Speed Limit Traffic Sign Recognition in Night Mode Based on Deformable Spatial Pyramid

Supervisor: Rubel Biswas

Co-supervisor: Moin Mostakim

Conducted by: Subroto Nag Pinku 11201019

Department of Computer Science and Engineering,

BRAC University

Submitted on: 20th April 2016

Acknowledgement

All thanks to Almighty, the creator and the owner of this universe, the most merciful, beneficent and the most gracious, who provided us guidance, strength and abilities to complete this research.

We are especially thankful to Rubel Biswas and Moin Mostakim, our thesis supervisor and co-supervisor, for their help, guidance and support in completion of our project. We also thankful to the BRAC University Faculty Staffs of the Computer Science and Engineering, who have been a light of guidance for us in the whole study period at BRAC University, particularly in building our base in education and enhancing our knowledge. Finally, we would like to express our sincere gratefulness to our beloved parents, brothers and sisters for their love and care. We are grateful to all of our friends who helped me directly or indirectly to complete our thesis.

Abstract

Traffic signs play a vital role in transportation system. Many road accidents occur due to over speed. Detecting and classifying different group of Speed limit traffic signs can save our lives as well as resources. In this research we propose a novel approach towards the detection of these signs. In our proposed system with the help of color and non-color information of traffic signs we first detect the presence of a sign and the classify it. For detection we used circle Hough transformation along with segmentation and labeling. After extracting sign from a scene we match the sign against a classified dataset. We used deformable spatial pyramid matching for recognition of the sign. Once we find a match, our system returns the class or speed limit. Our experiment shows that the recognition rate is very high and we compare our result with another approach at the end.

Contents

Declaration 2

Acknowledgement 3

Abstract 4

Contents 5

List of Figures 8

List of Tables 9

Chapter 1 : Introduction 10

1.1 Introduction 10

1.2 Motivation 10

1.3 Objective 11

Chapter 2 : Problem Formulation 12

2.1 Problem Formulation 12

2.2 Related Work 12

2.3 Possible Approaches 13

Chapter 3: Background 15

3.1 Speed Limit Traffic Sign 15

3.2 Night Vision 16

3.2.1 Low Light Images 16

3.3 Retinex Theory 16

3.3.1 Single Scale Retinex (SSR) 17

3.3.2 Multi-Scale Retinex with Color Restoration (MSRCR) 17
3.4 Image Segmentation 19
3.5 Edge Detection 20
3.5.1 Canny Edge Detector 20
3.6 Connected Component Labeling Algorithm 22
3.7 Hough Transformation 23
3.7.1 Circle Hough Transformation 23
3.8 OTSU Method 24
3.9 Deformable Spatial Pyramid (DSP) Model 25
3.9.1 The Pyramid 25
3.9.2 Matching Objective 25
3.9.2.1 Scale Invariant Feature Transform (SIFT) Detector and Descriptor 28
3.9.2.2 Belief propagation 33

Chapter 4 : Proposed System 35
4.1 Overview of System Design 35
4.2 Sign Extraction 36
4.2.1 Application of MSRCR 36
4.2.2 Image Segmentation 37
4.2.3 Edge Detection and Connected Components Labeling 37
4.2.4 Detection of Circle objects using CHT 38
4.2.5 Crop Sign and Convert to Binary Bmage 39
4.3 Recognition of Speed Limit Class 39

Chapter 5 : Experiment & Result Analysis 41

5.1 Dataset 41

5.1.1 Data Collection 41

5.1.2 Data Analysis 41

5.2 Experiment 42

5.2.1 Tools (Environment) 42

5.2.2 Implementation 42

5.3 Result Analysis 42

Chapter 6 : Conclusion 48

6.1 Concluding Remarks 48

6.2 Limitations 48

6.3 Future Work 48

References 50

List of Figures

01	Figure-3.1.1: Speed limit traffic signs	15
02	Figure-3.9.2.1: The Graph Structure of DSP	27
03	Figure-3.9.2.2: Sketch of DSP method	27
04	Figure-3.9.2.1.1: Scale space and DoG Pyramid	29
05	Figure-3.9.2.1.2: Comparison with Neighbors	30
06	Figure-3.9.2.1.3: SIFT descriptor from gradients	32
07	Figure-3.9.2.2.1: Markov Random Field, Bayes Net, Factor Graph	33
08	Figure-3.9.2.2.2: Converting a Factor Graph into a MRF	34
09	Figur-3.9.2.2.3: Loopy Belief Propagation	34
10	Figure-4.1.1: Block Diagram of the Proposed System	35
11	Figure-4.2.1.1: Before and After Applying MSRCR	36
12	Figure-4.2.2.1: Segmented Image	37
13	Figure-4.2.3.1: Detection of Edges and Image After Labeling	38
14	Figure 4.2.5.1: Bounded region (s) to crop in the image	39
15	Figure-4.3.1: Test Image and Image Returned by DSP	40
16	Figure-4.3.2: Illustration of Our Proposed System	40
17	Figure-5.3.1: Comparison Chart of DSP and SVM	46
18	Figure-5.3.2: Results Returned by Our System	47

List of Tables

01 Table-1: Results Obtained via DSP 43

02 Table-2: Results Obtained via SVM 44

03 Table-3: Comparison between SVM and DSP 45

Chapter 1: Introduction

We have come up with the goal of making a system which can recognize speed limit traffic signs. In this chapter the motivation behind our research and objective will be discussed.

1.1 Introduction

Modern technology has reached at a level where we cannot imagine our life without them. Throughout past century the development occurred. Compared to many other fields, the history of Computer Vision is not very old. In the scope of Computer Vision, along with many other tasks, the tasks of recognition played one of the major roles. As the computation evolved, the journey takes us to a new era of automation. We can now practically use automated car! Even in manual system i.e., car driven by human we have automated system which helps the driver in many extent. One of the tasks is automated speed controlling system.

1.2 Motivation

When we take a look at our transportation system what we see is, many dangerous road accidents which causes severe damage to human life and resources. We analyzed the reasons behind this and figured out that over speed is one of the major reasons along with many other infrastructural lacking. That is where we look forward to minimize the occurrence by controlling the speed automatically. Driver Support System (DSS) provide real time support to detect traffic signs. With this information provided to the driver we can save lives, resources and protect area from pollution as well.

1.3 Objective

The use of traffic sign recognition system is very challenging. Since rain, fog, snow etc. affect the whole system. Another thing is the light variation i.e., shadows, sun, clouds etc. The geometrical shape of the object and the perspective is also a big concern. So we must be able to come up with a system which can work under light variation and geometrical transformation of the objects in a scene. If we narrow down the scope we are particularly interested in detection of these signs under low light condition. In short, our objective is to make the system very accurate and efficient.

Chapter 2: Problem Formulation

This chapter describe other related research projects which have been done in past. Here we analyze their work briefly and look up for further direction to our research.

2.1 Problem Formulation

Formulating a problem is the first thing needed to solve it. When we look forward to develop a system we must know the problem precisely and then analyze the related work. In our task of recognition system, the problem is very complex. Firstly, we are doing it for low light images. That is for night mode images. From these kind of images we will have an input. Secondly, we want to extract sign from this. Thirdly, we want to detect the sign to use in real time without using any training approach. This is how we defined our problem in short.

2.2 Related Work

After the problem is defined, we have reviewed past projects, research papers. In the process we analyzed them and tried to find the limitations of these approaches.

Traffic sign detection is a rapidly growing field. The future vehicles will be in real need of such systems. Recognition rate, real-time implementation, categorizing the objects and night mode adaption ability, are the performance indexes.

An approach for detection of Norwegian speed limit signs was proposed by Torresen et al [1] which consists of three major parts. In the paper they first applied color based filtering to extracts the colors and reduce the color red among them which makes the matching algorithm more convenient. Then they located the sign in the image by template matching. According to the paper, they used the template matching which is efficient to implement in the hardware. The

last phase is detecting the numbers. For which they used their algorithm along with a classifier system. The accuracy of that system was 91% based on the experiment using 198 images. Though the system works quite well. The template matching requires hardware level implementation. By this time, a more general approach is needed.

Hoferlin and Zimmermann [2] developed a system by using local features of an image. They used Scale-Invariant Feature Transform (SIFT)[5] for local features. Their sign detection approach was content-based along with shape-based approaches. This is pretty good in the sense that they used SIFT feature which leads the way to invariant feature which later can be matched.

Another method was proposed by Liu and Maruya [3] where one approach was adaptive procedure for image processing whereas the other was more concerned about controlling the exposure automatically on the onboard cameras in vehicle.

Wu and Tsai [4] used video imaging for detecting signs which supports real-time road inventory data collection. Based on locally adaptive thresholding technique color segmentation was done .In the next phase the detection and extraction of the sign had done by means of OCR (Optical Character Recognition) and 2D correlation .In their work the recognition rate was 97%.

Aryuanto and Koichi [6] have proposed a new technique which uses geometric fragmentation to detect the circle sign in red color. They combined both fragments of right and left to detect the outer ellipse. Their method results into high recognition rate as well as lower cost.

2.3 Possible Approaches

Designing the system itself is a bigger challenge. So many tasks have been done. Various methods have been proposed. Some of them are more accurate and the others' are cost effective. We had to keep all these situations in head. We were looking for a design which is a balanced

combination of real-time implementation and accuracy. What we needed is, an algorithm which effectively extracts sign and matches any input image with respect to our database. We exclude any kind of learning approach. We have seen that many systems have designed special hardware. So, we could choose that approach as well but any special hardware design is also kept out of the whole designing process. Hardware support is assumed to be common for all vehicles. To keep it simple, our system is pretty straightforward and of course we tried to make it efficient. The whole system is described in Chapter-4.

Chapter 3: Background

In this chapter, the theoretical background which we have used to design our system will be discussed. We have tried to make it as precise as possible. Since many of the concepts described here relies on many other concepts. We shall not be going into the very details. We assume the reader has the basics of most of the things here.

3.1 Speed Limit Traffic Sign

Traffic signs play an important role in modern day transportation system. These signs lead the way to control speed of various vehicles as well as prohibiting few actions of them. In various part of the world they are based on different rules and regulations but they appear in similar shape across many countries. These signs are circle shape mostly, with a red circle rim and the interior consists of blue or white color. Yellow color is used in some countries for the interior i.e., Sweden. Prohibitory signs are grouped into few groups according to their action. For example, No U-turn, No entry, Speed limits, No overtaking, No parking and Wrong way etc. are the prohibitory sign group. Among all these groups speed limit signs are used to regulate vehicles' speed particularly. They represent maximum speed allowed in Sweden which is written in Black. Figure-3.1.1 shows speed limit traffic signs in Sweden.

Figure-3.1.1: Speed Limit Traffic Signs

3.2 Night Vision

The ability to see in low light is known as night vision. Generally the use of night vision was in the army but in the later part of the nineteenth century several devices had been developed as technology evolved. In the field of computer vision low light images are one of the few concerns for recognition.

3.2.1 Low Light Images

Due to lack of light the same object become unidentifiable which could be identifiable in a better lighting condition. Hence it becomes very important to enhance the image (or a frame of a video) to process further.

3.3 Retinex Theory

Retinex is a method which bridges the gap between human perception and images. The word retinex is a combination of Retina + Cortex. That is the human visual system is in the core of the theory. Color constancy is a feature which is defined as that the color of objects remains almost similar in illuminative variation. The theory was first described by Edwin H. Land in 1971. This suggests us that the eye and the human brain both are involved in the processing. The human eye though do the color compression and color constancy without any mentionable effort but for computer vision Based on this concepts we can define Single-Scale Retinex(SSR) and Multi-Scale Retinex(MSR). Color Restoration can be applied later with MSR.

3.3.1 Single Scale Retinex (SSR)

The single scale retinex (SSR) is given by,

$$R_i(x,y) = logI_i(x,y) - \log\left[F(x,y) * I_i(x,y)\right]$$

The retinex output is defined as, $R_i(x,y)$and The image distribution in the *i-th* color band is defined as $I_i(x,y)$. Here, '*' denotes convolution operation and $F(x,y)$ is called the surround function (Gaussian function). The Gaussian function is defined as,

$$F(x,y) = K(\exp\left(-\frac{x^2+y^2}{c^2}\right))$$

Where, the scale *c* , which is also called the Gaussian surrounding constant and *K* is found by taking double integral of Gaussian function. Such that,

$$\iint F(x,y)dxdy = 1$$

3.3.2 Multi-Scale Retinex with Color Restoration (MSRCR)

If we take weighted sum of several SSR with different scales we get the output as multi scale retinex (MSR).

The *i*-th color component of the MSR output is defined as, $R_{M_i}(x,y) = \sum_{n=1}^{N} w_n R_{n_i}(x,y)$

Here, the number of scale is *N,* the *i*-th component of the *n*-th scale is termed as $R_{n_i}(x,y)$, the weight associated *n*th scale is defined as sw_n.

According to the application the number of scales is chosen. The empirical values are, N=3, $w_n = 1/3$ and C=15, 80 and 250 corresponding to each scale respectively.

Since MSR is better for gray images than color images. Another concept is introduced. It is called color restoration (CR). MSR along with CR solves the problem.

MSRCR is given by, $R_{MC_i}(x,y) = C_i(x,y)R_{M_i}(x,y)$

Where, $R_{MC_i}(x,y)$ is the MSRCR output, $R_{M_i}(x,y)$ is the MSR output and $C_i(x,y) = f[I_i'(x,y)]$. Where, $f[I_i'(x,y)]$ is the *i*th band color restoration function. This is given by, $I_i'(x,y) = I_i(x,y) / \sum_{i=1}^{S} I_i(x,y)$. Here, *S* is the number of spectral channels. Generally, S=3. Experiment shows that the function which gives the best overall color restoration is, $C_i(x,y) = \beta \log[\alpha I_i'(x,y)]$.

Putting the value of $I_i'(x,y)$ we get, $C_i(x,y) = \beta\{[\log[\alpha I_i(x,y)] - [\log[\alpha I_i(x,y)]$ where β is a gain constant and α is the nonlinearity controller. Empirically the values of $\alpha = 125$ and $\beta = 46$. Using canonical gain/offset we get the final MSRCR. That is,

$$R_{MC_i}(x,y) = G[C_i(x,y)\{logI_i(x,y) - \log[F(x,y) * I_i(x,y)]\} + b]$$

Where, G and b are the canonical constants. The experimental values are, G=192 and b=-30.

3.4 Image Segmentation

Image segmentation is defined by partitioning an image into multiple homogenous regions based on certain characteristics. The representation involve more meaningful image than the original so that we can use it for further processing.

Locating and object, finding boundaries etc. are the tasks of image segmentation. The segmented images are nothing but the group of pixels having similar intensity values which can be represented by other colors that makes sense.

A faster and easier technique [9] for segmentation is described here. Let, R, G, B be the RGB components of the scene and G be the green channel of the scene. Let's say, S is the new segmented image where,

$$S=R-G$$

Using this equation almost all unnecessary objects can be removed and the segmented image and keeps all important objects intact

3.5 Edge Detection

In image processing edge detection is used to reduce data in an image while preserving outline of all the objects. There are several methods for detecting edge. Canny edge detector which is e developed by John F. Canny, is one of them.

3.5.1 Canny Edge detector

The summary canny edge detection algorithm can be written in five steps [7].

1. Noise removal using Gaussian filter.
2. Gradient at each pixel are determined using Sobel-operator
3. Suppression of the image makes blur edges to be sharp
4. Double thresholding is applied to make the edge detection better
5. Edges are tracked by hysteresis

These steps are briefly described below-

Every image has noise. To remove noise we do smoothing. The equation which describes this,

$$g(m,n) = G_\sigma\,(m,n) * f(m,n)$$

In the above equation *m* and *n* are the pixel coordinates, *f* is the given image and *g* is the image after smoothing. Here, the Gaussian function G_σ is defined as,

$$G_\sigma = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{m^2+n^2}{2\sigma^2}\right)$$

In the second step the gradient of $g(m,n)$ is found by any gradient operator ie, Sobel operator. Where

Magnitude, $M(m,n) = \sqrt{g_m^2(m,n) + g_n^2(m,n)}$ and

Angle, $\theta = tan^{-1}(\frac{g_m(m,n)}{g_n(m,n)})$

Threshold value (T) is chosen to suppress the noise in the image while preserving the edges. Where threshold $M_T(m,n) = \begin{cases} M(m,n) \ \ if \ M(M,n) > T \\ 0 \ otherwise \end{cases}$

The idea is, suppress non maxima pixels in the edges. To do that, we compared each non zero pixel with threshold value T if that is greater than threshold value, keep it unchanged otherwise set it to zero. After this step we again threshold this resulting image with two threshold values $T_1$and T_2 , where $T_1<T_2$.

We get two binary images from this and we link edges in the first image. At the last stage of these, link all edges in the second image, search neighbor edges from first to second image to make the gap less until any other edge encountered.

3.6 Connected Component Labeling Algorithm

Connected components labeling algorithm scans through an entire image and look up for pixel connectivity. When it finds connectivity among pixels it groups them together. The algorithm uses pixel intensity as the main criteria. The idea is connected pixels share similar intensity values. This algorithm works on binary or gray level images. Among different measures we choose 8-connectivity. For binary image a pixel has the value 0 or 1.The main steps of the algorithm go as follows,

1. *Scan the image(V) from top to bottom and left to right*
2. *For each pixel(p),*

If all four neighbors has the value 0

then assign new level

else if a single neighbor has 1,

assign it's label to p

else if multiple neighbor has 1

assign one of the labels to p and make an equivalence note

Once the scan is done, equivalence class is sorted and the pixels in equivalent class are assigned unique level. Then second pass is done. In which same label for same class is ensured. For display different colors or gray levels can be used.

3.7 Hough Transformation

Hough Transformation is a technique to isolate feature or finding shapes in an image. Hough transformation is mostly used for detecting lines, ellipse etc. Here we are going to describe circle detection using Hough Transformation.

3.7.1 Circle Hough Transformation

Circle Hough transformation [14] can be defined as a transformation of 2D plane to the parameter space. That is the center of to be transformed into a parameter space.

The equation for circle is given by,

$$(x - a)^2 + (y - b)^2 = r^2$$

Where *a*, *b* are the center of the circle and *r* is the radius.

The parametric representation is given by,

$$x = r \cos(\theta)$$

$$y = r \sin(\theta)$$

To determine if there is a circle we accumulate votes in a three dimensional parameter space (a, b, r). Our objective is to find the center coordinates. The idea is, true center point is common to all s in parameter space which can be found by accumulator array. Hough transformation uses a voting mechanism which defines how many votes are given to any object in the images (location, radius etc.). The circle is found by pick generation from voting mechanism. The presence of a pick determines the object.

3.8 OTSU Method

Otsu method is a method named after Nobuyuki Otsu is a cluster based thresholding method [ref]. This method extensively searches for the threshold which reduces intra-class variation.

This method calculates threshold for every pixels and iterate thorough all values neighboring a particular threshold.

It takes probabilistic approach to determine the ultimate thresholds. Using it, we can take any number of threshold values. In turn we can use those values for multilevel threshold as well.

3.9 Deformable Spatial Pyramid (DSP) Model

DSP model defines a pyramid and an objective function. Based on which it optimization is done.

3.9.1 The Pyramid

An image is divided into four rectangular grids. Each of the grids again divided into another four. This division is done till we get desired number of level. We used 3 levels as the author had used in their work. Along with these 3 layers another layer is added, the pixel layer. Then combining all these we represent them as a graph [Figure-3.9.1.1]. In the graph the size and spatial context of the nodes vary. The edges of the graph span all levels and all nodes. Each grid cell is also nodes. They are also connected with each other. Apart from the pixel layer, where the pixels are not connected to each other. Rather they are connected to their parent grid cell directly. This reduces the computation cost. A greater node assures regularization where smaller nodes are for smoothness.

3.9.2 Matching Objective

In DSP the SIFT descriptor is used for features. Though, it can be done using any other descriptor. i.e., HOG, GIST etc. We prefer to stay with SIFT. The target of DSP is to find the best translation from first image to second image. To achieve this goal, an energy function is modeled. The equation for energy function is given below.

$$E(t,s) = \sum_{i} D_i(t_i, s_i) + \alpha \sum_{i,j \in \mathrm{N}} V_i(t_i, t_j) + \beta \sum_{i,j \in \mathrm{N}} W_{ij}(s_i, s_j)$$

In the equation for any node *i* in the first image, $t_i$is the translation node in the second image. Here, D_i is the data term. The data term measures average distance between descriptor in two images. This is also called the appearance matching cost for node *i* at translationt_i. The data term is a multi-variable function and defined as,

$$D(t_i, S_i) = \frac{1}{z}\sum_q \min\left(\left|d_1(q) - d_2\left(S_i(q + t_i)\right)\right|\right)_1, \lambda)$$

This equation consists of d_1 and d_2 which are the descriptors for coordinate q and $q + \mathrm{t}_i$ within a node *i,* the total number descriptor is termed as *z.* For each descriptor d_1, we find another corresponding descriptor d_2. Followed by scaling factor S_i, a translation $t_i$determines the second descriptor in the other image. Optimal scale is taken for each node which influences best match for multi scale matching. In the first equation smoothness term is defined by$V_{ij} = \min\left|\left(t_i - t_j\right), \lambda\right|$, it penalizes large discrepancies and regularizes the solution. Truncated L1 norm is used and threshold value λ is used empirically. The scale smoothness term $W_{ij} = \left|s_i - s_j\right|_1$, where s_i is the scale variable. Here two constant weights α and β are associated with these two terms respectively.

DSPMatch (source image, target image) *returns energy-score*

1. Find descriptor(s) i.e., SIFT

2. Define deformable spatial pyramid

3. Define energy function

5. Minimize the function using Belief Propagation

6. Return energy-score

Different energy scores for different images are returned. We take the minimum of all scores. Minimum score defines a better match. For example, two identical images will result in zero energy score here.

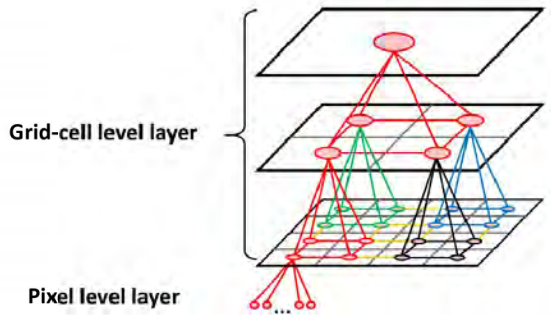


Figure-3.9.2.1: The Graph Structure of DSP [16]

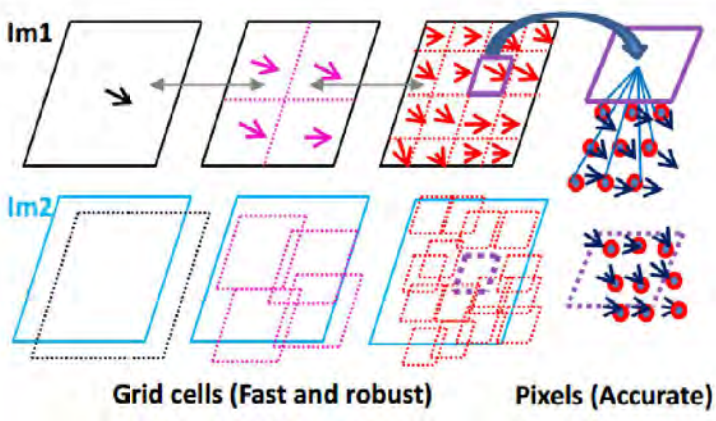


Figure-3.9.2.2: Sketch of DSP method. (First and second rows show source and target images respectively.) [16]

3.9.2.1 Scale Invariant Feature Transform (SIFT) Detector and Descriptor

The idea of SIFT is to convert image content to features which are scale invariant. That is, different size, view point, illuminations, scales etc. doe not affect the descriptor. Once we can detect such features, we can describe them with a descriptor. Then that descriptor can be used against a database to find a match.

The steps of finding SIFT features are given bellow.

Step-1:

Construction of scale space:

To create scale space Gaussian kernel is used. If we convolve a Gaussian kernel with an image we get the output image on that scale.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

Here I is the given image and L is the output image.

The Gaussian function is given by,

$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

Step-2:

Take Difference of Gaussians:

To build scale space pyramid laplacian of Gaussian can be taken. Since, the computational cost is very high; Difference of Gaussian (DoG) is taken as a close approximation. We need all these to detect extrema.

$$\sigma\nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

This implies,

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2\nabla^2 G$$

So we get,

$$D(x, y, \sigma) = \big(G(x, y, k\sigma) - G(x, y, \sigma)\big) * I(x, y)$$

$$= L(x, y, k\sigma) - L(x, y, \sigma)$$

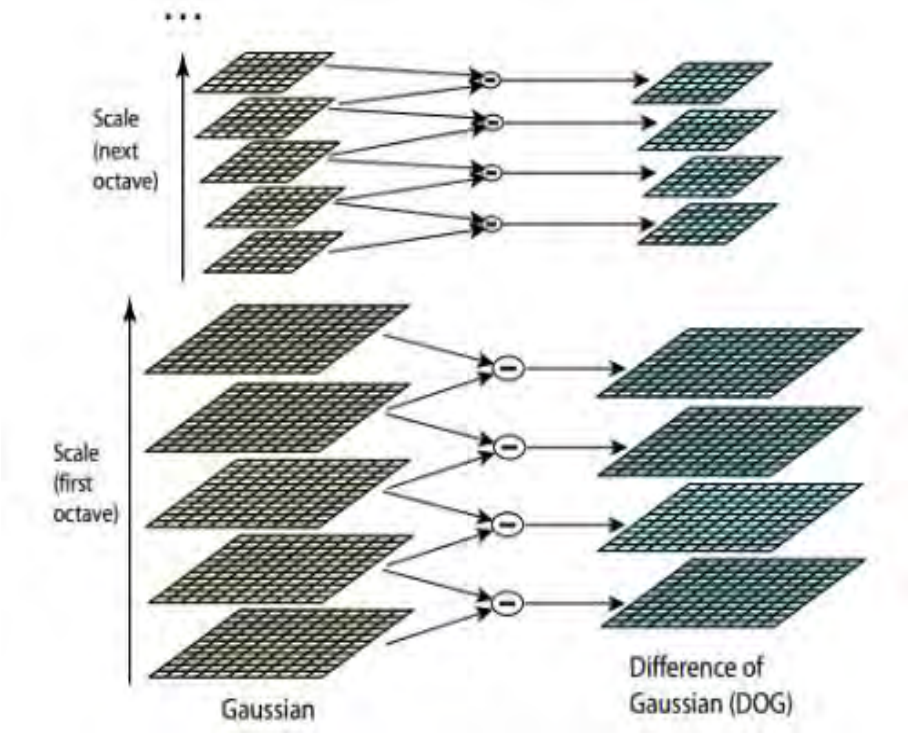


Figure-3.9.2.1.1: Scale Space and DoG Pyramid [5]

Step-3:

Locating the extrema of DoG:

To find extrema we look scan through each DoG image. By looking at each neighboring pixel including scale we determine min and max from them. For this we need 26 comparison illustrated in figure- Figure-3.9.2.1.2.

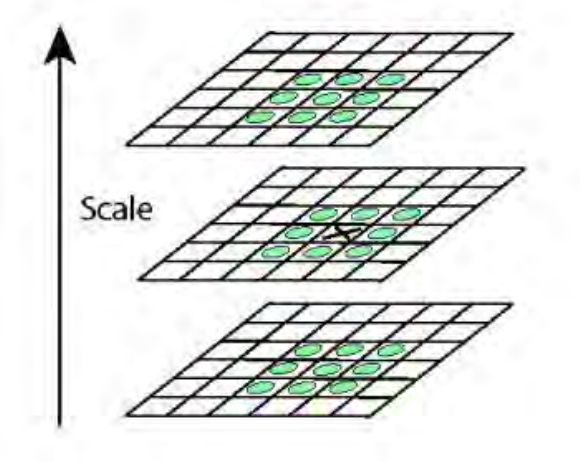


Figure-3.9.2.1.2: Comparison with Neighbors [5]

Step-4:

Sub pixel localization:

The Taylor series expansion is given by,

$$D(\boldsymbol{x}) = D + \frac{\partial D^T}{\partial \boldsymbol{x}} + \frac{1}{2}\boldsymbol{x}^T \frac{\partial^2 D}{\partial \boldsymbol{x}^2}\boldsymbol{x}$$

We differentiate and set it to 0 to get location is terms of (x, y, σ)

Here, $\hat{x} = \frac{\partial^2 D^{-1}}{\partial x^2} + \frac{\partial D^{-1}}{\partial x}$

Step-5:

Filter Low contrast points:

For this we use scale space value found in previous location.

$$D(\hat{x}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \boldsymbol{x}}\hat{x}$$

Step-6:

Edge response elimination:

We know a pick has high response along edge but low response in any other direction. We can use hessian to eliminate this response. It says that Eigen values are proportional to curvatures. We use, trace and determinant also.

Here,

$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta,$

$Det(H) = D_{xx}D_{yy} - \left(D_{xy}\right)^2 = \alpha\beta,\ \frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}$

Step-7:

Assign Orientation:

To assign orientation, we compute gradient for each blur image.

Magnitude, $m(x,y) = \sqrt{\left(L(x+1,y) - L(x-1,y)\right)^2 + \left(L(x,y+1) - L(x,y-1)\right)^2}$

Angle, $\theta(x,y) = tan^{-1}(\left(L(x,y+1) - L(x,y-1)\right)/\left(L(x+1,y) - L(x-1,y)\right))$

We create a histogram for region around a key point. The histogram consists of 36 bins for orientation. We weight each point with a Gaussian window of 1.5σ. Then we create key point with value >= .8 max bin.

Step-8:

Building a descriptor:

We find the blurred image of the closest scale. Then sample point around key points. The next phase we rotate the gradients and coordinates by previously computed orientation. At the end, we separate the region into sub region. Lastly, creation of histogram for each sub region is needed.

Figure-3.9.2.1.3 shows the process. We take 4x4 descriptors from 16x16 that results into a 4x4x8=128 element vector.

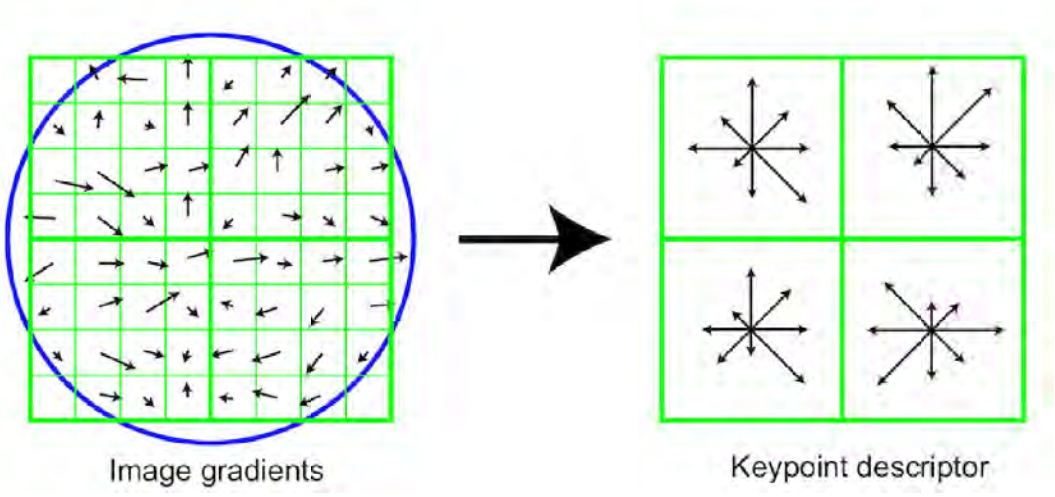


Figure-3.9.2.1.3: SIFT Descriptor from Gradients [5]

3.9.2.2 Belief propagation

Belief propagation [11] is an algorithm which is used to find marginal distribution for each unobserved node. This algorithm is also known as message passing algorithm. It tends to approximate or update belief of a node. Inference problems are NP- hard. BP approximates closely most of the times.

Suppose, we have hidden variables Y, observed variables X and some model regarding P(X|Y) and we want to make some analysis of P(X|Y). Often we can see that, $P(X,Y) = \prod_k \emptyset_k$ where, $X_{C_k} \subseteq X \cup Y$

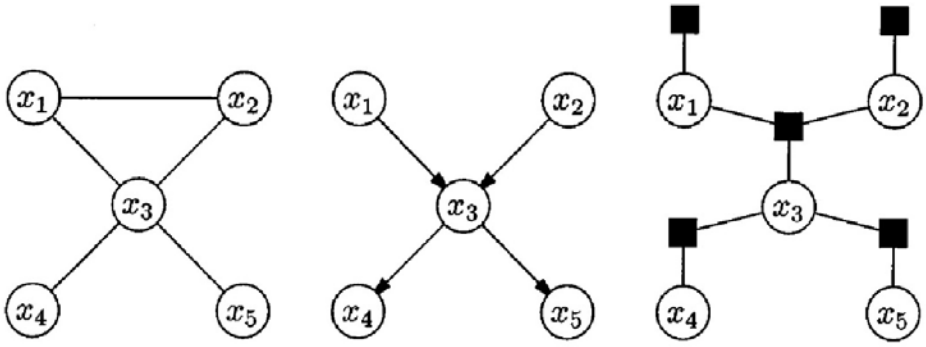


Figure-3.9.2.2.1: Markov Random Field, Bayes Net, Factor Graph [13]

The above figure-3.9.2.2.1 shows different representations of same graph. Using any of this we can implement BP. For MRF we know, $P(X) = f_1(x_1,x_2,x_3)\, f_2(x_3,x_4)\, f_3(x_3,x_5) / Z$, for Bayes net, $P(X) = P(x_3|x_1,x_2)\, P(x_4|x_3)\, P(x_5|x_3)$ and lastly for factor graph, $P(X) = f_1(x_1,x_2,x_3) f_2(x_3,x_4) f_3(x_3,x_5)\, f_4(x_1)\, f_5(x_2) / Z$.[12]

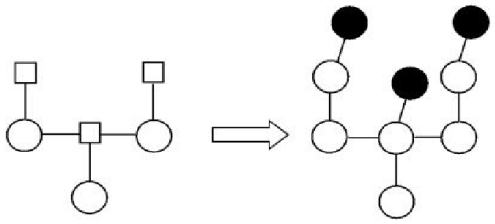
The sum-product algorithm which is also known as belief update can compute marginal distribution very quickly.

By using *message passing* we can compute marginal $P(X_n)$. Message from node *n* to node factor *m* is defined as, $V_{n.m}(X_n) = \prod_{i \in N(n) \setminus n} \mu_{i.n}(X_n)$ and from factor node *m* to variable

node n message is defined as, $\mu_{m.n}(X_n) = \sum_{X_{N(n)\backslash n}}\left[f_s\left(X_{N(s)}\right)\prod_{i\in N(n)\backslash n} V_{i.m}(X_i)\right]$. Now we can write Marginal distribution $P(X_n) \propto \prod_{m\in N(n)} \mu_{m.n}(X_n)$

A node pass message to another node if and only if it has received message from all other nodes except the receiver node. Here the intuition is, from a node n to node m if message is passed we get, $P(X_m|S_n)$. Where, S_n is the set of all children of node n.

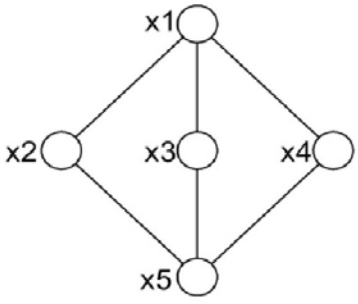
To work with BP we can convert factor graph to MRF [12] as well. Any of the models would work and depends on the user. The figure bellow shows a conversion.



.

Figure-3.9.2.2.2: Converting a Factor Graph into a MRF [13]

Loopy belief propagation is when we apply BP even if there is a loop. In this case, each node sends messages in parallel. Figure-3.9.2.3.2 shows such illustrations.



Figur-3.9.2.2.3: Loopy Belief Propagation [13]

Chapter 4: Proposed System

In this chapter our proposed system is described. We will first give the overview and then details of the system.

4.1 Overview of System Design

In our proposed system, speed limit signs are detected in three major steps. Firstly, we apply MSRCR to get a brighter image and detect the presence of a speed limit sign in the image. Secondly, if one or more sign is found in the image, interior of the image is extracted. Lastly, our system recognizes the number on the sign. For recognition we use a database which was created using many images of the extracted signs and grouped together according to their constituent number. In short, our system take any image, extract sign if exists and then return match result against this database. Figure-4.1.1, illustrates our approach.

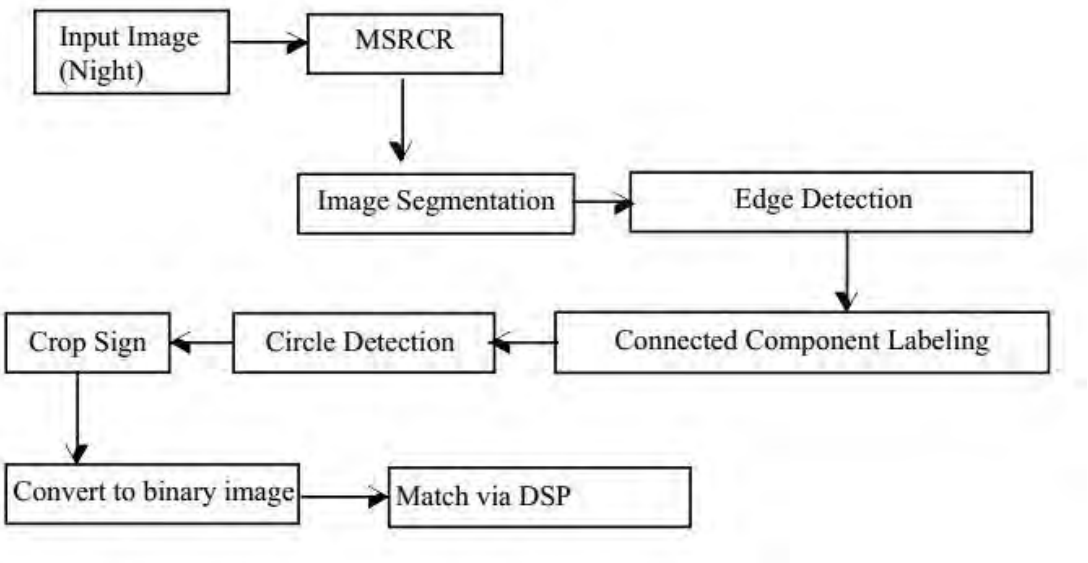


Figure-4.1.1: Block Diagram of the Proposed System

4.2 Sign Extraction

To extract speed limit sign from an image we go through several steps. First of all, applying MSRCR, the given image is segmented. Then the edges of the objects in the segmented image are detected. After that, connected components labeling algorithm is used for identification of the present objects. Finally using circle Hough transformation the sign is detected.

4.2.1 Application of MSRCR

For a dark image, it is difficult to detect presence of many objects in the scene. To overcome this in our system we apply MSRCR to get a brighter image which can be more effective and accurate to use. The MSRCR has always been effective with high probability for dark images. Figure-4.2.1.1 shows the changes of a scene before and after applying MSRCR.

Figure-4.2.1.1: Before and After Applying MSRCR [9]

4.2.2 Image Segmentation

To segment an image we had many options but we choose to keep it as simple as it can be. In our system no color segmentation is used to reduce the search space. Rather we used which are necessary to detect a sign. We simply used the segmentation method what described in section-3.4. Figure-12 shows the image after segmentation. The image is from previous step having speed limit 50.

Figure-4.2.2.1: Segmented Image

4.2.3 Edge Detection and Connected Components Labeling

After the segmentation the edges are detected. For this we used Canny Edge Detector. It is a multi step algorithm. We described the process in section-3.5.1. We used the segmented image to find its edges. Figure-4.2.3.1 shows the resulted image after edge detection. Once the edges are detected we apply connected components labeling algorithm. We apply this to the image we got in last step to find all possible objects in our input image. The reason of this step is to remove any unconnected edge in the image.

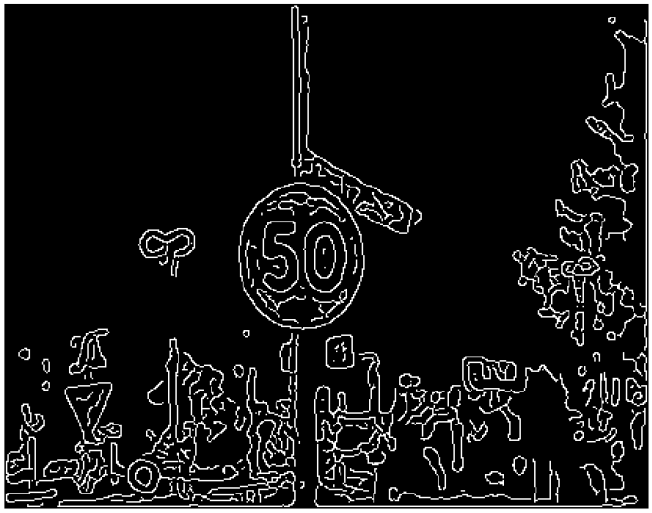


Figure-4.2.3.1: Detection of edges and Image after Labeling

4.2.4 Detection of Circle objects using CHT

The last step of sign extraction is done by invoking Circle Hough Transformation. Using this we get the circle area of the image containing the number. In a scene any speed limit sign contains two circle shape objects. One is the outer circle of traffic sign and another is the number '0' (zero) inside the sign. Based on this principle we approach. We described CHT in section-3.7.1. Generalized HT takes usually longer. That's why we used CHT. Figure-4.2.5.1 shows detected s in an image.

Figure 4.2.4.1: Bounded Region (s) to Crop in the Image

4.2.5 Crop Sign and Convert to binary image

Once we detect the speed limit sign in an image, we now have the centre and radius of the circle we are concerned about. Now from the image after MSRCR we crop the bounding box that binds the circle. Then we apply Otsu's threshold method on our cropped image get the binary image and resize it to 50x50pixels image. Figure-4.2.6.1 shows few of the extracted signs from different images.

Figure-4.2.5.1: Binary image of extracted sign

4.3 Recognition of Speed Limit Class

We used Deformable Spatial Pyramid (DSP) matching algorithm for recognition. This method is very fast and accurate in case of dense matching.

Resized (50x50) image from previous step is taken for this step. We match our test image's sign with a database which is created using such resized binary images of signs. The steps in matching is given below,

1. Take an image (source, from previous step)

2. for each target image in database,

2.1 Energy-scores ← DSPMatch(source image, target image)

3. Match ← min(Energy-scores)

4. Show Class (Match)

The figure bellow shows image from this step.

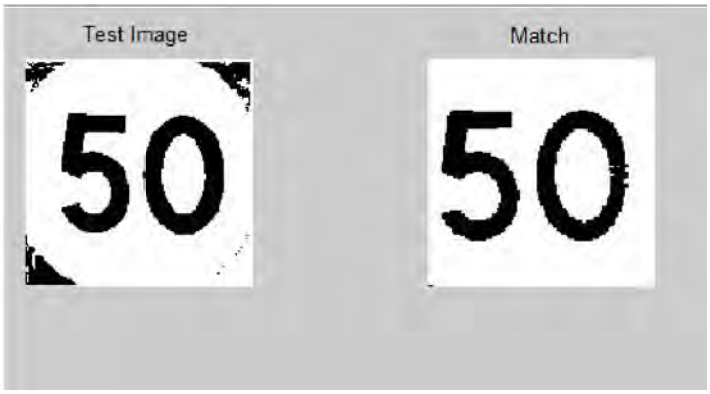


Figure-4.3.1: Test Image and Image Returned by DSP

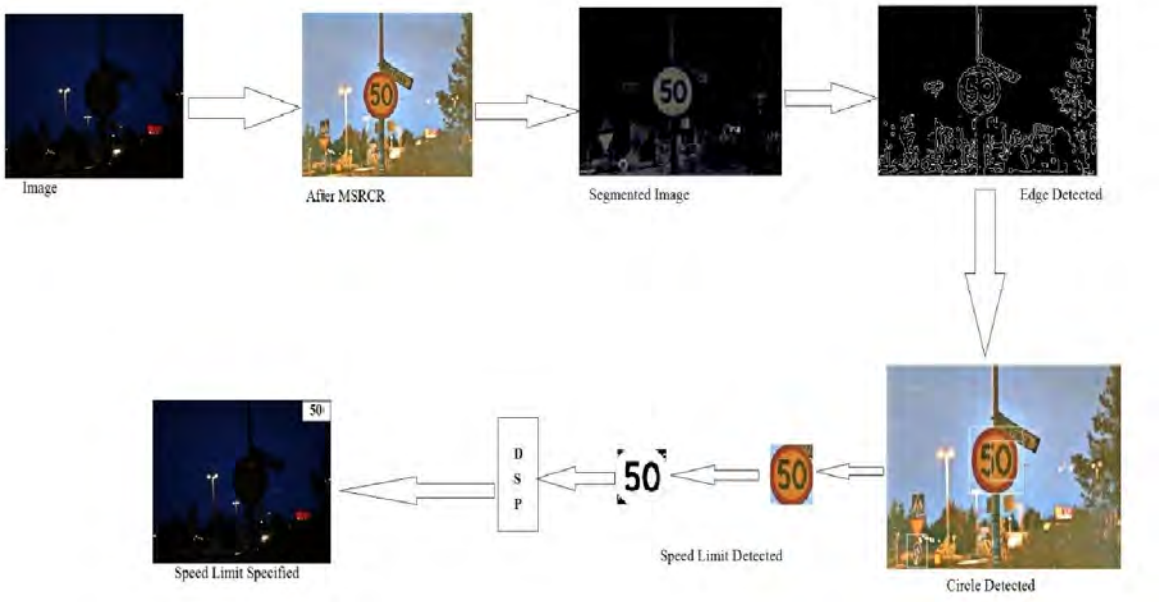


Figure-4.3.2: Illustration of Our Proposed System

The above figure illustrates our system.

Chapter 5: Experiment & Result Analysis

In this chapter we will discuss our experimental results. We will try to illustrate our idea.

5.1 Dataset

To validate an idea we must compare it with the state of the art of that topic. Since we are proposing a system to recognize speed limit traffic sign, we should do experiment with any particular standard data set. We need to be sure that previously the data set was tested and analyzed.

In our work we used road signs in Sweden. We have found that previously few works has been done using these data. That is the reason we have chosen this dataset, so that we can validate our idea.

5.1.1 Data Collection

We have taken the images from Dr. Hasan Fleyh'sdataset [8]. The data set was used against in many experiment and many papers have been published based on those experimental results. That is the reason we had chosen to work with it.

5.1.2 Data Analysis

Once we decided to work with a particular dataset, we analyzed it very carefully if it can fulfill our requirements for proposed system. We found that there are many types of road signs in the dataset. i.e., Speed Limit, Sunny Condition, Blurred, Low light etc. These are the categories in the dataset. We took random images from various classes.

5.2 Experiment

For the experiment we had first set up an environment to work with and then implemented our proposed system.

5.2.1 Tools (Environment)

Our experiment was done on a computer of following configuration,

Processor: Intel(R) Core (TM) i3 CPU M 370 @2.40GHz 2.39GHz

RAM: installed 4 GB memory (3.8GB usable)

System type: 64 bit OS, x64 based processor

5.2.2 Implementation

Our system implementation was done by combining considerable amount of codes, library etc. Since SIFT has a patent issue. We used the authors open source code for that. For DSP, we used the authors provided code [19]. Apart from this, we wrote a system which can match images against a database of images. The database was consist of several sign classes, against which we will match each input and tell the user which class it falls.

5.3 Result Analysis

In the experiment we took 200 test images which contained 210 speed limit signs and other 270 non-speed limit signs. Our method shows an accuracy of 98.54%, which is a very good outcome of the system we propose. Table-1 depicts the result of our system and Table-2 bellow shows the results obtained using the SVM approach. The SVM approach shows an accuracy of 98.12%.

Table-1: Results Obtained via DSP matching

Signs group	**Speed Limit and Non Speed Limit Signs**						
	No. of images	**Positive objects**	**TP**	**FP**	**Negative objects**	**TN**	**FP**
Speed Limit-30	22	22	22	0	25	24	1
Speed Limit-50	20	20	19	1	22	22	0
Speed Limit-60	23	23	23	0	28	28	0
Speed Limit-70	29	32	30	2	55	55	0
Speed Limit-80	24	24	24	0	26	27	0
Speed Limit-90	27	30	29	1	30	30	0
Speed Limit-100	10	10	10	0	17	17	0
Speed Limit-110	15	15	15	0	23	23	0
Speed Limit-120	10	10	10	0	19	19	0
Non Speed Limit Sign	20	23	21	2	25	25	0
Total	200	210	204	6	270	269	1

Table-2: Results Obtain via SVM

Signs group	**Speed Limit and Non Speed Limit Signs**						
	No. of images	**Positive objects**	**TP**	**FP**	**Negative objects**	**TN**	**FP**
Speed Limit-30	22	22	22	0	25	24	1
Speed Limit-50	20	20	19	1	22	22	0
Speed Limit-60	23	23	23	0	28	27	1
Speed Limit-70	29	32	30	2	55	55	0
Speed Limit-80	24	24	24	0	26	27	0
Speed Limit-90	27	30	29	1	30	30	0
Speed Limit-100	10	10	10	0	17	17	0
Speed Limit-110	15	15	14	1	23	23	0
Speed Limit-120	10	10	10	0	19	19	0
Non Speed Limit Sign	20	23	21	2	25	25	0
Total	200	210	203	7	270	268	2

TP, FP shows the number of speed limit signs recognized was correct and wrong respectively in case of positive objects (speed limit sign). TN, FP shows the number of non-speed limit signs

recognized correctly and wrong speed limit signs in case of negative objects (non-speed limit sign).

Comparison between our approach (DSP) and the other approach (SVM) is given bellow.

Table-3: Comparison between SVM and DSP approach

	SVM	**DSP**
Number of Image	**200**	**200**
Positive Object	**210**	**210**
TP	**203**	**205**
FN	**7**	**5**
Negative Object	**270**	**270**
TN	**268**	**270**
FN	**2**	**1**

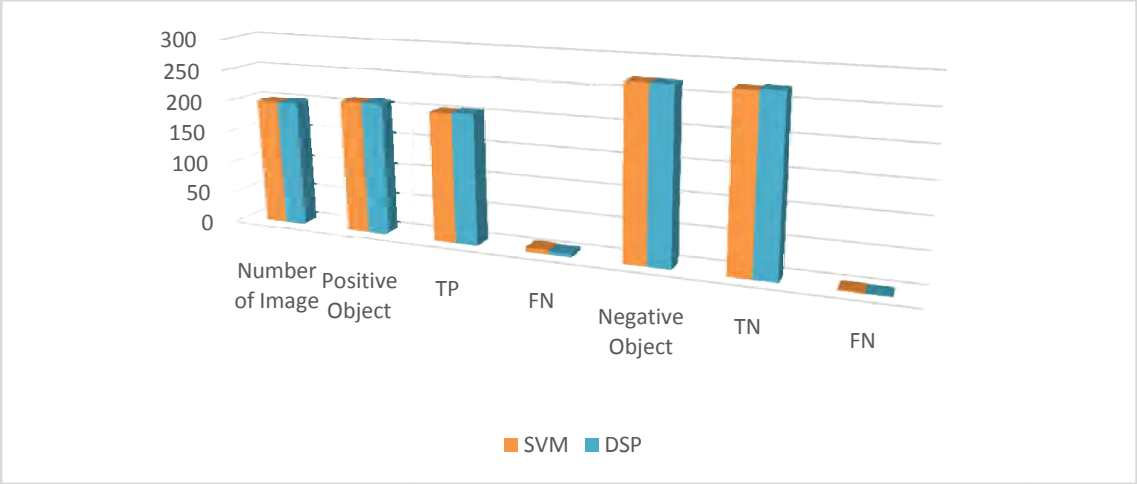


Figure-5.3.1: Comparison Chart of DSP and SVM Approach

Figure-5.3.2 below shows few of the results side by side. Left side shows Input images and Right side shows Output images.

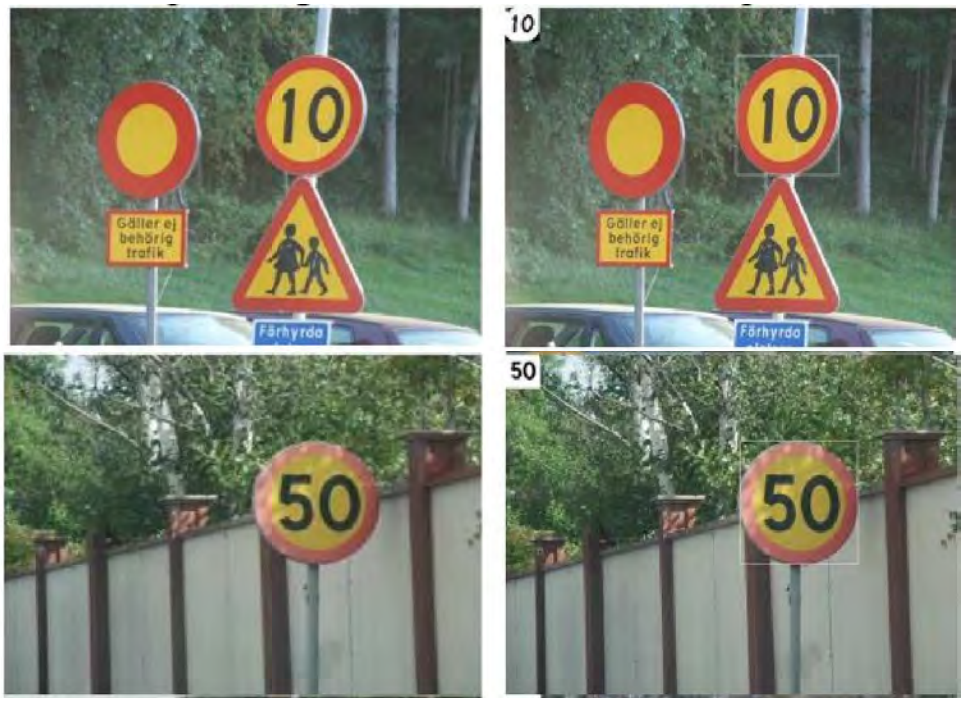


Figure-5.3.2: Results Returned by Our System

Chapter-6

In this chapter we discuss our limitations and future scope of the work. We also discuss problems we faced and possible approach for solving these issues as well. We also conclude our report in this chapter.

6.1 Concluding Remarks

In our research we introduced a novel approach to recognize speed limit signs. We have done the experiments over and over to proof correctness of our system. We believe we have a successful attempt. As automation takes place in everyday life, we hope our system will make it more enjoyable.

6.2 Limitations

The first thing which was our concern was time. Since our system is efficient enough in the sense of correct recognition rate. The rate is higher compared to the state of the art methods. We faced problem regarding time of computation.

Another thing is, while work on our system we found there are not many standard dataset available online. So to compare our work we had to rely on previous works of very specific persons whoever had publications on the same topic using same data set we used.

6.3 Future Work

Our aim is to build a real time system which is efficient at all sort of costs. To make our system or reliable we look forward to come up with parallel processing which in turn reduces computation time.

Since, we can process multiple classes of images at the same time in parallel. The whole process will take less time. We also look forward to come up with an app for multiple platforms. Though our method is straightforward, we want this system to be more developed and time efficient in future. All of these ideas are under experiment and we are really excited to implement in near future.

References

[1] J. Torresen, J. Bakke, and L. Sekanina, "Efficient Recognition of Speed Limit Signs," presented at the 2004 IEEE Intelligent Transportation Systems Conference, Washington, USA,2004.

[2] B. Hoferlin and K. Zimmermann, "Towards reliable traffic sign recognition," in 2009 IV symposium, 2009, pp. 324-329.

[3] W. Liu and K. Maruya, "Detection and Recognition of Traffic Signs in Adverse Conditions," in 2009 Intelligent Vehicle Symposium, 2009, pp. 335-340.

[4] J. Wu and Y. Tsai, "Real-time speed limit sign recognition based on locally adaptive thresholding and depth-first-search," Photogrammetric Engineering & Remote Sensing, vol. 71, pp. 405-414, 2005.

[5] D. G. Lowe. "Distinctive image features from scale-invariant keypoints". IJCV, 60(2), 2004

[6] S. Aryuanto and Y. Koichi "A New Approach for Circle Traffic Sign Tracking from Image Sequences". 21st Fuzzy System Symposium (Chofu, Sept. 7-9, 2005)

[7] John Canny. A computational approach to edge detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on, PAMI-8(6):679–698, Nov. 1986.

[8] Hasan Fleyeh, Traffic and Road Sign Recognition, PhD thesis, Napier University, Scotland, 2008. Link: http://users.du.se/~hfl/traffic_signs/

[9] Rubel Biswas, Arif Khan, MD. ZahangirAlom; "Night Mode Prohibitory Traffic Signs Recognition"; *International Conference on Informatics, Electronics & Vision (ICIEV), 17-18 May, 2013; ISBN: 978-1-4799-0397-9.*

[10] VLFeat Open Source Library. http://www.vlfeat.org/

[11] P. Felzenszwalb and D. Huttenlocher. “Efficient Belief Propagation for Early Vision”. IJCV, 70(1), 2006.

[12]Computational Neuroscience(CNBC) Group, Carnegie Mellon University “Loopy Belief Propagation”.

[13] Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. “Understanding Belief Propagation and its Generalizations”. *TR2001-22 November 2001*

[14] Just Kjeldgaard Pedersen, Simon. “Circle Hough Transform.” Aalborg University, Vision, Graphics, and Interactive Systems. November 2007.

[15] N. Otsu, "A threshold selection method from gray level histogram," IEEE Trans. Syst.ManCybern, vol. SMC-9, pp. 62-66, 1979.

[16] J. Kim, C. Liu, F. Sha, and K. Grauman. Deformable spatial pyramid matching for fast dense correspondences. In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pages 2307–2314. IEEE, 2013.