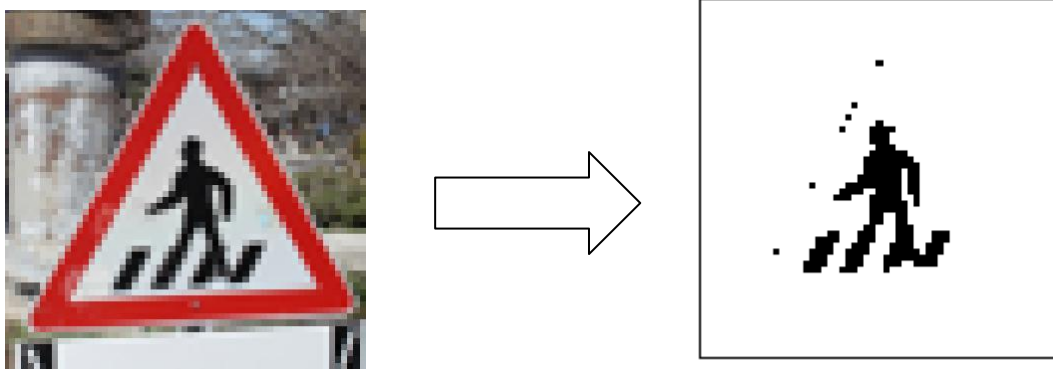


Figure 3-18 Output of Binarization Process

3.2.2 MORPHOLOGICAL OPERATIONS

There can be some pixels which are in the detected triangle but should not be in ROI for accurate results. Since the detected lines are continuous and pixel coordinates are discrete values, some informative pixels at the borders should be discarded. To remove these pixels a morphological opening is applied, first binary image is dilated then it is eroded with a 2x2 structuring element. Structuring

elements used in dilation and erosion and the binary image before opening and after opening are shown in Figure 3-19 and Figure 3-20, respectively.

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

Figure 3-19 Structuring Elements used in Dilation and Erosion

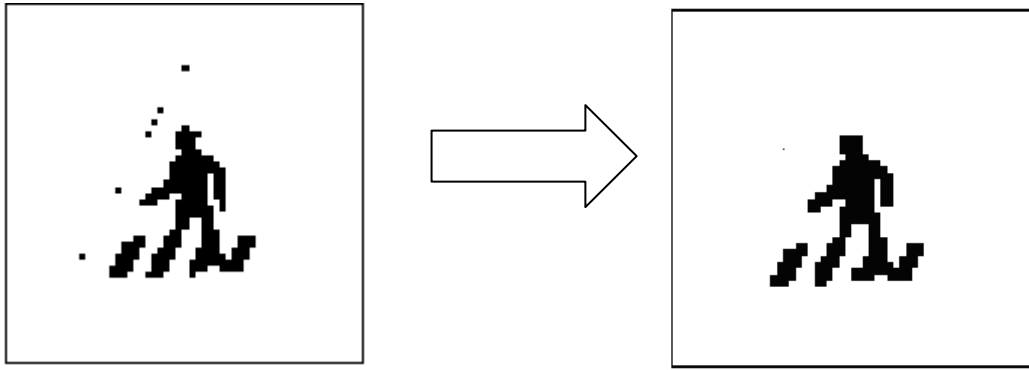


Figure 3-20 Binary Image before and after Opening

3.2.3 RECOGNITION USING INFORMATIVE PIXEL PERCENTAGE FINDING AND MATCHING

Recognition is simply a comparison of features of template and candidate. The best matched template with the candidate is the recognition result. The key question is which features of the image should be used for matching. The feature can be whole image as in the template matching based algorithms or histogram of the image as in histogram matching based algorithms. In this thesis, the image is divided into regions and features are taken as the IPP of each region. Recognition is done using these percentages. The number of divided regions depends on the sign shape and it is determined experimentally using the number of template sign in the database for that shape. Higher number of signs in database for a class of shape results in more divided regions in the shape. There are 15 triangular, 13 circular, 4 rectangular signs

in the database. The numbers of divided regions are 6 for triangular, 5 for circular and 4 for rectangular signs. The method is explained for each shape in the next sections.

3.2.3.1 TRIANGULAR SIGN RECOGNITION

After a binary ROI image is found, triangle ROI is divided into six regions as shown in Figure-3.20. In order to divide the triangle, corners and midpoints of each border should be determined. Corners are detected using the intersection of the detected lines and using the corner coordinates, the midpoint of each line is determined. After corners and midpoints are determined, from each corner to the corresponding midpoint a virtual line is drawn. Using these three virtual lines, the triangle is divided into six regions as shown in Figure 3-21.

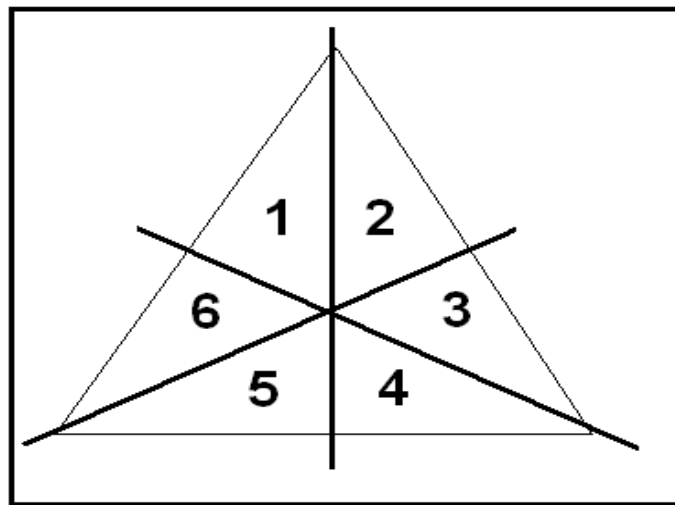


Figure 3-21 Divided Regions of Triangular Sign

Calculating the percentage of informative pixels in each region gives information about the traffic sign. A traffic sign can be modeled using these six percentages. After finding these six percentages, these values are compared to the each template signs' percentages in an absolute error sense. The template with the minimum SAD

with the candidate sign is the recognition result. As stated in Section 3.2, IPPs have advantages as compared with the other methods. Using IPPs in triangular sign recognition has an additional advantage. Since the reference points are corners and midpoints, which are rotation invariant features of the triangular signs, IPPs are not affected by rotation in triangular sign recognition.

3.2.3.2 CIRCULAR SIGN RECOGNITION

In the detection process, the ellipse is determined with the centers and borders. In addition to that, from the color segmentation process, the colors on the sign are known. For the circular traffic sign, the color options are;

- *the border is red, and inner sign is black, background is white.
- *the border is red, and inner sign is blue.
- *the border is blue, and the inner sign is white, background is also blue.

In Figure 3-22, example circular signs can be seen which can be classified into three groups: red border no blue color, blue border, red border blue color. These classes are used for a pre-classification, before the recognition process, in order to increase the recognition rate. Since the class of the candidate sign is known from the segmentation process, in the recognition process, candidate sign is compared with templates only in that class.







RED BORDER NO BLUE COLOR	BLUE BORDER	RED BORDER BLUE COLOR
		
		

Figure 3-22 Classification of Circular Signs

The procedure for the circular sign recognition is very similar to the triangular sign recognition. After binarization, binary image is constructed. Since there is no corner in circular traffic sign, the division is done using the borders of the traffic sign. Circular traffic sign is divided 3x3 regions for the recognition process. However, corner regions are not used, since for the circular signs, the patterns in the corner regions are the same. Therefore, there is no need to use these regions. Division of traffic sign is shown in Figure 3-23. The utilized regions are numbered.

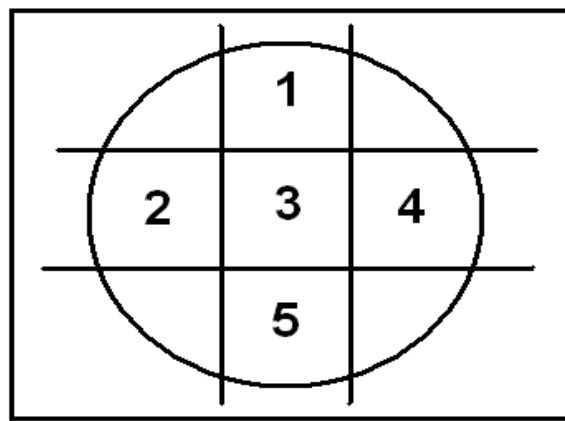


Figure 3-23 Divided Regions of Circular Sign

After calculating the IPPs of candidate sign, the percentages are compared to the IPPs of template signs in the class which the candidate sign belongs to. The recognition result is the template sign which has the minimum SAD with the candidate sign.

3.2.3.3 RECTANGULAR SIGN RECOGNITION

Similar to other signs, the rectangular sign is divided four equal regions, and the IPPs of these four regions are used for matching process. These divided regions are shown in Figure 3-24. After the IPPs of these four regions are found, the

percentages are compared to percentages of the templates. The result is the template which has the minimum SAD with the candidate sign.

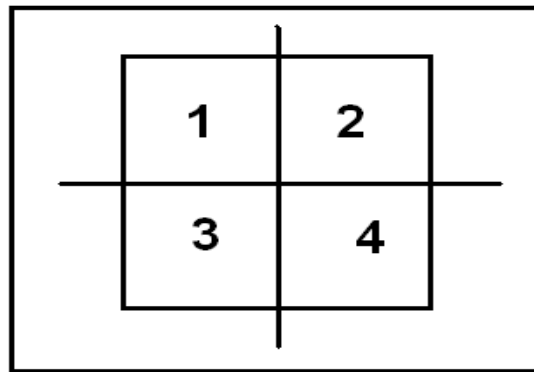


Figure 3-24 Divided Regions of Rectangular Sign

CHAPTER 4

RECOGNITION OF TRAFFIC SIGNS USING FPGA HARDWARE

The traffic sign recognition algorithm explained in Chapter 3 is implemented in Xilinx ML507 FPGA board. Before the FPGA implementation, the algorithm is built and verified in MATLAB.

The parts, which are implemented in FPGA, are synthesized and loaded to FPGA with Xilinx XST. The parts, which are implemented in processor, are generated and loaded with Xilinx EDK Platform Studio.

For the design and simulation of the system, HDL Designer and ModelSim Simulation Software are used. In addition, for the real time debugging, Xilinx Chipscope is used.

4.1 HARDWARE ARCHITECTURE

TSR algorithm is developed on the FPGA platform. Xilinx ML507 Demo Board is selected as the development environment. In order to understand the TSR system better, before giving the FPGA architecture, hardware architecture is introduced in this section.

4.1.1 XILINX ML507 DEMO BOARD

Xilinx ML507 FPGA demo board includes Virtex-5 series FPGA, Flashes, RAMs, PROMs. The board is shown in Figure 4-1. Detailed information about Xilinx ML507 FPGA Board is given at the Appendix-A.

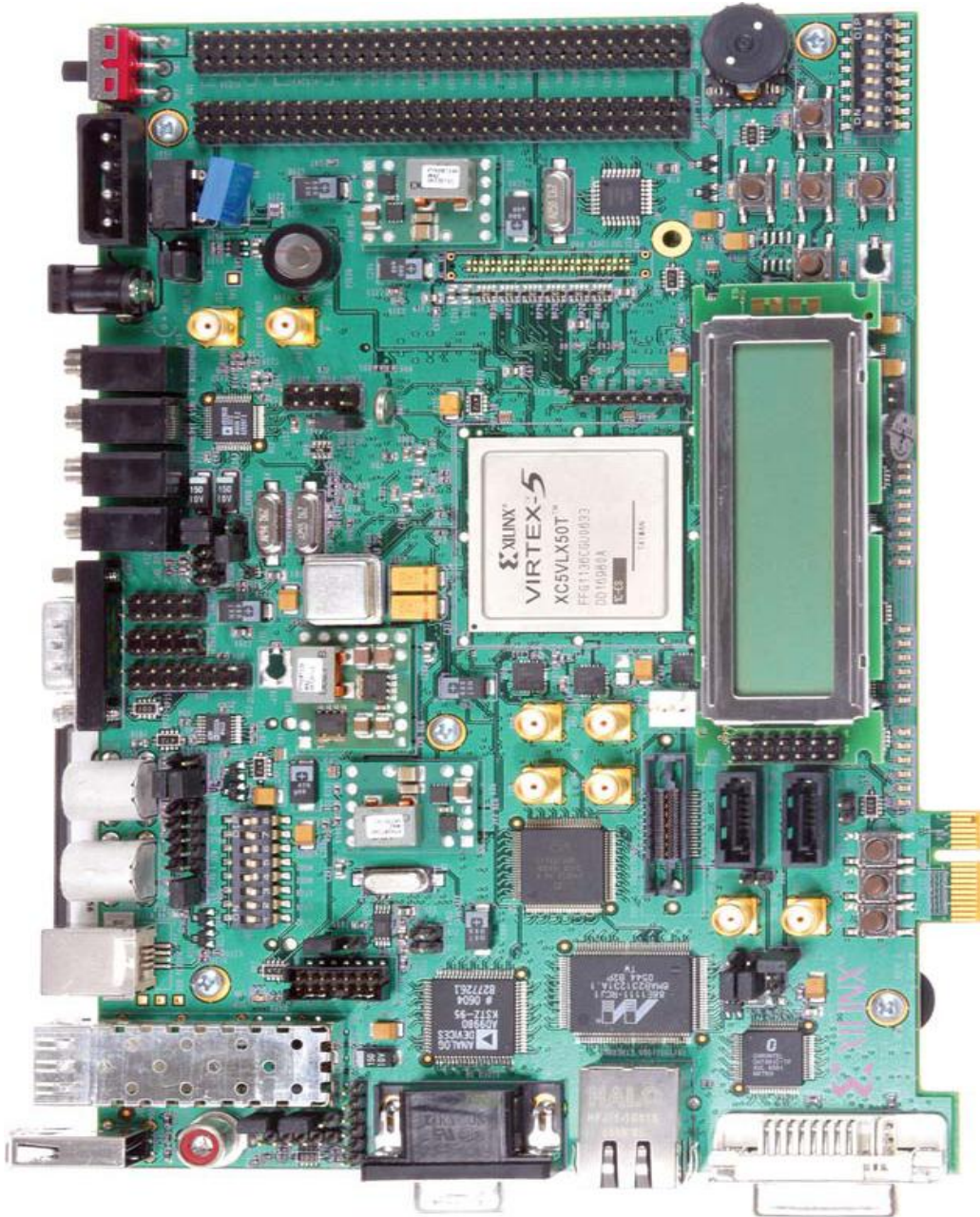


Figure 4-1 ML507 FPGA Development Board

4.1.2 USED COMPONENTS AND FUNCTIONS

In the TSR system, five components are used: FPGA, PROM, Linear FLASH, DDR2SDRAM and Character LCD. These components and functions are explained in detail in this section.

4.1.2.1 VIRTEX5-FX70T FPGA

On the ML507 demo board, Virtex-5 FX70T is used. These FX series FPGA's have 500 MHz maximum operating frequency and include Hard PPC440 Processor Core having 450 MHz maximum operating frequency for the Software Applications. In the TSR system, both FGPA and PPC440 are used. The image patch is received by FPGA from Serial Port, and then image processing algorithms are applied to the image in order to recognize the traffic sign. Some image processing algorithms are handled by FPGA and the others are handled by PPC440. The applications and architecture of the system are given in detail in the following sections.

4.1.2.2 XILINX PLATFORM FLASH PROM

FPGAs can be divided into two main groups in terms of configuration memory options. Some FPGA companies, like Xilinx and Altera, have RAM based configuration memories whereas some companies like Actel have flash-based configuration memories. RAM based FPGAs have volatile configuration memories, so at the power down the configuration memory is cleared. Therefore, at each power up, FPGA should be configured from an external non-volatile memory. For the flash-based FPGAs, the configuration memory is not cleared after power down, so there is no need to use an external memory for configuration, which is the main advantage of flash-based FPGAs. Since there is no configuration file loading exists in flash-based FPGAs, the initialization time of these FPGAs is in the order of microseconds. However, for the RAM based FPGAs the initialization time is in the order of milliseconds and it depends on the configuration memory size of the FPGA

and the bandwidth of the non-volatile memory for the transfer of the configuration file from non volatile memory to the FPGA.

In the ML507 demo board, Xilinx PROM is used to store the configuration memory. After power up, the configuration file is loaded from PROM to the FPGA and if FPGA is successfully configured, it starts to run. Successful configuration can be understood by looking the “DONE” and “ERROR” leds on the demo board. “DONE” led is ON means FPGA is configured successfully, if FGPA is not configured properly, then “ERROR” led will be ON.

4.1.2.3 LINEAR FLASH CHIP

FPGA designs that incorporate software embedded systems using processors also utilize external volatile memory to execute the software code. Systems using volatile memory must include a non-volatile device to store the software program during the power down. After power up, a small application which is stored in the configuration memory starts and reads the software application from the non-volatile memory to the volatile memory. Therefore, in order to store the software application a linear flash device of Intel (JS28F256P30T95) is used. The flash interface of the device is given in Figure 4-2.

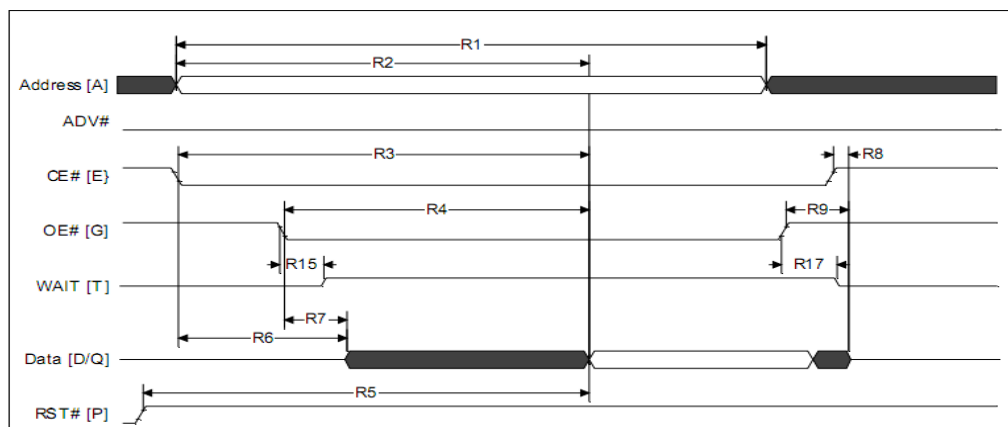


Figure 4-2 Flash Interface Timing Diagram

4.1.2.4 DDR2 SDRAM

The ML507 Demo Board has a single-rank unregistered 256 MB SODIMM. DDR2 SDRAM. DDR2SDRAM is used for the application memory, after the software application is copied from Flash to the DDR2SDRAM, application will start to run on the DDR2SDRAM. ML507 board has also an SRAM in which the application memory can be stored. However, SRAM is much slower and more expensive, so DDR2SDRAM is selected for the application memory.

4.1.2.5 16X2 CHARACTER LCD

Character LCD is used to print the result. Since the LCD has two lines, and each line has 16 characters, it can be written totally 32 characters for the sign recognition result which is enough to print the recognition result without any shortening the words.

The physical interface to the LCD is made through FPGA. The signal timing requirements of the LCD are given Figure 4-3. The functions contained in the FPGA application controls what is shown on the LCD adjusting the timings of data and control signals according to the Figure 4-3.

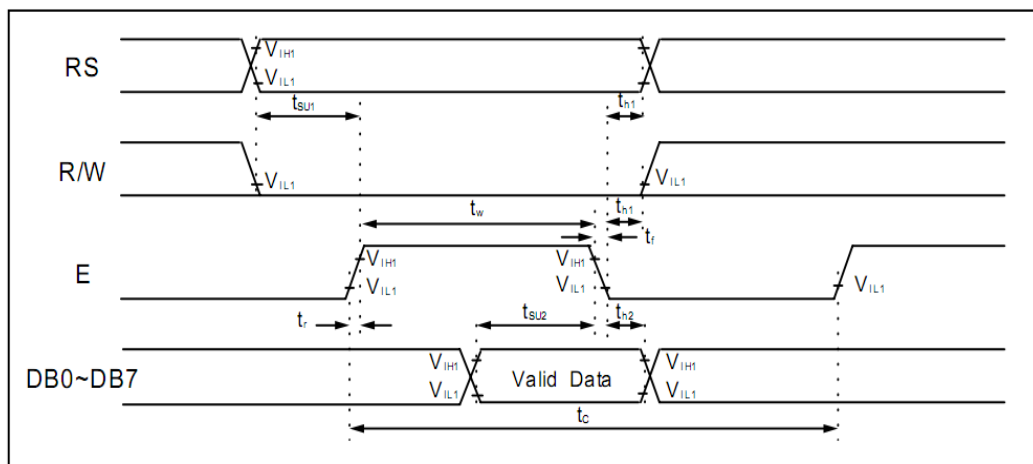


Figure 4-3 Timing Diagram of the Character LCD

4.1.3 SYSTEM OVERVIEW

The block diagram of the hardware architecture and flow chart of the power up initialization sequence are shown in Figure 4-4 and Figure 4-5, respectively. After power up, FPGA is configured from the PROM. After FPGA is successfully configured, in the processor side, a small bootloader application starts to run and read the application from the BPI flash and copy it to the DDR2SDRAM. When the copying is finished, program counter is set to the application starting address of the RAM and application starts to execute.

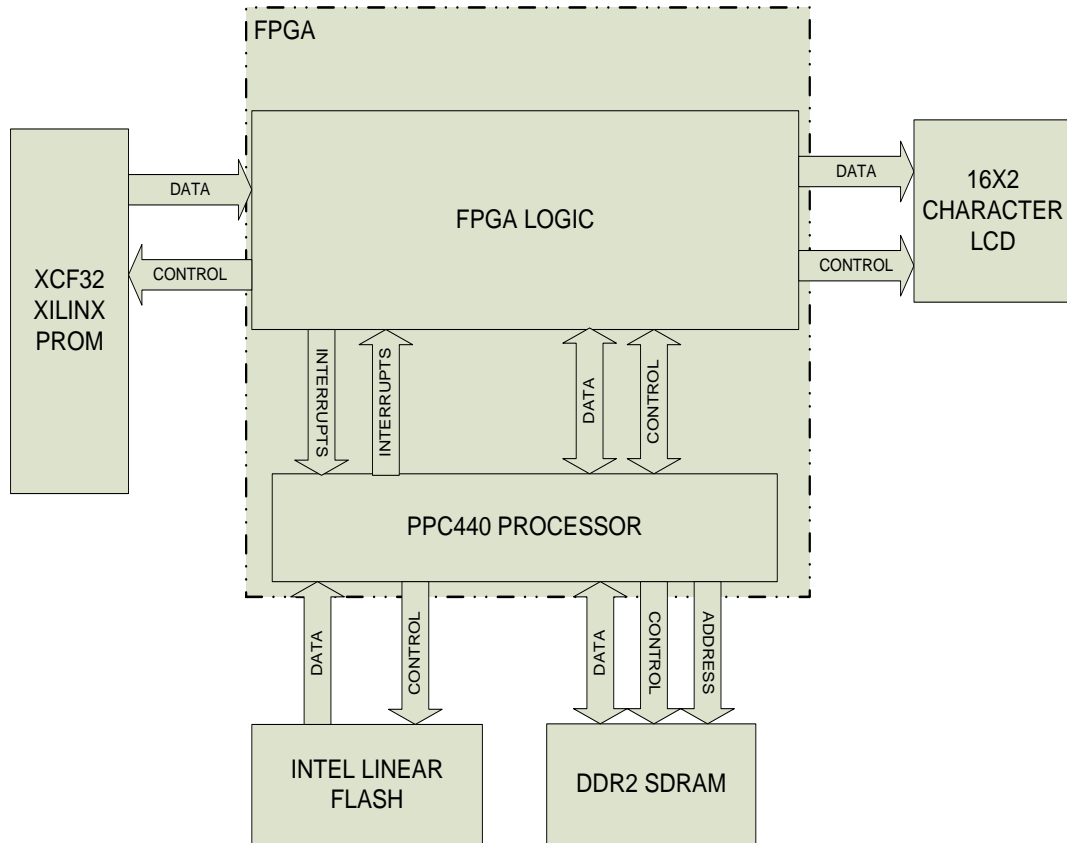


Figure 4-4 Hardware Architecture of the System

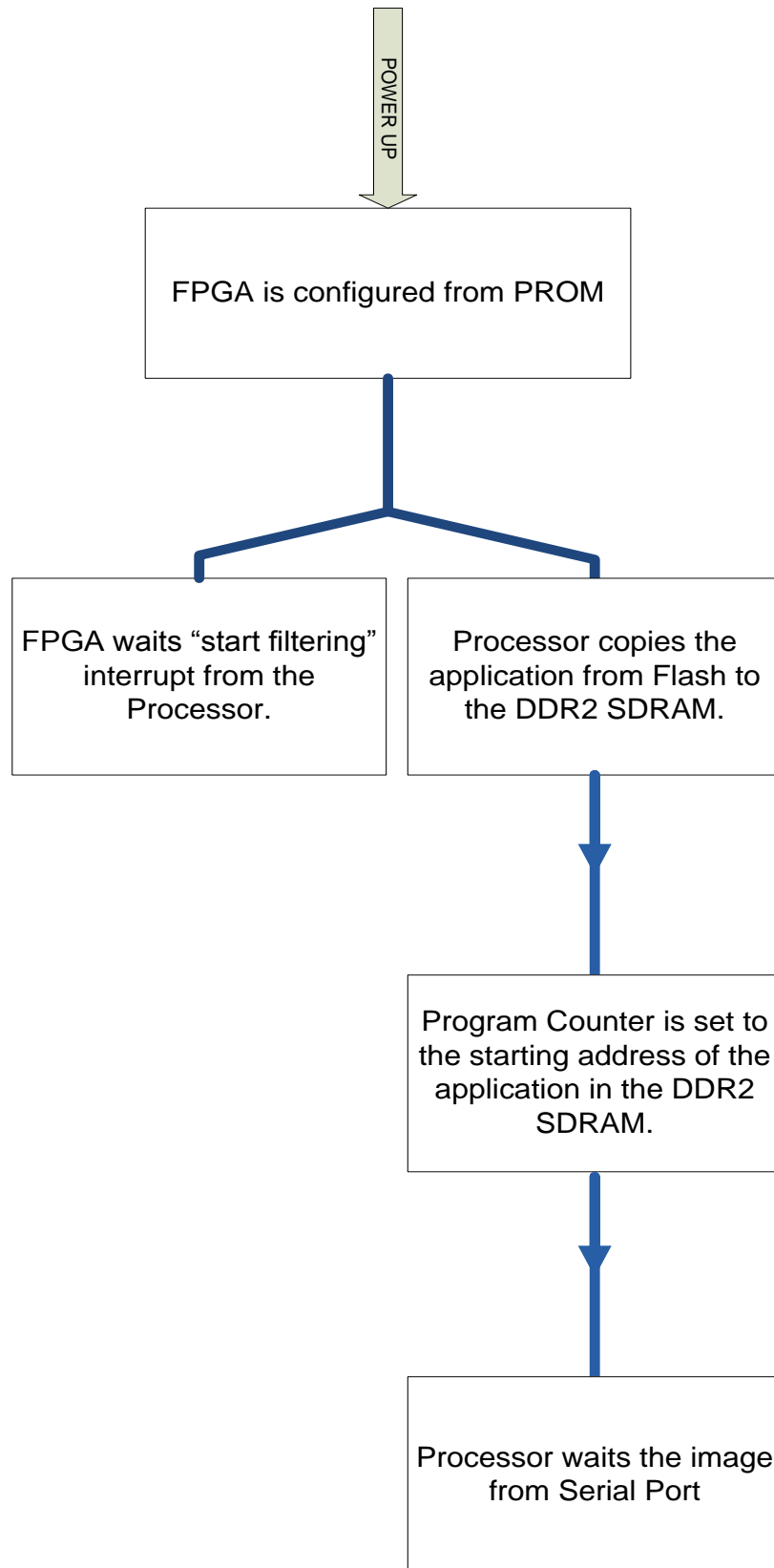


Figure 4-5 Flowchart of the Power Up Initialization Sequence

4.2 FPGA ARCHITECTURE

4.2.1 INTRODUCTION

Virtex-5 FX70T FPGA has an embedded PPC440 processor in it. The benefit of using an embedded processor in FPGA is to create hardware and software platform for both hardware and software engineers in a single programmable logic device. Hardware engineers are using FPGA logic to create hardware designs and software engineers are using processor to create software designs. Combining these two abilities makes a high performance embedded application in a single chip. In this single chip, the workload is divided between FPGA and processor, since both have some advantages and disadvantages as compared to the other. Using an FPGA in the projects requires longer design times, and the ability to change the designs is severely limited. However, the main advantage of FPGA designs is having parallel processing capabilities and pipelined operations, which decrease the execution times. On the other hand, processor based designs are easy to debug, test and verify.

Main challenge in these mixed platforms is the decision regarding division of workload. The parts, which require complex operations, should be done in the FPGA side. The parts, which require simple operations, loops or recursive operations, should be done in the processor side. They are suitable for processor-based architectures instead of logic-based architectures.

Each step of TSR system is assigned to either FPGA or PPC considering points discussed above. The system starts by loading the image. The image patch is loaded using PC via the serial port. A GUI is designed to load image and show the calculated results in the intermediate steps. It works as a TSD module. The detailed description of the developed GUI is given in Appendix B. The system stops with displaying the recognition result on the LCD. The block diagram of the system is shown in Figure 4-6. Each step is explained in detail in the following sections. For the processor side blocks, the pseudo code of the algorithm is given; for the FPGA

side blocks, the hardware block diagram of the block is given for a better explanation the system.

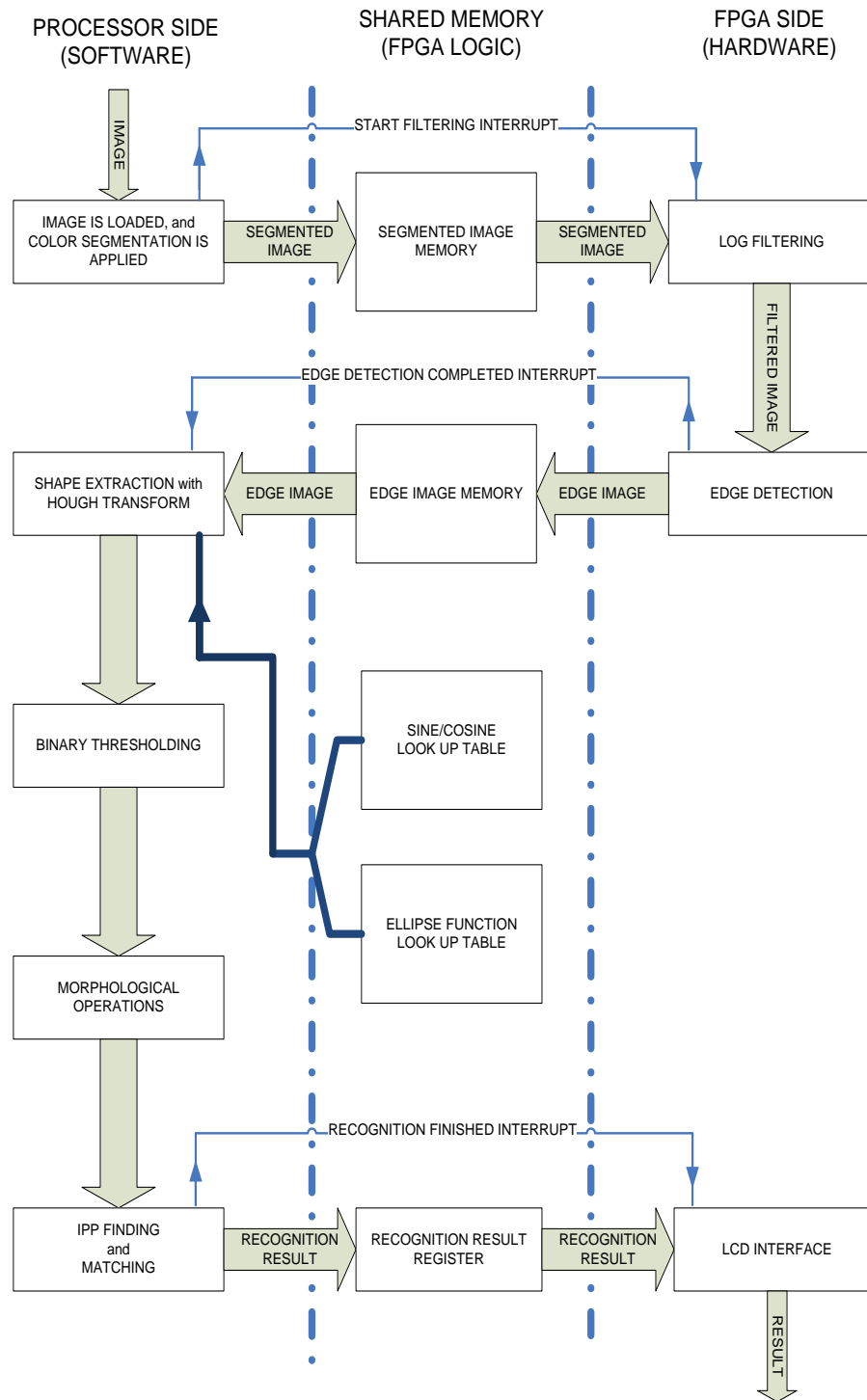


Figure 4-6 Block Diagram of Traffic Sign Recognition System

4.2.2 COLOR SEGMENTATION

Color Segmentation is a simple module which only compares blue pixel value with the twice of red and green pixel values in blue segmentation and similarly red pixel value with the twice of blue and green pixel values in red segmentation as explained in Section 3.1.1. This step is done by PPC processor because this is not a complex operation and requires only a comparison.

After color segmentation is completed, a “segmentation completed” interrupt is sent to the FPGA in order to start the edge detection process. The pseudo code of the color segmentation is given below.

```
for x from 1 to 60  
  for y from 1 to 60  
    if red component is greater than twice of blue or green component at (x,y)  
      set the pixel at (x,y) for red segmented image  
    if blue component is greater than twice of red or green component at (x,y)  
      set the pixel at (x,y) for blue segmented image
```

4.2.3 EDGE DETECTION

The extraction of the shape of traffic sign is very crucial step for the sign recognition. In order to recognize traffic sign, the borders of the sign should be detected with high accuracy. Thus, in the edge detection step, number of non-detected edge pixels and false edges should be as small as possible. This requirement necessitates the usage of a good edge detector.

4.2.3.1 LOG FILTERING

In TSR system, Laplacian of Gaussian (LoG) edge detector is used as explained in Section 3.1.2. The kernel of the LoG edge detector is given in Figure 4-7;

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 4-7 Kernel of LoG Edge Detector

Since the kernel of the LoG edge detector is 9x9 in size and the elements of the kernel is not small numbers, this step requires many arithmetic operations such as multiplication and addition. In addition, these operations can be executed in a pipelined and parallel manner. Thus, LoG filtering is done in the FPGA side using the dedicated multipliers, accumulators of the FPGA. The elements of the Kernel of LoG Edge Detector are stored in the block RAMs of the FPGA. Hardware block diagram of the LoG Filter Block is shown in Figure 4.8.

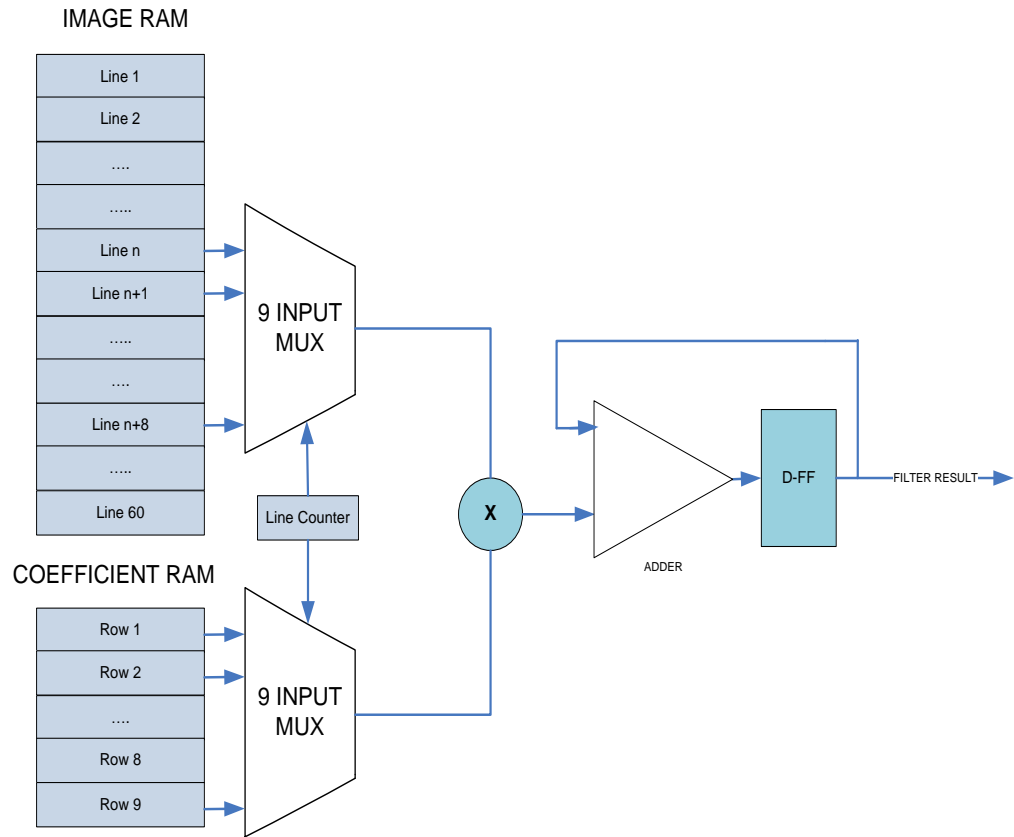


Figure 4-8 Hardware Block Diagram of LoG Filter Block

4.2.3.2 ZERO CROSSING DETECTOR

After the image is filtered, the next step is finding the edges. As explained in Section 3.1.2.2, zero crossed points of the filtered image are the edge pixels in the image. Zero Crossing Detector block detects this zero crossing points at which the sign changes occur in both x and y direction. Truth tables used in finding the edges in x and y direction, and the edges in the image are shown in Table 4-1, Table 4-2 and Table 4-3. From the tables, logical XOR of the sign bits in the neighboring pixels gives the edges in x and y direction. Logical OR of these edges gives the edges in the image. Using the truth tables, a hardware system is designed for the FPGA. Hardware Block Diagram of the Zero Crossing (Edge) Detector is shown in Figure 4-9.

Table 4-1 Truth Table used to find the edges in x direction

Sign at (x,y)	Sign at (x+1,y)	Edge Value in x direction
Positive	Positive	0
Positive	Negative	1
Negative	Positive	1
Negative	Negative	0

Table 4-2 Truth Table used to find the edges in y direction

Sign at (x,y)	Sign at (x,y+1)	Edge Value in y direction
Positive	Positive	0
Positive	Negative	1
Negative	Positive	1
Negative	Negative	0

Table 4-3 Truth Table used to find the edges in the image

Edge Value in x direction	Edge Value in y direction	Edge Value at (x,y)
0	0	0
0	1	1
1	0	1
1	1	1

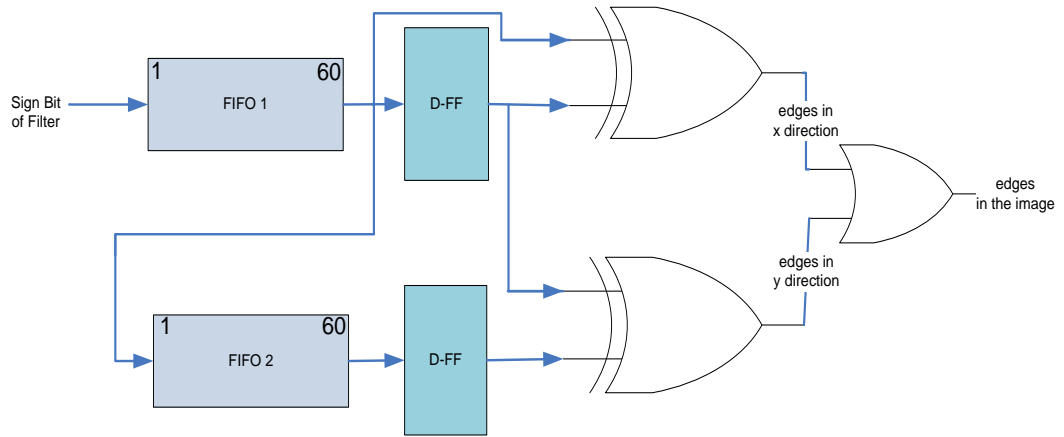


Figure 4-9 Hardware Block Diagram of Zero Crossing Detector

When the detected zero crossing points, which are the edge pixels, are calculated, it is written to the shared memory and an “edge detection completed” interrupt is sent to the processor in order to trigger the processor to start the shape extraction process.

4.2.4 SHAPE EXTRACTION

4.2.4.1 TRIANGULAR SHAPE EXTRACTION

Hough Transform is used for the triangular shape extraction step. Actually, Hough Transform is a very time consuming algorithm for real time applications, since for each edge pixel all the possible combinations of the parameters are used while creating the accumulator array. In the line detection, radius and angle are the variable parameters. So, for each edge pixel in the image all possible radius and angle combinations are used to create the accumulator array. Radius values are calculated using the angle and the edge pixel locations (x,y). The pseudo code of the algorithm is given below;

```

for x from 1 to 60
  for y from 1 to 60

```

```

if edge exists at (x,y)
  for angle from 0 to 180
    calculate radius using  $r = x \cdot \cos(\text{angle}) + y \cdot \sin(\text{angle})$ 
    update accumulator array at (r,angle) index

```

Using the pseudo code of the algorithm, the time required to create accumulator array can be calculated. Two multiplications, one addition and calculation of cosine and sine of the angle are required for radius calculation. Assuming that color segmentation is not used and 30% of the pixels are the edge pixels in the image, then line detection algorithm requires;

*total number of multiplication = $60 \times 60 \times 0.30 \times 180 \times 2 = 388,800$ multiplications

*total number of addition = $60 \times 60 \times 0.30 \times 180 \times 1 = 194,400$ additions

*cosine/sine calculation = $60 \times 60 \times 0.30 \times 180 \times 2 = 388,800$ cosine/sine calc.

In order to decrease the complexity, color segmentation is used and valid sign assumption is made. A triangular sign is validated if the angle between any two line segments is between 54 and 66 degrees (60 ± 6 degrees). Using this assumption, some simplifications can be done. First, only related angles (i.e. 0, 60, and 120 degree) and ± 6 degree vicinity of these angles are searched for the lines. Second, since the number of angles is small, there is no need sine and cosine calculation, the values of sine and cosine functions can be stored in look up tables using the FPGA logic. Using the above facts, the new pseudo code becomes;

```

initialize cos[] array for the related angles
initialize sin[] array for the related angles
for x from 1 to 60
  for y from 1 to 60
    if edge exists at (x,y)
      for angle from 0 to 6

```

```

    calculate radius using  $r=x.\cos[angle] + y.\sin[angle]$ 
    update accumulator array
for angle from 54 to 66
    calculate radius using  $r=x.\cos[angle] + y.\sin[angle]$ 
    update accumulator array
for angle from 114 to 126
    calculate radius using  $r=x.\cos[angle] + y.\sin[angle]$ 
    update accumulator array
for angle from 174 to 180
    calculate radius using  $r=x.\cos[angle] + y.\sin[angle]$ 
    update accumulator array at  $(r,angle)$  index

```

Applying color segmentation and making optimization decrease the complexity. Utilizing color segmentation decreases the number of detected edge pixels from 30% to 5% on the average. Then, using the optimized method and color segmented image, the number of operations becomes;

*total number of multiplication = $60 \times 60 \times 0.05 \times 36 \times 2 = 12,960$ multiplications

*total number of addition = $60 \times 60 \times 0.05 \times 30 \times 1 = 6,480$ additions

*no cosine/sine calculations

As compared the non optimized case, the number of multiplication and addition operations decreased to 3% at the optimized case. In addition, the calculation of sine and cosine of the angles are eliminated in the optimized case.

After creation of accumulator array, the two peak points of the accumulator array at 0, 60, 120 degree and around are found, one for inner border and one for outer border as explained in Section 3.1.3.1. Inner part of the inner triangle is the extracted ROI.

4.2.4.2 CIRCULAR SHAPE EXTRACTION

In order to extract circular shapes, a modified version of CHT is used. As explained in Section 3.1.3.2. The parametric equation of an ellipse is given in (4.1);

$$(x - x_c)^2 + k.(y - y_c)^2 = r^2 \quad (4.1)$$

Using this equation, for each edge pixel at a location (x,y), all the possible (x_c, y_c, k, r) combinations are used to create the accumulator array. Therefore, accumulator array is a four dimensional array. The parameter “k” is included because of the geometric distortion of circular signs. It is assumed that “k” can take the values between 0.5 and 2. When k is equal to one, then the searched sign becomes a circle. The pseudo code of the algorithm is;

```
for x from 1 to 60
  for y from 1 to 60
    if edge exists at (x,y)
      for xc from 1 to 60
        for yc from 1 to 60
          for k from 0.5 to 2
            calculate radius using  $r = \sqrt{(x-xc)^2 + k(y-yc)^2}$ 
            update accumulator array at the (xc,yc,k,r) index
```

This circular shape extraction algorithm is much more complex than the line detection algorithm. As seen from the pseudo code, the algorithm consists of five loops, and for each radius calculation, there are one addition, two subtractions, three multiplications and one square root operation. Assuming color segmentation is not used and total number of edge pixels is 30% of the whole image, then total number of operations can be calculated as;

*total number of addition = $60 \times 60 \times 0.3 \times 60 \times 60 \times 16 = 62,208,000$ additions

*total number of subtraction = $60 \times 60 \times 0.3 \times 60 \times 60 \times 16 \times 2 = 124,416,000$ subtractions

*total number of mult. = $60 \times 60 \times 0.3 \times 60 \times 60 \times 16 \times 3 = 186,624,000$ mults

*total number of square root operation = $60 \times 60 \times 0.3 \times 60 \times 60 \times 16 = 62,208,000$
square root operations

It is impossible to do all these operations in the order of milliseconds even using the highest performance computers. In order to create the accumulator array in PPC440 processor, some optimizations are required. Therefore, three optimizations are done to decrease the complexity. These are,

*Color segmentation is used to decrease the number of edge pixels.

*Assuming that in the 60x60 patch image, center of the traffic sign coordinate is between (20,20) and (40,40). In other words, $20 < x_c < 40$ and $20 < y_c < 40$.

* Instead of calculating the radius, a look up table is generated using the FPGA logic. Look up table is a three dimensional array and in the address (x_c, y_c, k) , the corresponding radius (r) is written. Therefore, no square root calculation is required.

The pseudo code of the optimized algorithm is;

```
for x from 1 to 60
  for y from 1 to 60
    if edge exists at (x,y)
      for xc from 20 to 40
        for yc from 20 to 40
          for k from 0.5 to 2
            update accumulator array at the  $(x_c, y_c, k, r)$  index
```

In the optimized algorithm, there is no addition, no subtraction, no multiplication and no square root operation. The only thing is the updating of the accumulator array which is a one clock cycle operation. Thus, total number of clock cycles to create the accumulator array (assuming 5% pixels of the image are the edge pixels);

total number of clock cycles= $60 \times 60 \times 0.05 \times 20 \times 20 \times 16 = 1,152,000$ clocks

Total number of clock cycles in the optimized case is much lower than the non-optimized case. In the non-optimized case, only making multiplication operation lasts;

Total number of clock cycles for multiplication= $T = n \times t$

n =total number of multiplication = $60 \times 60 \times 0.3 \times 60 \times 60 \times 16 \times 3 = 186,624,000$

t = number of clocks for one multiplication = 6 clocks

$T = 186,624,000 \times 6 = 1,119,744,000$ clocks

Even only multiplication lasts 1000 times longer as compared with the optimized case.

4.2.4.3 RECTANGULAR SHAPE EXTRACTION

Rectangular Shape Extraction has the simplest algorithm of three shape extraction algorithms. Since the lines in the rectangular are 0 and 90 degrees and the sine and cosine of these angles are “0” or “1”, then it becomes a simplified version of line detection algorithm. Using x and y projections of the image gives the information of the 0 and 90 degree lines in the image as explained in Section 3.1.3.3. In the projections, the two peak histogram values are the two lines in the image. Using x projection of the image, 90 degree lines can be detected and using the y projection 0 degree lines can be detected. In order to eliminate the affect of small rotations in the traffic sign, projections are created also using the neighbor pixels. The pseudo code of the algorithm is given below;

```
for x from 1 to 60
  for y from 1 to 60
    if edge exists at (x,y) or (x+1,y)
      increment the y projection by one at point y
```

if edge exists at (x,y) or (x,y+1)
increment the x projection by one at point x

After creating the projections, in each projection the two peak histogram are taken as the detected lines. Thus, detecting these four lines means the extraction of the rectangle. Inner part of these four lines is the ROI.

4.2.5 BINARY THRESHOLDING

Image is binarized using a threshold, which is the average pixel value of the ROI, as explained in Section 3.2.1. Since green image is saved to the block rams of the FPGA in the image loading process. Processor reads the pixel values of ROI from the block ram, and calculate the average pixel value, which is the threshold value for the binarization process. Then, the green image pixel values are compared with the threshold and the corresponding binary values are written to the block rams again. Actually, calculation of average pixel value which is equal to the sum of ROI pixel values divided by the total number of pixels in ROI, should be done on the FPGA side is more efficient, however doing that work on the processor side can be acceptable because ROI is much smaller than the 60x60 image patch, so less number of operations does not last long times. In addition, this makes the design more flexible, since changing the threshold function is an easy task for the processor unless a complex threshold function is selected.

4.2.6 MORPHOLOGICAL OPERATIONS

In order to eliminate the border pixels in the ROI, morphological opening is applied to the binary image. First, binary image is eroded to remove the border pixels and then dilated to recover the effects of erosion on the ROI.

4.2.6.1 EROSION

Binary Image is eroded with the following mask;

1	1
1	1

This means that if all the elements in a 2x2 square in the binary image are “1”, then pixel location of the first element of that 2x2 square in the binary image is “1” in the eroded image. The pseudo code of the algorithm is given below;

```
for x from 1 to 60  
  for y from 1 to 60  
    if binary_image exists at (x,y) and (x+1,y) and (x,y+1) and (x+1,y+1)  
      set eroded_image one at point (x,y)
```

4.2.6.2 DILATION

Binary Image is dilated with the following mask;

1	0
0	0

This means that if a single pixel is “1” in the eroded image in a 2x2 square, then set all the pixels “1” at a 2x2 square in the dilated image starting from the location of that single pixel. The pseudo code of the dilation is given below;

```
for x from 1 to 60  
  for y from 1 to 60  
    if eroded_image exists at (x,y)
```

set dilated_image one at point (x,y) and (x+1,y) and (x,y+1) and (x+1,y+1)

4.2.7 IPP FINDING AND MATCHING

In the shape extraction part, the exact location of the shape is extracted. After that step, sign is divided into regions. The number of divided regions is six for triangular, five for circular and four for rectangular traffic signs. The divided regions are explained in Section 3.2.3. For each divided part, the number of informative pixels is calculated and then they are normalized. In other words, result of calculation shows the percentage of informative pixels in each region.

IPP calculation is done by the processor. Since it is only a counting operation and does not require complex additions. The pseudo code is given below.

```
clear IPPs of all regions
for x from 1 to 60
  for y from 1 to 60
    if dilated_image exists at (x,y)
      switch location of (x,y)
        region one
          increment IPP of region one by one
        region two
          increment IPP of region two by one
        .
        .
        region n
          increment IPP of region n by one
      normalize IPPs
```

After IPP of each region is calculated, these percentages are compared with the template signs in the database with the class of the input sign. In section 3.2.3.2, it

is explained that the database consists of four main sign classes, namely triangular sign classes, circular red sign classes, circular blue sign classes, and rectangular sign classes. In the comparison part, the template sign with the minimum SAD with the input sign is the recognition result.

When the recognition result is found, the numerical code of the sign is written to the shared memory in order to read the result by FPGA for the LCD interface.

4.2.8 LCD INTERFACE

16x2 CHARACTER LCD is used to show the recognition result on the board. When the recognition process is finished in the processor side, processor sends a “recognition finished” interrupt to the FPGA with writing the recognition result to the shared memory. FPGA decodes the recognition result written by processor and using the database displays the result on the LCD. The block diagram of the LCD interface is given in Figure 4-10.

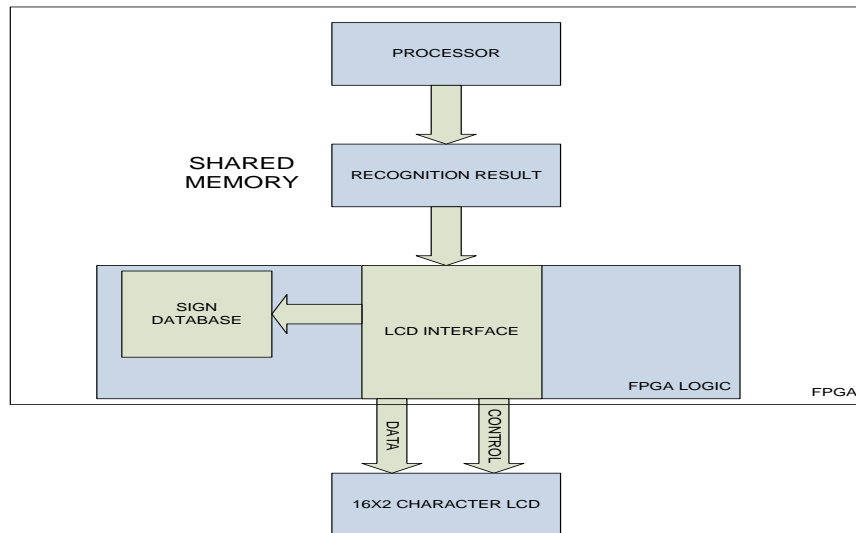


Figure 4-10 Block Diagram of the LCD Interface

CHAPTER 5

EXPERIMENTAL RESULTS

5.1 INTRODUCTION

The traffic sign recognition algorithm is first developed and verified in MATLAB. After the development is finished and the verification is completed, the algorithm is converted to the HDL code for the FPGA. In addition, in each step in the FPGA implementation, the MATLAB and FPGA results are compared.

For the testing of the algorithm on the FPGA, 60x60-image patches are loaded to the FPGA via Serial Port using PC. After the image is successfully taken, the FPGA starts to run and displays the result on the character LCD.

5.2 TEST RESULTS

Two sets of images with the following properties are used in testing of the FPGA implementations of algorithms.

Set-1: This set includes well illuminated 134 images which contains 51 triangular, 69 circular (39 red, 30 blue signs) and 14 rectangular traffic signs. An example for Set-1 images is given in Figure 5.1. The recognition results of the Set-1 images are given in Table 5-1.



Figure 5-1 An example Traffic Sign from Set-1

Set-2: This set consists of poorly illuminated, shadowed or occluded 51 images which contains 17 triangular, 24 circular (15 red, 9 blue signs) and 10 rectangular traffic signs. An example for Set-2 images is given in Figure 5.2. The recognition results of the Set-2 images are given in Table 5-2.



Figure 5-2 An example Traffic Sign from Set-2

Table 5-1 Recognition Results of Set-1 Images









SET-1	Triangular	Circular Red	Circular Blue	Rectangular	Total
Number of Signs	51	39	30	14	134
Number of Recognized Signs	47	38	28	14	128
Recognition Rate	92%	98%	97%	100%	96%

Table 5-2 Recognition Results of Set-2 Images

SET-2	Triangular	Circular Red	Circular Blue	Rectangular	Total
Number of Signs	17	15	9	10	51
Number of Recognized Signs	12	14	8	9	43
Recognition Rate	71%	93%	89%	90%	84%

The recognition results in Table 5-1 and Table 5-2 show that recognition rate is highest for rectangular signs from Set-1 and lowest for triangular signs from Set-2. The main reason of that is the number of the signs which are close to each other in terms of IPPs is highest for triangular signs. Some of the wrong recognitions are shown in Table 5-3. Moreover, Set-1 recognition rates are higher than Set-2 rates for all the shapes because in Set-2 images, segmentation and edge detection have lower performance and occlusion affects the shape extraction.

Table 5-3 Wrong Recognitions

Input Sign				
Recognition Result				

5.3 RESOURCE USAGE

The most important utilization parameter in the resource usage is the number of used Flip Flops in FPGA. Unused Flip Flops shows the availability of FGPA logic.

The second important parameter in utilization is the used memory in FPGA. FPGAs have dedicated on chip RAMs. Smaller memory requirements can be met using the on chip RAMs.

In Table 5-4, resource utilization of the TSR System is given. In the system, only 13% of the FPGA logic (Flip Flops) and 29% of the FPGA memory is used. This means that the TSR system can be implemented in a smaller and cheaper FPGA.

Table 5-4 Resource Utilization of the Traffic Sign Recognition System

Device Utilization Summary			
Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	5,890	44,800	13%
Number used as Flip Flops	5,890		

Table 5-4 Resource Utilization of the Traffic Sign Recognition System (Cont'd)

Number of Slice LUTs	6,842	44,800	15%
Number used as logic	6,458	44,800	14%
Number using O6 output only	5,862		
Number using O5 output only	205		
Number using O5 and O6	391		
Number used as Memory	349	13,120	2%
Number used as Dual Port RAM	108		
Number using O5 output only	32		
Number using O5 and O6	76		
Number used as Shift Register	241		
Number using O6 output only	241		
Number used as exclusive route-thru	35		
Number of route-thrus	249	89,600	1%
Number using O6 output only	234		
Number using O5 output only	9		
Number using O5 and O6	6		
Slice Logic Distribution			

Table 5-4 Resource Utilization of the Traffic Sign Recognition System (Cont'd)

Number of occupied Slices	3,818	11,200	34%
Number of LUT Flip Flop pairs used	9,310		
Number with an unused Flip Flop	3,420	9,310	36%
Number with an unused LUT	2,468	9,310	26%
Number of fully used LUT-FF pairs	3,422	9,310	36%
Number of unique control sets	731		
Number of slice register sites lost to control set restrictions	1,605	44,800	3%
IO Utilization			
Number of bonded IOBs			
Number of bonded	170	640	26%
IOB Flip Flops	298		
Specific Feature Utilization			
Number of BlockRAM/FIFO	43	148	29%
Number using BlockRAM only	41		
Number using FIFO only	2		
Total primitives used			
Number of 36k BlockRAM used	40		

Table 5-4 Resource Utilization of the Traffic Sign Recognition System (Cont'd)

Number of 18k BlockRAM used	2		
Number of 36k FIFO used	2		
Total Memory used (KB)	1,548	5,328	29%
Number of BUFG/BUFGCTRLs	6	32	18%
Number used as BUFGs	6		
Number of BUFIOs	8	80	10%
Number of DSP48Es	4	128	3%
Number of PLL_ADVs	1	6	16%
Number of PPC440s	1	1	100%
Number of RPM macros	64		

5.4 EXECUTION TIMES

In the TSR system, FPGA is running with a 100 MHz clock. In the FPGA, the clock is doubled for the processor so the processor is running at 200 MHz. The duration of the operations on the FPGA side is always the same. The total duration of filtering and edge detection operations is 2.9 ms independent of the sign size in the 60x60 image patch. However, the processor side operations depend on the sign size. If the sign is small in the 60x60 image patch, then the total duration is small. Increasing the sign size in the 60x60 image patch increases the execution times. Therefore, execution times are measured in the worst cases, the traffic sign is nearly 60x60 pixels in size in the 60x60 image patch. Execution times of the operations for each sign shape are shown in Table 5-5. From the figure, it is seen that red circular

signs have the longest execution time, it is nearly 70 ms. In 70 ms, a car can go 2 meters on the average with a 75 km/h speed which is an acceptable distance for the sign recognition systems. The execution time is much smaller for triangular and rectangular signs. The main time consuming operation is the shape extraction using Hough Transform in TSR system. Execution time for creating the accumulator array is nearly 90 % of the whole process time for circular signs. If in the shape detection module, which sends the 60x60 pixels image patch with the shape information, shape center is detected accurately, the execution times will be less than 8 ms for all shapes, since Hough Transform will be removed in the TSR system.

Table 5-5 Execution Times of the TSR System

Shape Operation	TRIANG.	CIRCULAR RED	CIRCULAR BLUE	RECT.
Color Segment.	0.547 ms	0.550 ms	0.546 ms	0.549 ms
Edge Detection	2.916 ms	2.916 ms	2.916 ms	2.916 ms
Creating Accumulator	3.634 ms	62.694 ms	51.200 ms	0.255 ms
ROI Extraction	1.778 ms	3.902 ms	3.906 ms	0.005 ms
Binary Thresholding	0.248 ms	0.203 ms	0.183 ms	0.169 ms
Morphological Operations	0.644 ms	-	-	-
IPP Calculation	1.913 ms	0.304 ms	0.294 ms	0.144 ms
IPP Match	0.015 ms	0.005 ms	0.002 ms	0.002 ms
Total Exec. Time	12.12 ms	70.574 ms	59.047 ms	4.040 ms

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 CONCLUSIONS

In this thesis, recognition of the traffic signs is studied and implemented on FPGA. The algorithm developed for the FPGA implementation is a combination of previous algorithms and new techniques such as IPPs matching. Implemented algorithm is tested with 60x60 image patches.

In the proposed algorithm, in the first step, red and blue color segmentation is used. This process decreases the complexity and increases the performance. Then, segmented image is used for the LoG edge detection algorithm. LoG edge detector is preferred because of its good edge detection capability. The Gaussian Smoothing removes the high frequency noise and Laplacian Operator finds the correct places of edge and tests wider areas around the pixel [31]. Moreover, unlike the first derivative operators, there is no threshold parameter adjustment in LoG edge detector. Its performance is very close to the Canny Edge Detector, which is the optimal edge detector.

In the shape extraction step, Hough Transform is applied to the edge image. The main advantage of the Hough Transform is that it is tolerant of the gaps in feature boundary. Moreover, it is relatively unaffected by image noise [32]. For the extraction of triangles and rectangles, line detection with Hough Transform is used. For the extraction of circles because of the perspective distortion), a modified

version of CHT is used. After shape is extracted, ROI is taken as the inner part of the shape.

In order to recognize traffic signs, ROI is divided into smaller regions and the IPPs are calculated for these regions. The number of divided regions depends on the sign shape and it is determined by the number of template sign in the database for that shape. Higher number of signs in database for a class of shape results in more divided regions in the shape. The main reason of using this technique is its scale invariance due to the percentage usage in matching process. Lower memory requirements and lower complexity are the other advantages of this technique. IPPs are found by counting the informative pixels in each region and normalizing the sum of informative pixels for each region. After calculating IPPs, these IPPs are compared with the database IPPs for each sign; best match is outputted as the recognized sign.

The algorithm is implemented on Virtex-5 FX70T FPGA which has a hard PowerPC440 processor. The workload of the algorithm is divided between FPGA logic and processor in it according to the capabilities of FPGA and processor. FPGA is running with a 100 MHz clock and the processor is running at 200 MHz. Both FPGA and processor can be run at higher frequencies. However, bottleneck of the system is the memory read/write operations which is determined by the processor bus clock frequency, so increasing the FPGA and processor clock frequencies would not increase the performance much further. Therefore, to decrease the execution times, the bus clock runs at the maximum bus operating frequency which is 125 MHz. As a result, implemented algorithm has low execution times, although it has high computational requirements. The execution time varies between 4-70 ms according to the size and shape of the traffic sign. Circular signs require more time to be recognized. In the worst case, if the sign is a circular with size 60x60, the execution time is around 70 ms. During this worst case execution time period, a car is going only two meters (with 75 km/h average speed) before the sign is recognized. Execution time for creating the accumulator array in the shape extraction step is nearly 90 % of the whole process time for circular signs. If in the

shape detection shape center is detected accurately, the execution times will be less than 8 ms for all shapes.

The recognition rate of the TSR system is higher than %95 for well illuminated images (Set 1). For poorly illuminated and shadowed images (Set 2), recognition rate is about %84. The main reason for wrong recognitions is unsuccessful color segmentation especially in poorly illuminated images. In addition, IPP based recognition is adversely affected by small ROI extraction errors in the shape extraction process which also decreases the recognition rate for all set of images.

The implemented algorithm uses 13% of the FPGA logic resources which is well under the capabilities of the chip. In addition, the other used components FLASH, PROM, and DDR2 SDRAM are very cheap since they are abundantly available. Thus, FPGA is the main costly part of the system and using a cheaper FPGA will decrease the cost of the system substantially.

6.2 FUTURE WORK

First, image statistics can be used to optimize the system parameters. Performance can be increased using optimized parameters for each set. Moreover, HSV color space can be used because of its robustness against variable conditions of luminosity.

In addition, the performance of the TSR system can be improved with increasing the number of divided regions. However, since execution time is important, complexity should not be increased much by adding more divided regions.

In order to make a final product, the TSR system can be combined with the shape detection process. Moreover, with the low demand for resources, the TSR system can be used as a subsystem in the electronic system of the car to improve the safety.

REFERENCES

1. Ezine Articles. <http://ezinearticles.com/?ADAS-Defined---Advanced-Driver-Assistance-Systems&id=3006760>. last accessed date: June 3, 2010
2. Volkswagen. <http://www.vw.co.za/about/technology/future/>. last accessed date: June 3, 2010
3. Mobileye. <http://www.mobileye.com/menu/1/2/20/52/60?SESS35bb6244eaa991f119e911f90d2a1fa8=5bb9975d4ea197f91bf647cafa053f59>. last accessed date: June 3, 2010
4. Wikipedia. http://en.wikipedia.org/wiki/Advanced_driver_assistance_systems . last accessed date: June 3, 2010
5. Aryuanto Soetedjo and Koichi Yamada: Fast and Robust Traffic Sign Detection, Proceedings of IEEE System, Man and Cybernetics (SMC) 2005, pp.1341-1346, Hawaii, USA 2005.
6. Andrzej Ruta, Yongmin Li, Xiaohui Liu. Traffic Sign Recognition Using Discriminative Local Features, IDA 2007, pp.355-366
7. Volker Rehrmann, Raimund Lakmann, Lutz Priese. A parallel system for real-time traffic sign recognition, International Workshop on Advanced Parallel Processing Technologies Northern Jiaotong University, Beijing, China September 26-27, 1995
8. Wang Yongping, Shimeiping, Wu Tao. A Method of Fast and Robust For Traffic Sign Recognition, Fifth International Conference on Image and Graphics Xi'an, Shanxi, China September 20-23, 2009
9. L. Pacheco, J. Batlle, X. Cufi. A New Approach to Real Time Traffic Sign Recognition Based on Colour Information, Proceedings of the Intelligent Vehicles Symposium, pp. 339-344, October, 1994.
10. Shuangdong Zhu, Lanlan Liu, Xiaofeng Lu. Color-Geometric Model for Traffic Sign Recognition, CESA, October 4-6, 2006.
11. H. Fleyeh. Color Detection and Segmentation for Road and Traffic Signs, Cybernetics and Intelligent Systems, 2004 IEEE Conference, vol. 2, pp. 809-814, 2004.

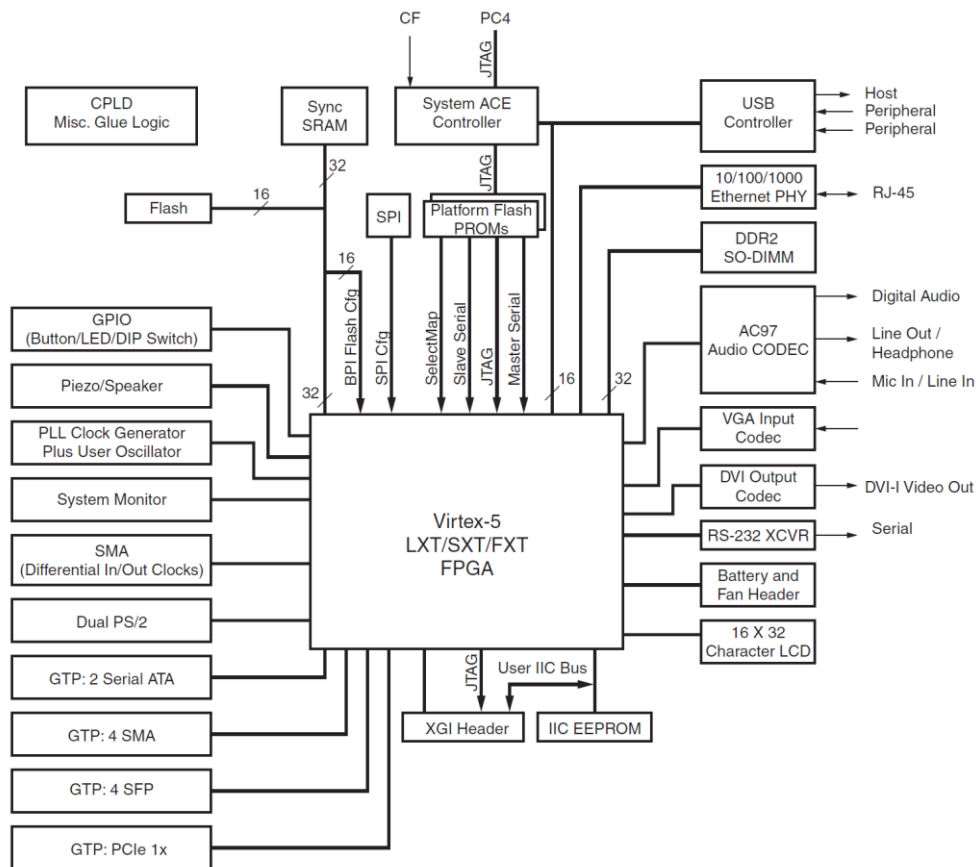
12. A. Hanbury and J. Serra. A 3D-polar Coordinate Colour Representation Suitable for Image Analysis, Technical Report PRIP-TR-77, PRIP, T.U. Wien, 2002.
13. Kantawong, Songkran. Road Traffic Signs Guidance Analysis for Small Navigation Vehicle Control System, 2007 IEEE International Conference on Vehicular Electronics and Safety Beijing, China. December 13 – 15, 2007.
14. S. Varun, Surendra Singh, R. Sanjeev Kunte, R. D. Sudhaker Samuel, and Bindu Philip. A road traffic signal recognition system based on template matching employing tree classifier, Proceedings of the International Conference on Computational Intelligence and Multimedia Applications (ICCIMA), pp.360–365, Washington, DC, USA, 2007.
15. Andrey Vavilin and Kang-Hyun Jo. Automatic Detection and Recognition of Traffic Signs using Geometric Structure Analysis, SICE-ICASE International Joint Conference, Busan, Korea 2006.
16. Miura, T. Kanda, and Y. Shirai. An active vision system for real-time traffic sign recognition, IEEE Conference on Intelligent Transportation Systems (ITS), pp 52–57, Dearborn, MI, 2000.
17. A. Arlicot, B. Soheilian, and N. Paparoditis. Circular road sign extraction from street level images using colour, shape and texture database maps, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, volume 38 (Part3/W4), pp 205–210, Paris, France, 2009.
18. Han Liu, Ding Liu, Jing Xin. Real-Time Recognition Of Road Traffic Sign In Motion Image Based On Genetic Algorithm, Proceedings of the First International Conference on Machine Learning and Cyberneucs, Beijing, vol.1 ,pp 83 - 86, November 2002 .
19. John Hatzidimos. Automatic traffic sign recognition in digital images, Proceedings of the International Conference on Theory and Applications of Mathematics and Informatics - ICTAMI 2004, pp 174–184, 2004.
20. Y.B. Damavandi, K. Mohammadi, Speed Limit Traffic Sign Detection & Recognition, Proceedings of the 2004 IEEE. Conference on Cybernetics an Intelligent Systems Singapore, December 1-3, 2004.
21. A. De la Escalera, L. E. Moreno, M. A. Salichs, J. M. Armigol. Road Traffic Sign Detection and Classification, Industrial Electronics, IEEE Transactions on, Vol. 44, No. 6. pp. 848-859, 1997.

22. Carlos Filipe Paulo, Paulo Lobato Correia. Automatic Detection And Classification Of Traffic Signs. Eight International Workshop on Image Analysis for Multimedia Interactive Services, pp 282-286, 2007.
23. Wen-Yen Wu, Tsung-Cheng Hsieh, and Ching-Sung Lai. Extracting Road Signs using the Color Information. Proceedings Of World Academy Of Science, Engineering And Technology Issue 32, August 2007, Issn: 2070-3724
24. C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofer, T. Koehler. A System for Traffic Sign Detection, Tracking and Recognition using Color, Shape, and Motion Information, IEEE Intelligent Vehicles Symposium, pp. 255-260. June 2005
25. S. Estable, J. Schick, F. Stein, R. Ott, R. Janssen, W. Ritter, Y. Zheng. A Real-Time Traffic Sign Recognition Systems, IEEE Intelligent Vehicles Symposium, pp. 24-26, October, 1994
26. M. A. Souki, L. Boussaid, M. Abid. An Embedded System for Real-Time Traffic Sign Recognizing, IDT'08 Workshop , Monastir, Tunisia, December 20-22, 2008
27. Axel Braun, Oliver Bringmann. Design of an automotive traffic sign recognition system targeting a multi-core SoC implementation, Design, Automation and Test in Europe, DATE 2010, Dresden, Germany, March 8-12, 2010. pp 532-537
28. T. P. Cao, G. Deng. Real Time Vision-based Stop Sign Detection System on FPGA, 978-0-7695-3456-5/08, 2008 IEEE Computer Society
29. Robert Fisher, <http://homepages.inf.ed.ac.uk/rbf/HIPR2/spatdom.htm>. last accessed date: June 3, 2010
30. Wikipedia. http://en.wikipedia.org/wiki/Hough_transform, last accessed date: June 3, 2010.
31. Mohammed Roushdy, Comparative Study of Edge Detection Algorithms Applying on the Grayscale Noisy Image Using Morphological Filter. GVIP Journal, Volume 6, Issue 4, December, 2006
32. The University of Edingburg, School of Informatics Home Page. <http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm>, last accessed date: June 3, 2010

APPENDIX A

ML507 DEVELOPMENT BOARD

In this appendix, general information about ML507 Evaluation Board is given.
The block diagram of the ML507 Evaluation platform is,



UG347_03_110708

Figure A-1 Block Diagram of ML507 Board

The board includes,

- Xilinx Virtex-5 FPGA ML507 (XC5VFX70T-1FFG1136)
- Two Xilinx XCF32P Platform Flash PROMs (32 Mb each) for storing large device configurations
- Xilinx System ACE™ CompactFlash configuration controller with Type I CompactFlash connector
- 64-bit wide, 256-MB DDR2 small outline DIMM (SODIMM), compatible with EDK supported IP and software drivers
- Programmable system clock generator chip
- One open 3.3V clock oscillator socket
- External clocking via SMAs (two differential pairs)
- General purpose DIP switches (8), LEDs (8), pushbuttons, and rotary encoder
- Expansion header with 32 single-ended I/O, 16 LVDS-capable differential pairs,
- 14 spare I/Os shared with buttons and LEDs, power, JTAG chain expansion capability, and I2C bus expansion
- Stereo AC97 audio codec with line-in, line-out, 50-mW headphone, microphone-in jacks, SPDIF digital audio jacks, and piezo audio transducer
- RS-232 serial port, DB9 and header for second serial port
- One 8-Kb I2C EEPROM and other I2C capable devices
- PS/2 mouse and keyboard connectors
- ZBT synchronous SRAM, 9 Mb on 32-bit data bus with four parity bits
- JTAG configuration port for use with Parallel Cable III, Parallel Cable IV, or Platform
- USB download cable
- 5V @ 6A AC adapter

APPENDIX B

GUI of TSR System

The GUI used for sending the image and shape information via Serial Port is developed on Microsoft Visual Studio 2008. It is used as a TSD module. Moreover, the calculation results in each step of the TSR system is displayed in order to debug while implementing the algorithm in FPGA.

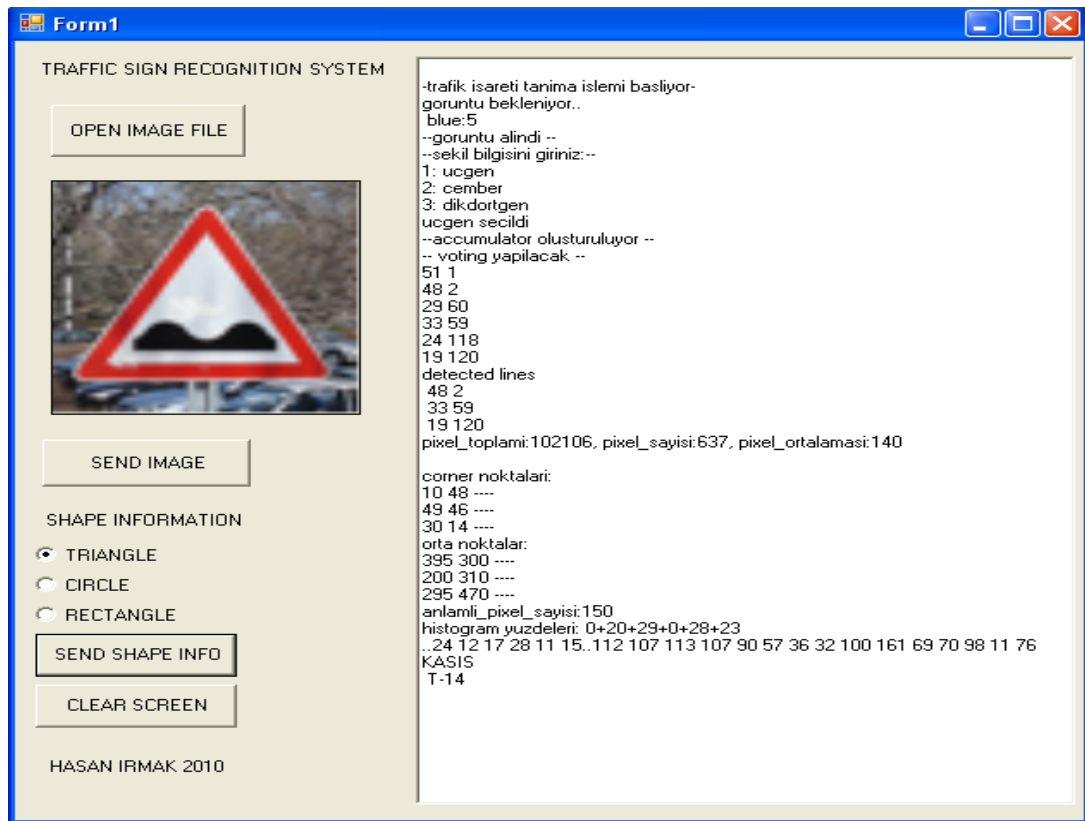


Figure B-1 TSR GUI