



UNIVERSIDADE DE ÉVORA

ESCOLA DE CIÊNCIAS E TECNOLOGIA

DEPARTAMENTO DE INFORMÁTICA

**Computer vision for driving support systems:
automatic traffic signs detection and proximity
analysis**

Ramez Aldwihe

Orientação: José Saias

Mestrado em Engenharia Informática

Dissertação

Évora, 2017



UNIVERSIDADE DE ÉVORA

ESCOLA DE CIÊNCIAS E TECNOLOGIA

DEPARTAMENTO DE INFORMÁTICA

**Computer vision for driving support systems:
automatic traffic signs detection and proximity
analysis**

Ramez Aldwihe

Orientação: José Saias

Mestrado em Engenharia Informática

Dissertação

Évora, 2017

For my dear mother and father and all the ones I love.

Acknowledgements

First of all, I would like to thanks my professor Jose Saias who was very nice and helpful to me with each step in my thesis. All sincere thanks to Helena Barroco who gave me the chance to come to live and study in this beautiful university and beautiful country or as I would like to say(my home). All sincere thanks to President Jorge Sampaio the founder of the Global Platform for Syrian Students which provided and still providing all kind of support. I would like to thanks, sweet and amazing Silvia because she has been always with me and still encourage me. Big thanks to my great friend Israa for standing along with me in all my happiness and sadness moments.

Contents

Contents	x
List of Figures	xii
List of Tables	xiii
List of Acronyms	xv
Sumário	xvii
Abstract	xix
1 Introduction	1
1.1 Background	1
1.2 Objectives	2
2 Concepts And State Of The Art	3
2.1 Traffic Signs Overview	3
2.2 Potential Challenges	5
2.3 Traffic Sign Recognition	5
2.4 Preprocessing Techniques	6
2.5 Detection Techniques	6
2.5.1 Detection Based On Color(Color Thresholding)	6
2.5.2 Detection Based On Shape	7
2.6 Recognition Techniques	8
2.7 Evaluation Metrics	9
2.8 State-Of-Art	10

3 The Proposed System	11
3.1 Overview	11
3.2 The Approach	11
3.3 German Traffic Sign Recognition Benchmark	12
3.4 Implementation	13
3.4.1 Image Acquisition	13
3.4.2 Detection Stage	13
3.4.3 Recognition Stage	21
3.5 Experiment Setup	21
3.5.1 Dataset	21
3.5.2 Preprocessing	22
3.5.3 Architecture Of The CNN	22
3.5.4 Training a model	23
3.5.5 Regularization	24
3.5.6 Implement The Training Phase	24
3.5.7 Summary	25
4 Results and Discussion	27
4.1 Detection Results	27
4.2 Recognition Results	30
4.3 Discussion	31
5 Conclusion	35
5.1 System Overview	35
5.2 Future Works	36

List of Figures

1.1	proposed system.	2
2.1	Warning Signs [Typ].	4
2.2	Mandatory Signs Red [Typ].	4
2.3	Mandatory Signs Blue[Typ].	5
2.4	Service Signs [Typ].	5
2.5	Challenges In Traffic Sign Detection	5
2.6	Typical block diagram of a CNN[SL11].	8
2.7	tp, fp and fn [BGF15].	9
2.8	Precision And Recall with tp, fp and fn [Rec].	9
3.1	Flowchart of the approach.	12
3.2	Color segmentation	14
3.3	Noise Filter	15
3.4	Applying size filter.	15
3.5	Connected Object Example [SSSI12].	16
3.6	Aspect ratio example.	17
3.7	Centroid test example.	18
3.8	Extraction Contours[pic]	19
3.9	template matching flowchart	20
3.10	Class distribution acroos training data.	21
3.11	Pool Layer [pol]	23
3.12	Data Augmentation.	23

3.13 Dropout Explanation [SHK ⁺ 14]	24
3.14 result curve.	25
3.15 Detected image[res]	25
4.1 Recognition example	31
4.2 Noise border problem[res]	32

List of Tables

2.1 Datasets	4
3.1 Normalized Values (Hue, Saturation)	14
3.2 Relationship between region and bounding box area	17
3.3 Matching Result Values	21
3.4 Results for first stage	25
4.1 Detection Results	27
4.2 Recognition results of image	30
4.3 Different resolution of image and execution time	32
5.1 Different results of image and execution time	36
5.2 Different results of recognition stage	36

List of Acronyms

ADC Auto Driving Cars

ADAS Advanced Driver Assistance Systems

GTSRB German Traffic Sign Recognition Benchmark

CNN Convolution neural networks

ITS Intelligent Transport System

UE Universidade de Évora

SVM Support Vector Machine

TSR Traffic Sign Recognition system

ML Machine learning

HDR High Dynamic Range

ROI Region Of Interest

Sumário

Visão Computacional aplicada a sistemas de apoio à condução: deteção automática de sinalização de trânsito e análise de proximidade

O futuro da indústria automóvel nos próximos anos irá depender fortemente das técnicas de inteligência artificial.

Esta tese propõe uma técnica para a deteção automática e o reconhecimento de sinais de trânsito a partir de imagens, para proporcionar um sistema de alerta ao condutor.

O sistema desenvolvido neste trabalho inclui algoritmos para detetar, classificar e reconhecer sinais de trânsito, nomeadamente um conjunto pertencente a uma base de dados alemã.

Os principais sinais são circulares e triangulares, os quais têm duas cores diferentes, nomeadamente vermelho e azul. Vários exemplos de imagens, em diferentes cenários, são tirados das estradas alemãs, e são usados para testar a eficácia do sistema proposto.

Os sinais de trânsito são detetados analisando a informação de cor e forma. Os sinais detetados são classificados a partir da técnica CNN Machine Learning, podendo ser classificados em 43 classes diferentes, de acordo com classificação prévia já existente na base de dados de referência.

Após deteção da presença de um sinal de trânsito, o reconhecimento do mesmo é feito comparando os sinais de trânsito detetados nas imagens com os sinais existentes na base de dados.

O acerto do reconhecimento geral é de 75% e o processamento é feito normalmente em 1.6 segundos. Este projeto foi implementado com a ferramenta OpenCV e a linguagem de programação Python.

Palavras chave: Traffic Sign Recognition(TSR), Image processing, Computer vision, CNN.

Abstract

Computer vision for driving support systems: automatic traffic signs detection and proximity analysis

The future of the automotive industry in the coming years will depend heavily on artificial intelligence techniques.

This thesis proposes a technique for automatic detection and recognition of traffic signs from images, to provide a driver alert system.

The system developed in this work includes algorithms to detect, classify and recognize traffic signs, based on a set belonging to a German database.

The main signs are circular and triangular, which have two different colors, namely red and blue. Several examples of images, in different scenarios, are taken from the German roads, and are used to test the effectiveness of the proposed system.

Traffic signs are detected by analyzing the color and shape information. The detected signs are classified according to the CNN Machine Learning technique, and can be classified into 43 different classes according to previous classification already existing in the reference database.

After detecting the presence of a traffic signs, the traffic signs is detected by comparing the traffic signs detected in the images with the signs in the database.

The overall recognition accuracy is 75 % and processing is normally done in 1.6 seconds. This project is implemented with the OpenCV tool and the Python programming language.

Keywords: Traffic Sign Recognition(TSR), Image processing, Computer vision, CNN.

1

Introduction

1.1 Background

In the last years, information technology sector has undergone significant development, especially in the Artificial Intelligence field. Many car companies (e.g., BMW, AUDI) have recently developed Auto Driving Cars (ADC). Advanced Driver Assistance Systems (ADAS) consists of multi-systems. Improving detection and classification accuracy would increase the robustness of the systems, therefore many manufacturers tend to develop in deep learning, particularly Convolutional Neural Networks(CNN), which used in a wide range of computer vision tasks like classifications and detection[HZRS16]. To drive an automated vehicle, it must ensure that cars avoid accidents and help drivers by announcing them about traffic sign on roads, in compliance with traffic laws. Normally, drivers need an effort to identify road signs in normal conditions. But in bad conditions like weather, tension, and illness. Identification traffic signs become harder. Therefore predictive techniques are being developed to maximize the safety and smoothness of driving automated vehicle in which the traffic sign recognition system takes an important position. Those kinds of intelligent systems need a good digital infrastructure. Traffic signs data must be well collected and well-formed and reliable such as German Traffic Sign Recognition Benchmark (GTSRB) dataset. When we have a good infrastructure, the system will work in right and trustful way.

1.2 Objectives

The main goal of this thesis is to build a traffic sign recognition system analyzing the image/ video taken with a camera installed in the car. As a specific objective, we plan to implement a traffic sign classification using GTSDB dataset, inspired by the previous works, and using new methods like dropout and batch normalization [SSSI12]. The aim is to build a subsystem included in ADAS to recognize traffic signs from the surrounding environment and to help drivers to avoid car accidents.

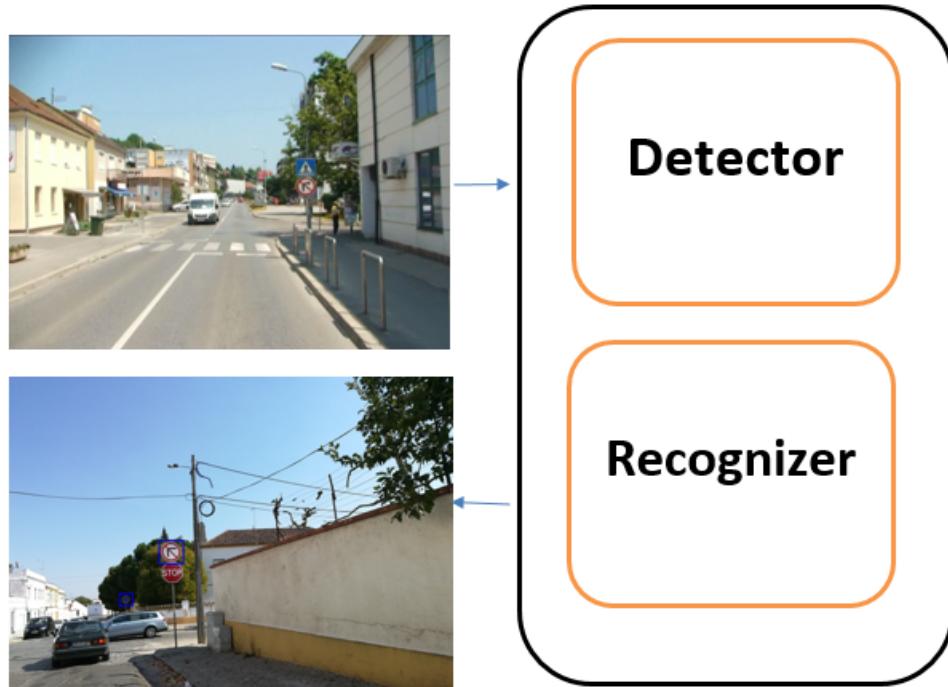


Figure 1.1: proposed system.

Structure of The Dissertation

This dissertation is composed of 5 chapters:

Chapter 1: is the introduction.

Chapter 2: is the concepts and state of art.

Chapter 3: is the proposed system and the implementation.

Chapter 4: is about the results and discussion.

Chapter 5: consists of the conclusion and the future works.

2

Concepts And State Of The Art

This chapter presented the concepts of traffic signs systems, related projects belong to the develop detection and classification systems, the convolution neural networks as a method belonging to the deep learning family.

2.1 Traffic Signs Overview

Traffic signs designed in special kind of colors and shapes, to be different from the surrounded environment and it contains specific symbols and text in a high contrast to the background. Traffic signs could be considered as a special language, that all the drivers could understand it even if they have different languages and cultures. In general traffic signs are divided to three shapes triangle, circle, and square, and have three main colors: red, blue, and yellow. The table 2.1 shows different types of traffic sign datasets.

Belgium dataset has 9,006 annotation images related to 4,591 physically distinct traffic signs, the accepted signs in the dataset took from the distance less the 50 meters from the camera. Belgium dataset has 3 main subparts: "Training", "2D Testing", and "3D Testing", as well as a classification subset[TG11]. The LISA Traffic Sign Dataset is a set of videos and annotated frames containing US traffic signs. It has two stages

Table 2.1: Datasets

datasets	images	avg. sign/image	country	classes
Belgium	4,591	3	Belgium	62
German	39,956	30	Germany	43
Liza	7,855		U.S	47

one with only the pictures and one with both pictures and videos. Sign sizes from 6x6 to 167x168 pixels. Images obtained from different cameras. Image sizes vary from 640x480 to 1024x522 pixels[MTM12].

GTSRB is a publicly available dataset, it divides into 43 classes [SSSI12], images in the data set contain one traffic sign each with a resolution between 15x15, 250x250 pixels and different lighting and weather conditions motion-blur, viewpoint variations, partial occlusions, physical damage and other real-world variabilities. The dataset was published during a competition held at the 2011 International Joint Conference on Neural Networks (IJCNN).

Types Of Road Signs

1. Warning Signs

Warning signs give advance warning of hazards on or near the road. The standard shape for warning signs in Germany is a red triangle. Warning Sign is designed with red/white combination, where red color is the border of the sign and white is the background (figure 2.1).



Figure 2.1: Warning Signs [Typ].

2. Mandatory Signs

Mandatory signs provide regulation. The standard shape for most regulatory signs is a circle. Some of them are blue and others have red borders and white background. A red circle indicates something which is prohibited, while a blue disc indicates something required (figure 2.2, 2.3).



Figure 2.2: Mandatory Signs Red [Typ].



Figure 2.3: Mandatory Signs Blue[Typ].

3. Service Signs

These traffic signs displays the services along the roads or the information that could be in driver's priorities. Service signs are a symbol in a white field on a blue square or rectangle (figure 2.4).



Figure 2.4: Service Signs [Typ].

2.2 Potential Challenges

Automated driver cars will operate in the real traffic environment. Therefore the traffic sign system must help the driver in a robust way. But in a real environment will face sometimes bad conditions like weather and illuminations conditions. Due to this problems, traffic signs are difficult to be identified. The following are the potential challenges in traffic sign recognition system shown in (figure 2.5).

- Signs may be damaged, disoriented or occulted,
- The color of the sign is corrupted with time as a result of long exposure to the sunlight
- There are variations in the lighting conditions according to the time of the day, the season, cloudiness, fog, rain and snow which causes shadowing or highlighting that effects the sensitive color information.



(a) foggy



(b) traffic light



(c) buildings

Figure 2.5: Challenges In Traffic Sign Detection

2.3 Traffic Sign Recognition

Traffic Sign Recognition system (TSR) is one of the most common vision applications in automotive industry. Because TSR is a part of Advanced Driver Assistance Systems(ADAS) performance and accuracy

is crucial factors. According to many papers[dlEAM03], [ZLZ⁺16], the common approach for traffic sign recognition system consists of three stage: preprocessing, detection and recognition. The various methods and techniques used by traffic sign recognition systems are:

Preprocessing: set up the camera configuration and resolution.

Detection: it could have two phases:

1. Color segmentation: the method could use in color segmentation is:

- Color thresholding

2. Shape segmentation: the method could use in color segmentation are:

- Edge detection
- Machine learning techniques
- Template algorithm
- Hough transform

Recognition: it has methods like:

- Template matching
- Machine learning

2.4 Preprocessing Techniques

Preprocessing stage considers as an important factor to make detection process faster. The camera captured the image from surrounding environment which has many conditions could affect the quality of the image as illumination and weather. Several configurations must set up to the camera like High Dynamic Range(HDR), resolution and size of the image.

2.5 Detection Techniques

Detection stage is the process that receives the acquired image and tries to distinguish the Region Of Interest(ROI) that could be a traffic sign and send it to the recognition stage. Several methods used to correct the detections based on color, shape and machine learning.

2.5.1 Detection Based On Color(Color Thresholding)

Detection techniques based on color threshold aim to find the interesting area from the capture frame depending on the colors of the traffic signs. There are many color spaces can be used in color threshold to separate color of interest from the background. Several models have been designed. C.H Lai and C.C.Yu in [LY10] performed the color detection in the HSV color space. The authors in [DLEMSA97] used color threshold to segment the image.

2.5.2 Detection Based On Shape

Detection based on a color is affected by changing illumination conditions. For that shape detection is useful in this case. Usually, the shape extracted by using edges, template matching or by machine learning.

2.5.2.1 Edges Detection Method

Most of the edge detection techniques depend on edge or contour information of signs. This technique is based on detection of rectangular and circular regions in the frame and use some algorithms to matching patterns. The ellipse equation has been used, in recent work, to detect the circle in an image [ASP09].

2.5.2.2 Template Matching Detection Method

We can also use template matching techniques. It based on making a comparison between the captured object and images of a known good traffic sign in the database. It used Hausdorff algorithm to implement the matching. Sheldon Waite and Oruklu in [WO13] applied this technique in face recognition and Object matching.

2.5.2.3 Machine Learning Detection Method

There are several machine learning algorithms could use in detection like Support Vector Machine(SVM) and Convolutional Neural Networks(CNN). They are used for signs detection because of ability to detect the shapes accurately [SL11]. The dataset used by machine learning algorithms is represented using a set of features that can continuous, binary or categorical. If the instances in the dataset are given with known labels then the learning is called supervised, in another hand, if the instances in the dataset are given without known labels then the learning is called unsupervised. Those algorithms are used with several datasets which have instances with known labels(supervised learning) like Belgium dataset [TG11], German and Liza[MTM12].

Convolution Neural Network(CNN)

Convolution Neural Network(CNN) is a mathematical model simulating the structure of biological neural networks [Kar15]. It consists of multi hidden layers which have an interconnected group of artificial neurons. Neurons are used complex algorithms to process information that flows within the networks. CNNs are used in variety of areas, including image and pattern recognition, speech recognition, natural language processing, and video analysis. The motivation behind designing CNN is imitation the work of visual cortex in the brain. The cells in cortex responsible for detecting the light which called receptive fields. The work of this filters are to act as filters over the input space. The convolution layer in a CNN performs the function that is performed by the Cells in the visual cortex [Con]. A typical CNN for recognizing traffic signs is shown in Figure 2.6. Each feature of a layer receives inputs from a set of features located in a small neighborhood in the previous layer called a local receptive field. With local receptive fields, features can extract elementary visual features, such as oriented edges and end-points which are then combined by the higher layers.

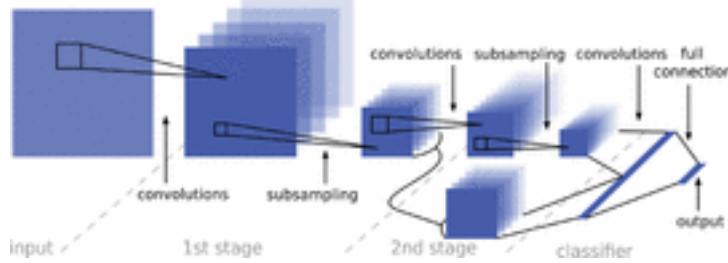


Figure 2.6: Typical block diagram of a CNN[SL11].

Layers used to build CNN

Simple CNN consists of a set of layers. The inputs through a differentiable function in these layers until we get the output score. Normal CNN will have three types of layers to build the network: Convocational Layer, Pooling Layer, and Fully-Connected Layer.

- "Convocational layers, which apply a specified number of convolution filters to the Image". For each subregion, the layer performs a set of mathematical operations to Produce a single value in the output feature map [Ten].
- "Pooling layers, which downsample the image data extracted by the convocational Layers to reduce the dimensionality of the feature map in order to decrease processing" Time [Ten].
- "Dense (fully connected) layers, which perform classification on the features Extracted by the convocational layers and downsampled by the pooling layers." In a Dense layer, every node in the layer is connected to every node in the preceding layer [Ten].

The TensorFlow library

TensorFlow is one of the recent libraries for machine learning. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research [Ten].

2.5.2.4 Hough Transform Detection Method

Hough transform is another detection techniques depend on curve fitting algorithm. the advantage of Hough transform is that not affected by noise in the image. The authors in[PT12] used it for detection of European speed limit signs.

2.6 Recognition Techniques

Recognition stage receives the ROI from the detection stage which could have one or more possibly signs. The process is to distinguish to which kind of signs that objects belong to. Because of that, this stage should be robust and reliable. The two main approaches are: machine learning and template matching which was mentioned in detection stage. Both techniques could use in detection or recognition.

2.7 Evaluation Metrics

Performance evaluation metrics can give researchers a big and clear image about their systems performance. In meantime allow them to discover the strength and weakness points as well. According to many papers[ÖAE⁺10], [GBS⁺14], the accuracy of the detection and classification can be computed using multiple performance indicators (precision, recall, and accuracy). In traffic signs recognition systems, each detected signs is referred to some class in the dataset. The indicators can let us know it the detected signs are related to one class in the dataset or it is not a sign or it is an error detection figure 2.7. To compute



Figure 2.7: tp, fp and fn [BGF15].

precision, recall and accuracy we need to know the values for some elements like(True Positive(TP), False Positives(FP) and False Negatives(FN)). The figure 2.8 explained how to calculate elements and give us a clear idea about each one. In other word we can say that: The true positive is the result of matching the same subjects, and the false positive is that of matching different subjects. False negatives (FN) is number of missing detection objects by the algorithm

- *Precision:* $tp / (tp + fp)$.
- *Recall:* $tp / (tp + fn)$.
- *Accuracy:* $tp+tn / (tp + tn)+(fp+fn)$.

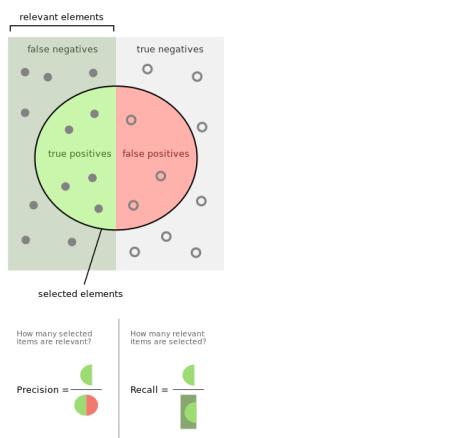


Figure 2.8: Precision And Recall with tp, fp and fn [Rec].

2.8 State-Of-Art

Since this thesis work had relied on others projects work, it is worth to mention the about the researchers.

1. *Flayeh in 2005*: A novel fuzzy approach to determine the shape of the traffic signs was presented. Every RGB image was converted into HSV color space and segmented by sing a set of fuzzy rules depending on the hue and saturation values of each pixel. For detection they rely on color segmentation method and for recognition rely on varients moments algorithm. The proposed algorithm is tested on hundreds of images and the results was robust and shows high recognition rate. The accuracy for triangles and circles is 93.3 %, the system did not work in real-time [Fle04].
2. *Pierre Sermanet and Yann LeCun in 2012*: Apply multi-task convocational Network (CNN) to use as a classifier in traffic sign system, It used GTSDB as a data set image. The input image was a grayscale 32 *32 pixels. The network accuracy is obtained 97.33%, the CNN used as detector and recognizer in the same time but they did not used as a real-time application[SL11] .
3. Zhe Zhu, Dun Liang Songhai Zhang in 2016: They created a large traffic sign benchmark dataset, provide more than 100,000 panoramic images containing 30,000 traffic sign instance. In CNN are used more layers to have more capability. CNN works simultaneously to detect and to classify traffic signs. The network achieved 84% accuracy and 94% recall [ZLZ⁺16].
4. Oukacha, S and El Kadmiri, O and Masmoudi: the proposed approach consists of detector and recognizer. The traffic signs detected by using "Hu moment method". The recognition is performed by comparing detected signs and images stored in the database. The database consists of 200 panoramic images. They used HSV color space in their project and they achieved 65% as an accuracy of the system[OEKM06].
5. Shaout, Adnan and Kaja, Nevrus and Awad, Selim: the proposed system divided into two stages. Firstly, in training stage, they used multiple traffic signs images based on different conditions as input. Then, they extract the features from images using SURF algorithm which process each image in different steps. Then, they compare the detected traffic signs with images in the database. They used the classifier as a layer on the top of an images recognition system. The classifier made by using SVM machine learning method. They achieved a good accuracy for the system 97.5% [SKA15].

3

The Proposed System

3.1 Overview

The main goal of this project is to build a traffic sign recognition system. There are many projects used different techniques to build TSR systems. Chokri Souani [SFB14] used color segmentation algorithm for detection and neural network for recognition, Oruklu [OPNG12] used color segmentation and both algorithm Template matching and Neural network for recognition. The proposed system has a new design flow. The system consists of two main stages:

- Detection the possible objects that could be candidates for traffic sign using HSV color space and using Hu variants Moments algorithm to detect signs based on their shapes
- Recognition of traffic signs using CNN classifier

3.2 The Approach

The proposed Traffic Sign Recognition system is based on the main steps given below:

1. *Image Acquisition stage*: the input data is images from the database or video
2. *Traffic Sign Detection Stage*: which includes steps to pre-processing images as follows:
 - The frames are converted into the (HSV) model from the Red, Green, and Blue (RGB) model for segmentation
 - Apply color threshold
 - The noise filter is applied to remove the noise from the frame
 - Extract connected objects.
 - Extract features from the detected objects.
 - The size filter is applied to remove the inappropriate sized objects
 - Applied "Template Matching" method which used *Hu invariants moments* theory
3. *Traffic Sign Recognition Stage*: Receive the detected objects from the previous module and identify the type of traffic signs using machine learning technique(CNN classifier)

The flow chart of the proposed approach is given below in figure 3.1:

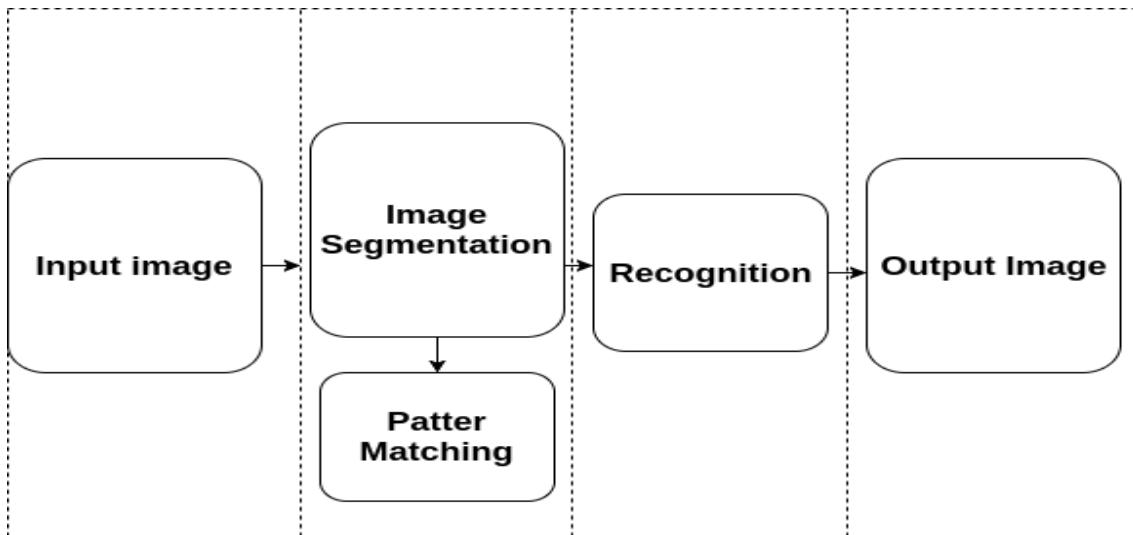


Figure 3.1: Flowchart of the approach.

German traffic sign dataset (GTSDB) is used in the proposed system, for detection, training the classifier and testing the recognizer.

3.3 German Traffic Sign Recognition Benchmark

Each country relies on specific standards for traffic signs on roads but all the standards require some properties that make the sign easily recognizable. The German Traffic Sign Dataset(GTSRB) has been chosen as an image dataset.

3.4 Implementation

As it was mentioned before, the proposed system has two stages and every stage has some steps to implement the whole system. In this section, we explained the methods and techniques used in every stage.

3.4.1 Image Acquisition

The first step in traffic sign recognition is image acquisition. In fact, the camera should be a high quality and it is better to have HDR based on CMOS (complementary metal-oxide semiconductor) technology which used in the field of driver assistance system. An input image could be taken directly from the camera installed on the car or it could be a video or an image input. The dimensions of the frame will be between 680 * 400 up to 1360*800 pixels. Once the input is taken, the image is converted to HSV color to make color and shape segmentation.

3.4.2 Detection Stage

3.4.2.1 Image Segmentation Based ON Color

The input image is acquired by the camera is in RGB color format. The three channels of color space are related strongly to each other. If the red color intensity is higher than the others for example, the image will tend to be redder. RGB color image system is sensitive to chromatic variation daylight. As a result, it will be very difficult to separate traffic signs from the background of the image. HSV color space is used for the traffic sign recognition system because it separates the chromatic and achromatic notion of light [FR98].

Converting RGB To HSV

HSV color space is named for three values: Hue, Saturation, and Value. Let $r, g, b \in [0, 1]$ be the components of RGB color space [FR98]. To find the hue angle $h \in [0, 360]$ for HSV, compute the following equation (3.1):

$$h = \begin{cases} 0 & \text{if } max = min \\ (60 * \frac{g-b}{max-min}) \bmod 360 & \text{if } max = r \\ 60 * \frac{b-r}{max-min} + 120, & \text{if } max = g \\ 60 * \frac{r-g}{max-min} + 240, & \text{if } max = b \end{cases} \quad (3.1)$$

The values for s and v of an HSV color are defined in equations 3.2 and 3.3 respectively:

$$S = \{ 0 \quad \text{if } max = 01 - MAX \quad \text{otherwise} \quad (3.2)$$

$$v = max \quad (3.3)$$

Color segmentation is the process based on color information in the frame to separate the possible candidate's traffic signs objects from the background of the frame. The colors analyzed according to the hue (H) component, the saturation (S) component is also used but for very low saturation values the color is no longer reliable. The intensity value (V) does not give any suitable information about the color therefore,

is not used for segmentation [dLEAM03]. A fuzzy color segmentation algorithm[GYJDoEE98] invoked to segment the frame according to the desired colors which in this thesis are(red and blue) see table 3.1.

Table 3.1: Normalized Values (Hue, Saturation)

Color	Hue Range [0,255]	Saturation Range [0,255]
Red	0-12,164-210	127-255
Blue	105-170	127-255

For each color threshold is applied by using the table 3.1. All pixels under threshold range will be assigned 1 and others will be assigned 0. The module will help to reduce the complex image processing by having two types of pixels "white and black" figure 3.2 showed the resulted image after segmentation.



(a) Input Image



(b) Red color Segmentation.

Figure 3.2: Color segmentation

Noise Filtering

Noise arises in the capturing process of the images when acquired by the camera or from video streaming. The noise can be recognized by a standard deviation of the pixels value in the image. The noise corrupts the information of the image. Therefore, reduction the noises are significance to computer vision applications [YZ12]. There are two different types of noise filtering:

- linear filter
- Non-linear filter

Between them, the linear filter (Bilateral Filtering) is chosen for many reasons: It uses one more (multiplicative) Gaussian filter component which is a function of pixel intensity differences which is different from the normal Gaussian filter. Multiplicative ensures that only those pixels with intensities similar to that of the central pixel ('intensity neighbors') are included to compute the blurred intensity value. As a result, this method preserves edges[NoF]. The figure 3.3 shows how the filter removes the noise.

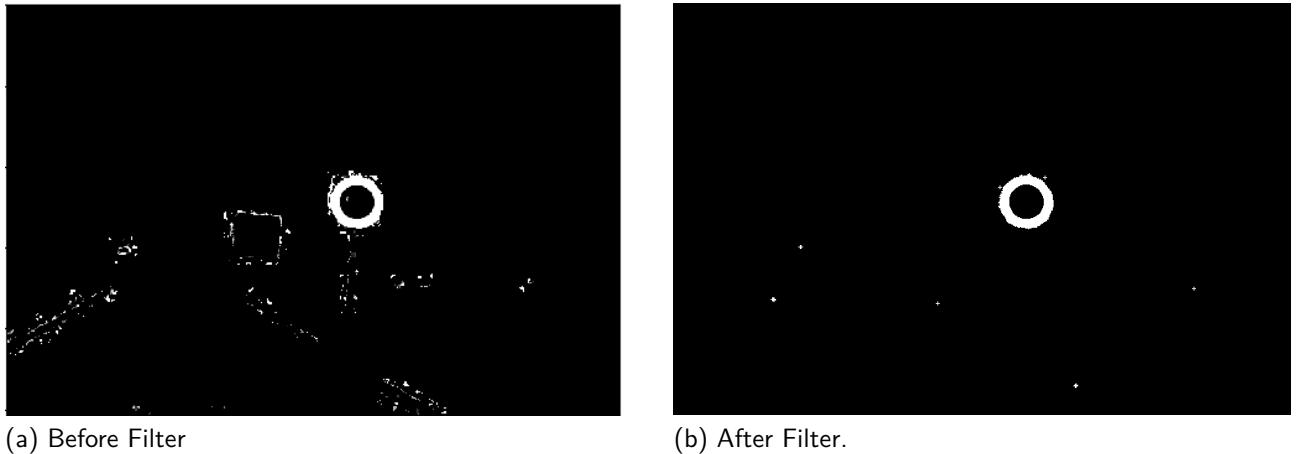


Figure 3.3: Noise Filter

3.4.2.2 Image Segmentation Based ON Shape

1-Size Filtering

Size filter is used to avoid those objects which are too small or too big to be traffic signs. Why do we need to make size filtering? Sometimes the camera installed on the car captures objects far away which without distinction. Because the system can not read the information from that distance. To make it easier for the system, we put threshold for object size. The minimum value is 20 *20 pixels. All objects less than this threshold will be deleted from the image. In meantime, we used the filter to avoid processing the objects when they came very close to the car because the driver could not have enough time to make the decision. The threshold value of the size is given by equation 3.4 which is suitable for image size 1360* 800 pixels. This saves a lot of time in real time processing because many objects will be cut, including the sky in the background with a blue hue.

$$\text{Max.Threshold.Value} = \frac{\text{imageheight}}{10} * \frac{\text{imagewidth}}{10} \quad (3.4)$$

In figure 3.4 there is an example of the use of the filter.

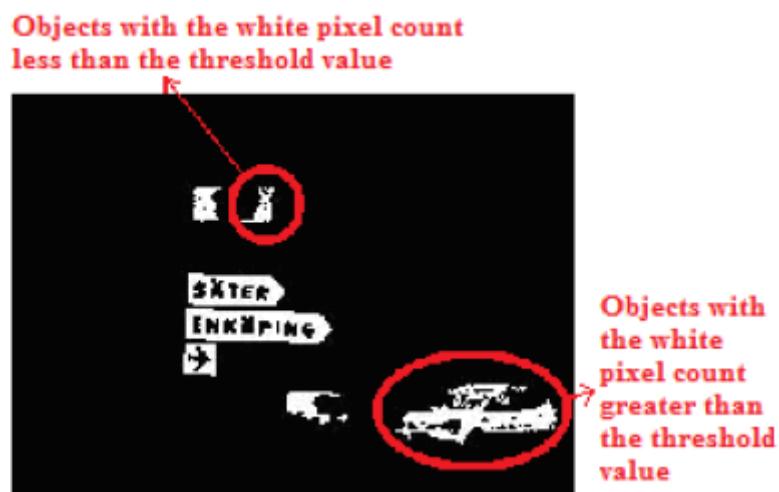


Figure 3.4: Applying size filter.

2-Connected Components

The process to detect the connected objects is depending on scanning the frame to find detect pixel groupings. This process consists of finding all pixels that share an edge or other features like summit or vertex, which could be belonging to the same group of pixels that has the possibility to be traffic signs [WO13]. "OpenCV" library offers Morphological Transformations methods to guarantee that we just detect close component.

Morphological Transformations

Morphological transformation is a technique for processing digital images depending on the shape and object form. It applies structure elements on the input image and trying to find the output pixels values in the output image by using the relation between each pixel and its neighbors. The main two basic transformations are Dilation and Erosion. Erosion play like scissors and trimming the edges while dilation has the effect of puffing out the edges[Ser82]. To create the open and close transformations we can combine them together. Which used in OpenCV library to detect the closed objects. Morphological transformations like (erosion, dilation), we used the following methods:

erosion = cv2.erode(img,kernel,iterations = 1)

dilation = cv2.dilate(img,kernel,iterations = 1)

The figure 3.5 shows the result after applying the transformations.



Figure 3.5: Connected Object Example [SSSI12].

3-Region Features

As mentioned before, the detection regions can be traffic signs and can be non-traffic signs because there are others objects similar to traffic signs like traffic lights which have the same colors and shape. Extracting and Analyzing some features allow us to have the desired candidates and give us more reliability. Some features that can be extracted from each region are:

- Area – Number of white pixels that belong to the region.
- Bounding box – The smallest rectangle, with edges parallel to the x-axis and y-axis, containing the region.

- Centroid – Center of the mass of the region.

R_i is the relation between the region area and the total area inside the bounding box area. By using this relation, it is possible to discard some of the previously detected regions that do not correspond to signs. Some examples of the regional implementation are presented in table 3.2. This result is experimental results from German traffic signs dataset, bounding box area relation values are between 0.25 and 0.90, and regions presenting values lower than 0.25 or higher than 0.90 can be safely discarded.

Table 3.2: Relationship between region and bounding box area

				
0,6830	0,6588	0,2765	0,6853	0,2797

Aspect Ratio

Aspect ratio is useful to detect a traffic sign with the bounding box. Empirically almost all the traffic sign aspect ratio is close to one. Aspect ratio is computed dividing the region bounding box width (W) by its height (H), as defined by the equation.

$$Ar = \frac{W}{H}$$

Figure 3.6 shows examples for some traffic signs aspect ratio.

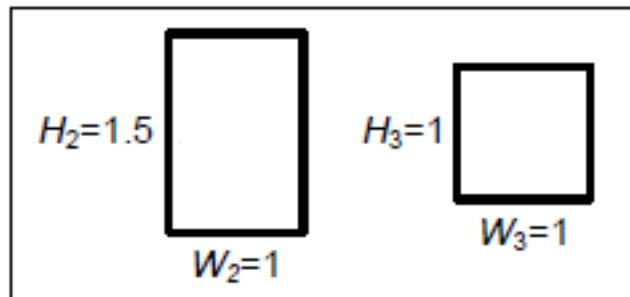


Figure 3.6: Aspect ratio example.

Centroid Test

Normally signs have approximately symmetrical shapes, having their centroid near the center of the bounding box. Considered a centroid in the center of the bounding box, in experimental test, The center of the area minus centroid less or equal 5 pixels in the x-axis and less or equal 12 pixels in y-axis consider as the center of traffic signs. Regions not respecting this rule do not correspond to traffic signs (figure 3.7).

4-Template Matching

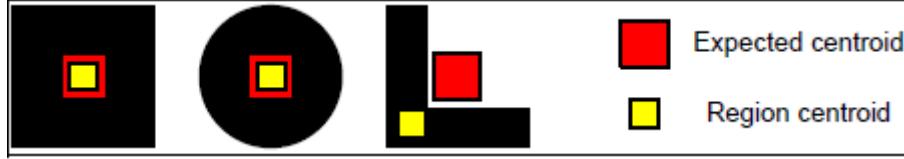


Figure 3.7: Centroid test example.

The last step in detection stage is template matching, in this thesis *Hu Moment Invariants* is used. There are some projects used template matching algorithm in recognition systems [OPNG12], [OEKM06]. The first one used the SURF algorithm and the average interesting point calculation time was 0.25 second for traffic sign with size at least 150 * 150 pixels, the second one used Hu Moments invariants algorithm to detect traffic signs in omnidirectional images and the accuracy of results was almost 65 %. In this proposed system, we applied this algorithm to detection stage with new threshold for the size of traffic signs which is at least 20 *20 pixels and in the meantime to discover the accuracy with using this method in detection and compare it with others projects to see if the system could lead real-time points with good results. *Hu Moment Invariants* is a member of a family methods of moments Invariants [Hu62] like Scale Invariant Feature Transform(SIFT) and Speed Up Robust Features(SURF) [BETVG08]. All are invariant to rotation, translation, and scaling. But *Hu Moment Invariants* is less complex and need less time than others. This method is based on Open Source Computer Vision(Opencv) [BK08], which is suitable for use in image processing.

Hu first proposed to the theory of two-dimensional moment invariants for planer geometric figures. According to [BK08], [HCT15], Two-dimensional $(p+q)$ th order moment are defined as follows:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dx dy, \quad p, q = 0, 1, 2 \quad (3.1)$$

Where $f(x, y)$ is a density distribution function. The invariant features can be achieved using central moments, which are defined as follows:

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q f(x, y) d(x - \bar{x}) d(y - \bar{y}) \quad (3.2)$$

Where, $\bar{x} = \frac{m_{10}}{m_{00}}$, $\bar{y} = \frac{m_{01}}{m_{00}}$

In a digital image $f(x, y)$, (p, q) th central and order moments are defined as:

$$m_{pq} = \sum_1^M \sum_1^N x^p y^q f(x, y), \quad p, q = 1, 2, \dots \quad (3.3)$$

$$\mu_{pq} = \sum_1^M \sum_1^N (x - \bar{x})^p (y - \bar{y})^q f(x, y), \quad p, q = 1, 2, 3, \dots \quad (3.4)$$

The normalized central moments η_{pq} define as:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \quad (3.5)$$

Where, $\gamma = (p + q)/2 + 1, p + q = 2, 3, \dots$

Based on (3.5) gamma equation, the seven moments invariants are:

$$h1 = \eta_{20} + \eta_{02}$$

$$h2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$h3 = (\eta_{30} - 3\eta_{12})^2 + 3(\eta_{21} - \eta_{03})^2$$

$$h4 = (\eta_{30} - \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$h5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + 3(\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$h6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$h7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

Contour Extraction

The main goal of contour extraction is to find a Hu moments variants for outer extracted contours for possible candidates and compare it with templates for traffic signs. The process to extract contours is, firstly, extract contours from the binary image, by using opencv methods for extraction outer contours which are useful in template matching process then fill the reserve contours. As we can see in the figure 3.8. In "Opencv", the function *findContours* retrieves contours from the binary image by using the algorithm in [S+85]. There are four contour retrieval modes as follows:

- CV-RETR-EXTERNAL retrieves only the outer contours
- CV-RETR-LIST retrieve all of the contours without establishing any hierarchical relationships
- CV-RETR-CCOMP retrieves all of the contours and organizes them into a two-level hierarchy
- CV-RETR-TREE retrieves all of the contours and reconstructs a full hierarchy of nested contours

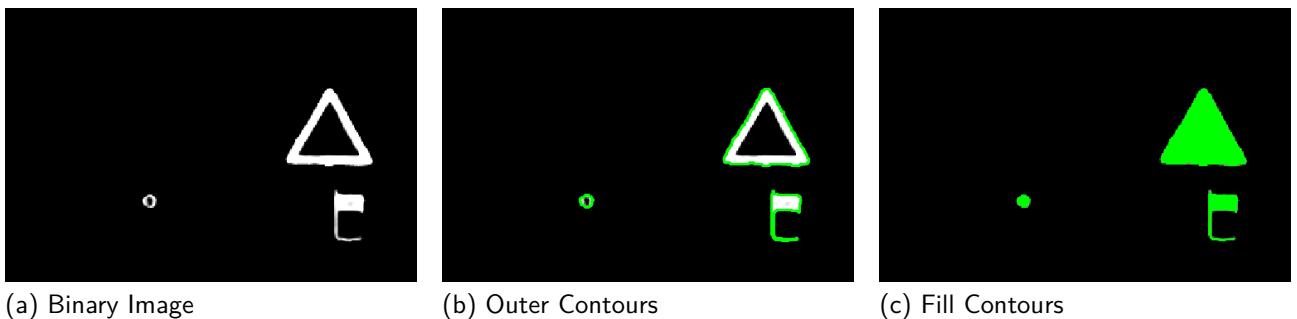


Figure 3.8: Extraction Contours[pic]

Template Matching

Template matching process steps are clarified in the flowchart 3.9 as follows:

First, pre-process template image and camera image. Extract the outer contours from template binary image, in meantime extract all the outer contours of the camera image. After that, calculate the contours Hu moments of the template and all possible contours in the camera image. Then match the contours of the camera image with contours in the template image. In "opencv", the function *matchShapes* used to compare two contours [BK08], there are three match methods available using the Hu moments:

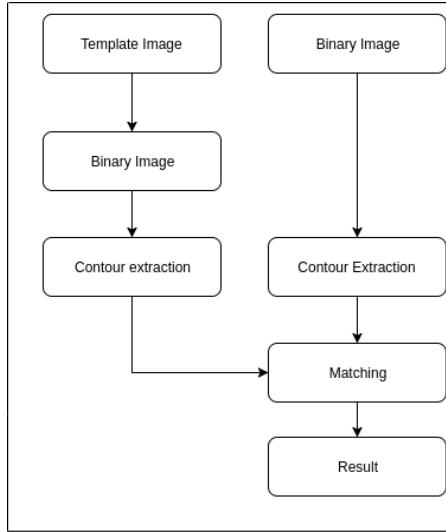


Figure 3.9: template matching flowchart

1. Method = CV-CONTOURS-MATCH-I1

$$I_1(A, B) = \sum_{i=1..7} \left| \frac{1}{m_i^A} - \frac{1}{m_i^B} \right| \quad (3.6)$$

2. Method = CV-CONTOURS-MATCH-I2

$$I_2(A, B) = \sum_{i=1..7} |m_i^A - m_i^B| \quad (3.7)$$

3. Method = CV-CONTOURS-MATCH-I3

$$I_3(A, B) = \max_{i=1..7} \frac{|m_i^A - m_i^B|}{|m_i^A|} \quad (3.8)$$

Where, A denotes object1, B denotes object2, m_i^B and m_i^A calculated by equations:

$$m_i^A = \sin(h_i^A) \cdot \log h_i^A \quad (3.9)$$

$$m_i^B = \sin(h_i^B) \cdot \log h_i^B \quad (3.10)$$

To consider that two contours are matching, the value of $I(A, B)$ should be approximate zero. The table 3.3, presents the maximum values of the matching results for each shape used in this thesis. All results less than values in tables consider that shapes are matching.

Summary

The detector took in average 0.9 seconds to analyze and detect the traffic signs from the input image.

Table 3.3: Matching Result Values

	Size pixels	Value
Triangle	[300 - 1,100]	0.05
Triangle	[1,100 - 10,000]	0.05
Circle	[300 - 1,100]	0.022
Circle	[1,100 - 10,000]	0.0115

3.4.3 Recognition Stage

In this work, we inspired from [Ale], the approach is based on traditional CNN network which means that the output of the last stage is fed the classifier. Each stage consists of a convolutional layer, a linear transform layer, and a spatial feature pooling layer. In general, CNN has one to three stages, covered by a classifier composed of one or two additional layers. In this work, we used TensorFlow open source package. Which is a new library and we wanted to compare the results to see if it is robust and works well.

3.5 Experiment Setup

This step will describe an approach to make a classifier for traffic signs using various deep learning techniques, using **TensorFlow** as a Machine Learning framework and a couple of dependencies like numpy, matplotlib and scikit-image from Python libraries.

3.5.1 Dataset

As it was mentioned before, we used GTSDB as a dataset. This dataset contains more than 40 thousands image for training and testing phases. In this experience, 39,209 32×32 px color images for training and 5,587 images that used for testing. Each image is a photo of a traffic sign related to one of 43 classes included in the dataset. Images are in RGB color system, each channel represents as in $[0,255]$ integer values. We have 43 classes and each one has a number of photos see figure 3.10.

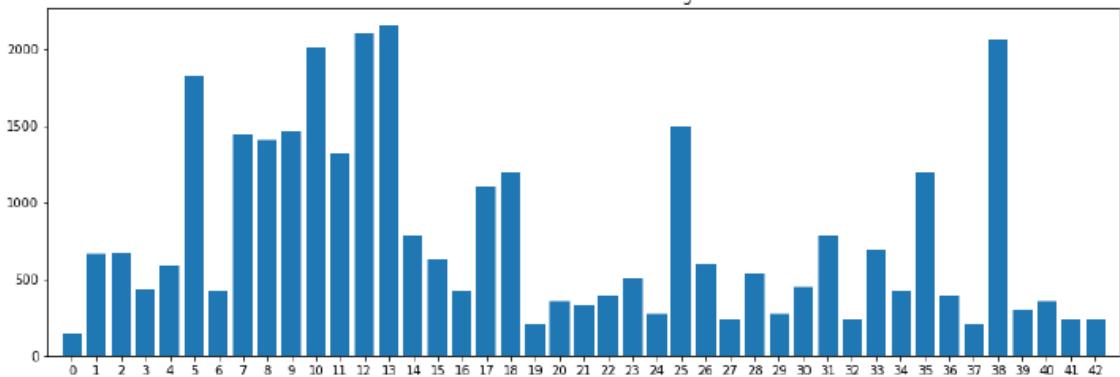


Figure 3.10: Class distribution across training data.

3.5.2 Preprocessing

All images are resized to 32 *32 pixels (the original size in the dataset is between 15 *15 and 250 *250 pixels) and in RGB color space. Representing labels in a one-hot encoding and shuffling.

3.5.3 Architecture Of The CNN

The overall architecture of "my" model is a convolutional neural network followed a similar structure as AlexNet [Ale]. The neural network architecture consists of the following 10 layers in order from input to output:

- Convolution with 3x3 kernel, stride of 1, depth of 16
- Max pooling with 3x3 kernel, stride of 1,
- Convolution with 5x5 kernel, stride of 3, depth of 64
- Max pooling with 3x3 kernel, stride of 1
- Convolution with 3x3 kernel, stride of 1, depth of 128
- Convolution with 3x3 kernel, stride of 1, depth of 64
- Max pooling with 3x3 kernel, stride of 1
- Fully-connected with 1024 hidden units
- Dropout
- Fully-connected with 1024 hidden units
- Dropout
- Fully-connected with 43 units, calculating the final 43 logits corresponding to target classes

For regularization, we used dropout on the final fully-connected layers.

3.5.3.1 Pooling layer

Pooling layers are used to shrink the images stack. In common, they added between stages of convolutions. The effect of pooling is that it allows detecting the feature even if it is in different locations in the image [KSH12]. The most common form of pooling is Max pooling where we take a filter of size F*F and apply the maximum operation over the F*F sized part of the image see figure 3.12. And to make it known a **stride** is how many pixels we slide when we apply the filter.

3.5.3.2 Fully-connected layer

Fully-connected layers usually come as the last layer in CNN classifier. The task is to process the learned features came from the previous layers and try to classify the input examples based on features [KSH12].

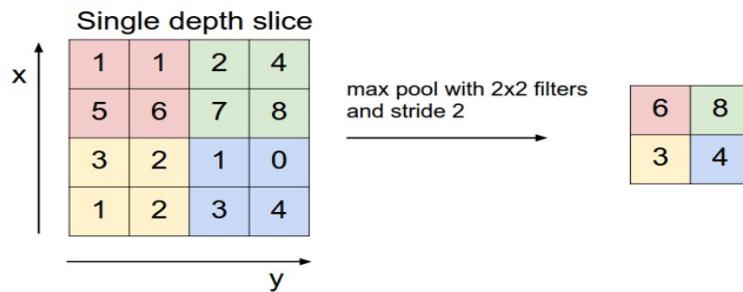


Figure 3.11: Pool Layer [pol]

3.5.4 Training a model

Training a CNN for classification means adjusting its internal parameters to increase its classification performance on the input data. In this step, we applying the augmentation to the dataset to add more 10,000 images to it. And control the capacity of the CNN to avoid overfitting by using the regularization for the dataset.

3.5.4.1 Image data augmentation

We used some methods to increase the capacity of the dataset to help us to get more accurate results.

- Choose random image from original training set.
- Perform random rotation and translation on that image.
- Add transformed image to the dataset
- Add appropriate label to augmented dataset.
- Save augmented dataset to use it.

In the figure below, it is an example of data augmentation.



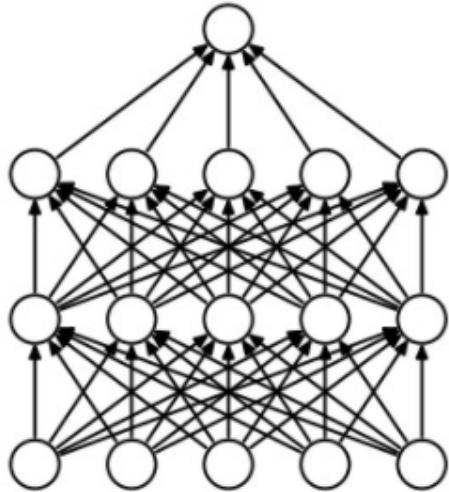
Figure 3.12: Data Augmentation.

3.5.5 Regularization

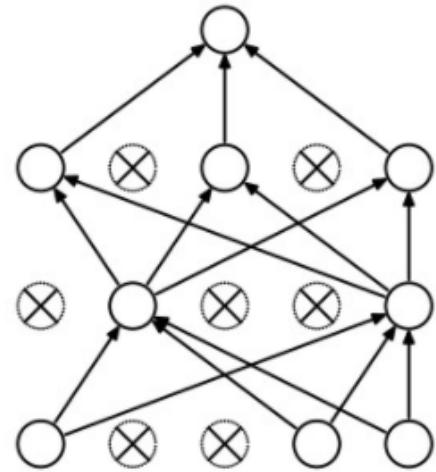
Regularization is the way that allow us to control the capacity of Neural Networks to prevent overfitting. There are many several regularization methods, In this work used two of them.

- **L2 regularization:** is the most common form of regularization. For every weight ω in the network, we add the term $1/2\lambda \omega^2$ to the objective, where λ is the regularization strength. Using the L2 regularization ultimately means that every weight is decayed linearly: $\omega' = -\lambda * \omega$ towards zero, I Used $\lambda = 0.0001$ which seemed to perform best [Neu].
- **Dropout:** is implemented by only keeping a neuron active with some probability P , or setting it to zero otherwise.

"Dropout expounds as sampling a Neural Network within the full Neural Network, and only updating the parameters of the sampled network based on the input data" [SHK⁺14]. Figure 3.14 below explains how dropout works with neurons before using it and after use it. Normally, when used dropout we noticed an improvement in performance on convolutional layers.



(a) Standard Neural Network.



(b) After Applying Dropout.

Figure 3.13: Dropout Explanation [SHK⁺14]

3.5.6 Implement The Training Phase

The dataset we used is contains 49,209 data images, split it to two parts one for training 80%, 20% for validation and 5,587 images for test.

1. We train the model using Normal dataset with TensorFlow OPT-minimize optimizer and learning rate set to 0.0001, see figure 3.15 and table 3.4.

Table 3.4: Results for first stage

Valid loss	0.07429439	Accuracy %	98.71
Test loss	0.13449727	Accuracy %	95.78
Total time:	6:03:48		

The figure 3.14 shows us how the accuracy rises all the way up to 98 percent by using 40 epoch for training the network.

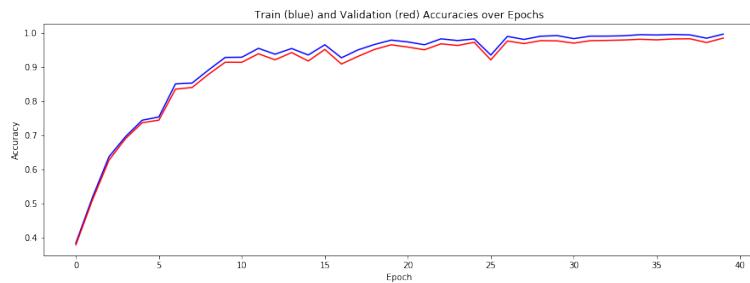


Figure 3.14: result curve.

3.5.7 Summary

In this part, we gave some example of recognition and how much time took to complete the process. The recognition took in avg 0.7 seconds, so in total, the process took 1.6 seconds. In result section, we discussed the results in more details.

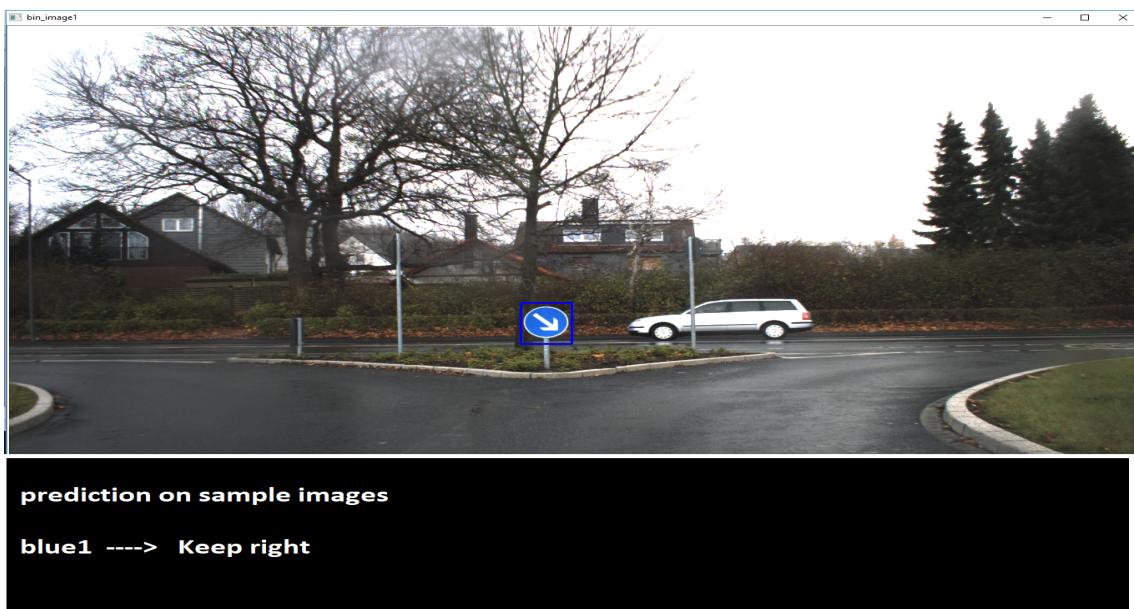


Figure 3.15: Detected image[res]

4

Results and Discussion

4.1 Detection Results

In detection phase, we used(381 1360*800) images from German dataset, for red and blue signs. We calculated the average system's response time manually by applying the snippet code below to all images, then calculate the average time. The table 4.1 shows the results and average time.

Table 4.1: Detection Results

Type	Numbers of signs	True	False	Recall	Precision	Accuracy	avg.time(second)
Red	316	185	131	0.585	0.98	0.591	0.9
Blue	65	39	26	0.6	0.951	0.582	0.9
Over all detection results	381	224	157	0.592	0.965	0.586	0.9

```
In [ ]: #method to calculate the time
st=time.time()

# read the image
name = "images/00097.ppm"
img = cv2.imread(name)

#apply the color segmentation
mask = fuzzyseg(img, (reda, redb, blue, yellow), sat)
mred = (mask[:, :, 0]*mask[:, :, 1])*mask[:, :, 4]
mblue = mask[:, :, 2]*mask[:, :, 4]

#normalize the value of images
mblue=mblue*255
mred=mred*255
mblue=mblue.astype(np.uint8)
mred=mred.astype(np.uint8)

#method that make the detector phase
im_out,im_out1 =ip.filter_image1(img,mred,mblue,contours2)

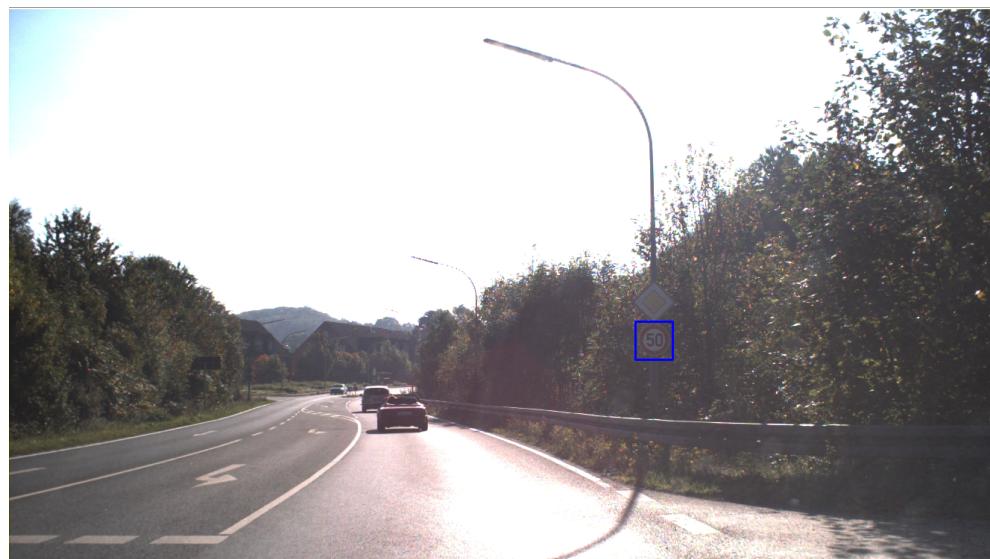
#show the output images
cv2.imshow("im_out",im_out)
cv2.imshow("im_out1",im_out1)

pd.pdd(img)
el=time.time()-st
print (el)#the end of the process for detector
delete()

cv2.waitKey()
```

In the figures below some examples are shown from German dataset. The traffic signs in examples have different sizes and the images have a different color condition. In this project, the minimum size for traffic signs is 20 *20 pixels.





4.2 Recognition Results

All the detected traffic signs are scaled to 32 *32 color images, then we executed the recognizer on the images. To evaluate the results, we considered some indicators:

- DR: detection and correctly recognized traffic sign
- DF: detection and wrongly recognized traffic sign

The table 4.2 shows the results of recognition stage. In figure 4.1, example of recognition image.

Table 4.2: Recognition results of image

Class	DR	DF	Amount	Accuracy
Keep right	9	7	16	56.25%
yield	22		22	100%
Roundabout mandatory	2	1	3	70%
Pedestrians	1	0	1	100%
Speed limit (30km/h)	22	1	23	95.65%
Speed limit (50km/h)	14	11	25	56%
Slippery road	2	0	2	100%
Go straight or left	1	0	1	100%
Ahead only	2	1	3	66.7%
No passing	8	3	11	72.72%
Stop	2	12	14	16.6%
Speed limit (60km/h)	2	5	7	28.57%
Bicycles crossing	1	0	1	100%
Speed limit (80km/h)	8	1	9	88.88%
No entry	1	0	1	100%
Turn left ahead	2	2	4	50%
Go straight or right	3	1	4	75%
Traffic signals	4	0	4	100%
Road work	4	0	4	100%
No vehicles over 3.5 metric tons prohibited	10	4	14	71.42%
Ahead only	1	1	2	50%
Speed limit (100km/h)	3	10	13	23.07%
General caution	7	0	7	100%
Beware of ice/snow	4	1	5	80%
Speed limit (70km/h)	1	2	3	33.33%
Right-of-way at the next intersection	9	0	9	100%
dangerous curve to the left	1	0	1	100%
Turn right ahead	6	2	8	75%
Speed limit (20km/h)	2	0	2	100%
Speed limit (120km/h)	2	7	9	22.22%
Bumpy road	2	0	2	100%
Dangerous curve to the right	1	0	1	100%
Total	160	72	230	75.98%



Figure 4.1: Recognition example

4.3 Discussion

As we notice that the accuracy in detection stage was almost 60 %, it is affected by many conditions as light condition and the size of traffic sign which it was in this thesis the minimum 20 *20 from image size 1360 *800 which is equal 0.03 % from the whole image. In another hand, the execution time is affected by the size of the image and the hardware are used too. In this thesis, we used a normal desktop computer with processor Intel i3 3.5 GHz, and 8 GigaByte ram.

Image Segmentation

Image segmentation is a critical part of the system. Because in this phase we collect the data related to the objects that have a possibility to be traffic signs. If we lost it then the traffic signs will be lost. We will spotte the problems in next discussions.

Problems With Hue In HSV Color Space

Hue coordinates are sensitive to any change in the RGB color space. It has three problems as stated by [Fle04]

- When the intensity is very low or very high the hue is meaningless
- When the saturation is very low, the hue is meaningless
- When the saturation is less then than the threshold value, the hue becomes unstable.

Problems With Noise Filtering

Noise filtering remove the noisy pixels from the image, but in same time it could has an effect on the edges of the traffic signs. As it was mentioned, Bilateral Filtering is used in this thesis. Bilateral Filtering shapes the edges for contours but The problem with Bilateral Filtering that it using the "Gaussian algorithm" which blurs the image. The effects have appeared with red traffic signs because it has two colors white and the border of signs are red, so when noise filter applied sometimes the borders are blurred and some pixels are removed. For that, the detector cannot detect the signs figure 4.2.

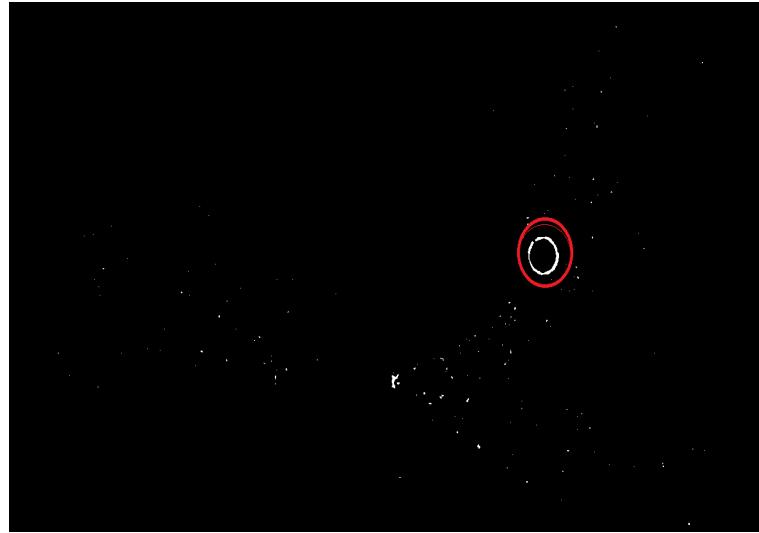


Figure 4.2: Noise border problem[res]

Size Filtering

Size filter increases the performance of the system. The system detects the blue and red traffic signs, normally in surrounding environment where the system is worked, the background includes the sky and many building with the same color as traffic signs. The filter removes the occlusion created by sky and buildings because of their large size.

Size Image problem

The size of the input image has an effect on the response time for the system. If the size of the image is small and traffic sign in the scene is small too, it will be hard to detect the sign. The table 4.3 shows 2 different results for two different sizes of the input image.

Table 4.3: Different resolution of image and execution time

Input image resolution	output image resolution	Execution time(second)
1360 * 800	1360* 800	0.9
640 *400	640 *0400	0.43

Misclassification appears in this system in two cases:

- The image has several traffic signs from the same color located in the same group
- When image has traffic signs and the background cover it with totally the same color of the sign



(b) misclassification illumination conditions.



(b) misclassification background color.

5

Conclusion

5.1 System Overview

In this thesis, we developed a traffic sign recognition system. The system divided into two stages: detection and recognition. The detector task is to separate the traffic signs from the image background, then the recognizer receives the possible traffic signs from the previous model. The task for the recognizer is to recognize the traffic signs depending on which class they belong to. In this project, we classified the traffic signs to 43 classes as GTSDB has. In detection stage, we used color and shape segmentation techniques to separate traffic signs from their backgrounds. Color segmentation relies on converting the image from RGB color space to HSV color space and then applying thresholding on it. Shape segmentation relies on "Hu variants moments algorithm", by comparing the candidates with original traffic signs. In recognition stage, we used machine learning (CNN technique) to build the recognizer.

The average accuracy achieved by the system are:

- In detection stage, we achieved almost 60%, and the system's response time was 0.9 second
- In recognition stage, we achieved almost 75%, and the system's response time was 0.7 second

In table 5.1, we made a comparison with others projects presented in chapter 2, and in table 5.2 we made a comparison for recognition with others projects that used GTSDB as a dataset too.

Table 5.1: Different results of image and execution time

Feature	This project	Flayeh [Fle04]	Pierre 2012 [SL11]	Erdeal Orukulu [OPNG12]
Detection technique	Color segmentation & shape segmentation	Color segmentation	CNN	Color segmentation
Recognition Technique	CNN	"Affin moments"	CNN	SURF
Detection result	60%	93%	97%	–
Recognizer result	75%	93%	97%	–
Min. sign size	20*20	–	–	150*150
Response time	1.6sec	–	–	0.25sec

Table 5.2: Different results of recognition stage

Project	Method	Accuracy%
IDSIA[CMMS12]	Committee of CNNs	99.46%
INI-RTCV[SSSI12]	LDA on HOG3	92.34%
This Project	CNN	73.96%

As any project, we faced some difficulties:

- Image color segmentation was a hard task within the project. Because of changing of light conditions due to shadow and highlights. For that, we always need to alter the parameters during the process
- Night images are another difficulty, because of illumination conditions. For that, in this thesis, we just considered the daylight images
- Deciding the size of the image is a difficult task. In this project, we recognized traffic signs in minimum 20*20. For that, we chose image size 1360*800, but at the same time, the big image is consumed more time for processing
- another problem is when some object is occluded by another objects.

5.2 Future Works

In the proposed system in this thesis, we achieved almost 70% accuracy. The results in detection and recognition stages which are not that satisfying since it is supposed to be used as a safety measures. As future work, the detection algorithm could be improved to better handle images with bad and highlight illumination conditions. Also, improved the algorithms for detection to avoid occlusion problems and reduce the error. Part of the future design of the system, for the classifier, we will use another CNN Technique which is: "feed-forward layered architectures the type of non-linearities used, use the connection that branched out and fed to the classifier"[SL11]. And improve the detection and recognizer efficiency to work

in real-time. To improve the result of the recognizer, we will extend the dataset with more traffic signs images, especially for the critical traffic signs. As a future work too, we will try to make a dataset for Portugal's traffic signs to make the experiences with a Portuguese dataset. As a future work, we will design a friendly user interface for the system.

Bibliography

- [Ale] alexnet, <https://github.com/tensorflow/tensorflow/blob/master/tensorflow/contrib/slim/>.
- [ASP09] Aurore Arlicot, Bahman Soheilian, and Nicolas Paparoditis. Circular road sign extraction from street level images using colour, shape and texture databases maps. In *Workshop Laserscanning*, pages 205–210, 2009.
- [BETVG08] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.
- [BGF15] Vahid Balali and Mani Golparvar-Fard. Evaluation of multiclass traffic sign detection and classification methods for us roadway asset inventory management. *Journal of Computing in Civil Engineering*, 30(2):04015022, 2015.
- [BK08] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.
- [CMMS12] Dan CireşAn, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338, 2012.
- [Con] Convolutional neural network, https://en.wikipedia.org/wiki/convolutional_neural_network.
- [dIEMAM03] Arturo de la Escalera, Jose M. Armingol, and Mario Mata. Traffic sign recognition and analysis for intelligent vehicles. *Image Vision Comput.*, 21:247–258, 2003.
- [DLEMSA97] Arturo De La Escalera, Luis E Moreno, Miguel Angel Salichs, and José María Armingol. Road traffic sign detection and classification. *IEEE transactions on industrial electronics*, 44(6):848–859, 1997.
- [Fle04] H. Fleyeh. Color detection and segmentation for road and traffic signs. *IEEE Conference on Cybernetics and Intelligent Systems, 2004.*, 2:809–814, 2004.
- [FR98] Adrian Ford and Alan Roberts. Colour space conversions. *Westminster University, London*, 1998:1–31, 1998.
- [GBS⁺14] Afzal Godil, Roger Bostleman, Will Shackleford, Tsai Hong, and Michael Shneier. *Performance Metrics for Evaluating Object and Human Detection and Tracking Systems*. US Department of Commerce, National Institute of Standards and Technology, 2014.

- [GYJDoEE98] Suwon South Korea Gang-Yi Jiang Div. of Electron. Eng., Ajou Univ. Robust detection of landmarks in color image based on fuzzy set theory. pages 968–971, 1998.
- [HCT15] Chaoxing Huang, Dan Chen, and Xusheng Tang. Implementation of workpiece recognition and location based on opencv. In *Computational Intelligence and Design (ISCID), 2015 8th International Symposium on*, volume 2, pages 228–232. IEEE, 2015.
- [Hu62] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IRE transactions on information theory*, 8(2):179–187, 1962.
- [HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [Kar15] Andrej Karpathy. Neural networks part 1: Setting up the architecture. *Notes for CS231n Convolutional Neural Networks for Visual Recognition, Stanford University*. <http://cs231n.github.io/neural-networks-1>, 2015.
- [KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [LY10] Ching-Hao Lai and Chia-Chen Yu. An efficient real-time traffic sign recognition system for intelligent vehicles with smart phones. In *Technologies and Applications of Artificial Intelligence (TAAI), 2010 International Conference on*, pages 195–202. IEEE, 2010.
- [MTM12] Andreas Mogelmose, Mohan Manubhai Trivedi, and Thomas B Moeslund. Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1484–1497, 2012.
- [Neu] neural-networks, <http://cs231n.github.io/neural-networks-2/>.
- [NoF] Noise Filters, <https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html>.
- [ÖAE⁺10] Bahadir Özdemir, Selim Aksoy, Sandra Eckert, Martino Pesaresi, and Daniele Ehrlich. Performance measures for object detection evaluation. *Pattern Recognition Letters*, 31(10):1128–1137, 2010.
- [OEKM06] S Oukacha, O El Kadmiri, and L Masmoudi. Road-signs detection and recognition in omnidirectional images. 2006.
- [OPNG12] Erdal Oruklu, Damien Pesty, Joana Neveux, and Jean-Emmanuel Guebey. Real-time traffic sign detection and recognition for in-car driver assistance systems. In *Circuits and Systems (MWSCAS), 2012 IEEE 55th International Midwest Symposium on*, pages 976–979. IEEE, 2012.
- [pic] picture, <http://www.machinelearningmentor.com/research>.
- [pol] poolayer, <http://cs231n.github.io/convolutional-networks/>.
- [PT12] Kerem Par and Oğuz Tosun. Real-time traffic sign recognition with map fusion on multicore/many-core architectures. *Acta Polytechnica Hungarica*, 9(2):231–250, 2012.
- [Rec] recall, <https://en.wikipedia.org/wiki/Precision-and-recall>.
- [res] testinphoto, <http://benchmark.ini.rub.de/>.

- [S⁺85] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.
- [Ser82] Jean Serra. *Image analysis and mathematical morphology, v. 1*. Academic press, 1982.
- [SFB14] Chokri Souani, Hassene Faiedh, and Kamel Besbes. Efficient algorithm for automatic road sign recognition and its hardware implementation. *Journal of real-time image processing*, 9(1):79–93, 2014.
- [SHK⁺14] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.
- [SKA15] Adnan Shaout, Nevrus Kaja, and Selim Awad. A smart traffic sign recognition system. In *Computer Engineering Conference (ICENCO), 2015 11th International*, pages 157–162. IEEE, 2015.
- [SL11] Pierre Sermanet and Yann LeCun. Traffic sign recognition with multi-scale convolutional networks. *The 2011 International Joint Conference on Neural Networks*, pages 2809–2813, 2011.
- [SSSI12] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural networks : the official journal of the International Neural Network Society*, 32:323–32, 2012.
- [Ten] TensorFlow, <https://www.tensorflow.org/tutorials/layers>.
- [TG11] R Timofte and LV Gool. Fast approaches to large-scale classification. In *submitted to International Joint Conference on Neural Networks*, 2011.
- [Typ] Signs Types, <http://www.gettingaroundgermany.info/zeichen.shtmlwarn>.
- [WO13] Sheldon Waite and Erdal Oruklu. Fpga-based traffic sign recognition for advanced driver assistance systems. *Journal of Transportation technologies*, 3(01):1, 2013.
- [YZ12] Cheng Huan Youlian Zhu. An improved median filtering algorithm for image noise reduction. *2012 International Conference on Solid State Devices and Materials Science*, 2012.
- [ZLZ⁺16] Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, and Shimin Hu. Traffic-sign detection and classification in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2110–2118, 2016.

