IuSigns (I understand Signs) – Detecting and Identifying Signs Using EmguCV

justin dale sandoval and henry villaber, Ateneo de Davao University

Many people nowadays migrate to other countries for their jobs and hopes of a better lifestyle. It is undeniable that traffic, street, disability signs are designed to suit the country's local preferences, and most migrants might have difficulty understanding these signs. Most of the time, symbols alone don't give people a clear understanding as interpretations differ from each person. Using Traffic Sign Recognition Systems, this paper aims to provide synthesized speech playback for these signs.

The research aims to develop an effective alternative for a neural network TSR system which accurately detects and identifies traffic signs, and to implement audio playback for the recognized signs. Using EmguCV, shape detection and contour extraction algorithms can be implemented in C#.

For testing and evaluation, live camera capture, as well as gathered videos and images are processed in the application. There are two types of scenes: those with traffic signs, and those without traffic signs. There should be no detection made when a scene without a traffic sign is processed in the application.

Categories and Subject Descriptors: **Traffic Sign Recognition; Contour Analysis; Shape Detection; EmguCV**

1. INTRODUCTION

1. Background of the Study

Traffic Sign Recognition (TSR) is a technology used by modern vehicles to recognize traffic signs on the road. TSR is part of the features collectively called ADAS (Advanced Driver Assistance System). Since most related studies utilize Neural Networks for recognition, this research wants to find out if the system can be implemented using EmguCV.

The goal of this research is to determine the capability of the proposed system, and its limitations. This research also aims to determine the proper angle and position of the camera inside the vehicle. Moreover, the research aims to determine the vehicle speed that in which the application can still clearly detect traffic signs.

2. Significance of the Study

This research provides Driver Assistance especially for new drivers that are not yet familiar with the different signs on the road. Since it is a driver assistance system for vehicles, it will vocally dictate the traffic signs that are detected by the system. As time goes by, with the help of this system, it will aid the beginner drivers to be familiar with different traffic signs. This research will not focus only on beginner drivers, but also on experienced drivers in assisting them to detect signs they pass by. With this system, we can help to decrease road accidents because the drivers will be guided and reminded on what traffic signs are on the road.

3.

Technology Application Context

There are many ways to implement TSR. Some of those are to use Neural Networks, image matching, shape detection, MSER, etc. In this research, the combination of shape detection and contour analysis was used. The proponents found that no study has been conducted using these algorithms in detecting traffic signs.

The research is done using EmguCV, a cross platform .NET wrapper which allows OpenCV library functions to be called from .NET compatible languages such as C#, C ++, and VB.NET. The wrapper can be run on Windows, Linux, and Mac OS X, and Android devices.

In this research, image processing methods such as image thresholding, noise reduction, canny image processing, and contour detection are used. The video feed will undergo these methods through OpenCV functions before identifying the traffic signs it contains. The shape detection algorithm is processed first to detect the signs. Once detected, it is cropped and the contour analysis algorithm is used to determine what sign has been detected.

4. Objectives of the Study

General Objectives:

(1) To implement TSR using EMGU CV

(2) To detect and recognize traffic signs in a moving environment.

(3) Voice playback on recognized signs.

Specific Objectives:

(1) Detect signs in real-time video capture in a moving environment with normal weather conditions.

(2) Enable to detect and recognize signs in different speed (KPH)

(3) Enable to distinguish the desirable camera position.

(4) Enable voice playback of the recognized signs.

5.

Conceptual Framework

Figure 1.a Conceptual Framework

The application takes input from the camera's live video feed. Each frame taken will be passed on to the processes. First, the frames are thresholded before edges are detected. Next, the application extracts the contours from the frame. The application then proceeds to look for specific shapes, which are commonly used in traffic sign borders. After detecting the shape, the contours inside are compared to our templates to look for a possible match. Finally, if the contour is recognized as a traffic sign, an audio file will be played, classifying the traffic sign.

6.

7. Scope and Limitations

The scope of this project is for the detection of road signs in different angles, with the speed of the car at 30kph to 60kph, and within the distance of 5 to 20 meters. Vehicle speed and recognition distance are greatly affected by the computer's system specifications, and the camera's capture quality. The better the specifications are, the faster the processing speeds is and the better the image quality, the greater the distance for signs to be recognized. The limitations of this research include: minor obstructions, vandalisms on the traffic sign, night condition, and foggy condition. The reflection of light on the traffic signs also prevents the application from detecting the signs. Moreover, when the car's dashboard is reflected on the windshield, it serves as unwanted noise on the video stream. This also has a big impact on the application's performance.

2. LITERATURE REVIEW

1.

1.1. Text Signage Recognition in Android Mobile Devices (Oi-Mean Foong et al., 2013)

The proponents of this paper attempted to develop a text signage recognition model for android mobile devices. This paper is aimed at visually impaired people who have trouble acquiring gadgets that would help them, because these devices are either too expensive or difficult to handle. They used the Tesseract OCR engine for the recognition of characters found in the sign, which would then be converted to text as input for the Text-To-Speech translation. The adaptation of OCR on mobile devices poses a challenge since mobile devices have much lower processing capabilities than standalone workstations or personal computers. There were many problems observed in the results of the research. Signage images with white colored text could not be recognized correctly, as the texts blended with the white background image where it was pasted to. Problems with the letters “Q” and “O” were also present, as the system recognized the character “Q” as “O” and vice versa.

1.2. Traffic Sign Recognition (C. ÖZGEN, 2009)

Designing smarter vehicles, aiming to minimize the number of driver-based wrong decisions or accidents, which can be faced with during the drive, is one of hot topics of today’s automotive technology. In the design of smarter vehicles, several research issues can be addressed; one of which is Traffic Sign Recognition (TSR). This research is conducted for safety driving for vehicle drivers. The research applies three stages, first is "Segmentation", second is "Reconstruction" and third is "Identification". In segmentation, there are two major distinctive properties to be considered, one is the color and the other is shape. Either it will undergo color segmentation or segment the image extracting shape information. The segmented image will then be thresholded in order to get the binary result and determine the shape of the traffic sign that is recognized. In the reconstruction phase, the segmented image will be resized and skewed for a clear picture. Then it will proceed to identification, which the third step. Features of the image will be extracted, and then it will be classified by template matching and neural networks.

1.3. Traffic Sign Recognition for Intelligent Vehicle/Driver Assistance System Using Neural Network on OpenCV (A. Lorsakul, J. Suthakorn, 2007)

The paper utilizes Neural Network technology and applies it to Traffic Sign Recognition. The proponents aim to develop a fast and efficient way for the detection of road signs by reducing the search space. This in turn requires a robust and faster intelligent algorithm in order to provide the accuracy necessary for the recognition of traffic signs. The acquired images are pre-processed, enhanced, and segmented according to the sign’s properties: color and shape. Traffic sign images are investigated to detect pixel regions where road signs might possibly appear. The potential objects are then normalized to a specific size before being treated as an input for the recognition phase. The research applies two strategies for reducing the computational cost in facilitating real-time implementation. First is to reduce the number of MLP inputs by pre-processing the captured image, and second is to search for the best network architecture which reduced complexity by selecting a suitable error criterion for training.

	Research Problem/Objective	Methodologies
Text Signage Recognition in Android Mobile Devices (Oi-Mean Foong et al., 2013)	Text Signage Recognition in Android mobile devices for Visually Impaired People.	• Image Pre-processing • Otsu's algorithms (Otsu, 1979) • Tesseract OCR • Speech Synthesizing Process
Traffic Sign Recognition (C. ÖZGEN, 2009)	Reconstruction of occluded traffic signs using three stages, "Segmentation", "Reconstruction", and "Identification".	• Segmentation • Reconstruction • Identification • Template Matching • Neural Networks
Traffic Sign Recognition for Intelligent Vehicle/Dri-ver Assistance System Using Neural Network on OpenCV (A. Lorsakul, J. Suthakorn, 2007)	Recognizing traffic sign patterns using Neural Networks.	• Image Extraction • Sign Detection • Form Recognition • Multilayer Perceptron (MLP)
Contour Analysis for Image Recognition in C# (P. Torgashov, 2011)	Describes the theoretical bases of the contour analysis and aspects of its practical application for image recognition	• Image Binarization • Contour Extraction • Contour Equalization • Template Matching

Figure 2.a Summary of Related Literature

2.

3.

METHODOLOGY

Traffic sign recognition is a technology which detects and recognizes different signs in the environment. So, this technology will help the user in interpreting what these signs mean.

3.1. Research Methods:

This research will follow a step by step procedure in approaching the field of study. The first part is the predevelopment and development stage. In this stage, the gathering of data and research papers takes place. The analysis of hardware and software that will be used in performing the implementation of the research comes after. After creating an algorithm in EmguCV/OpenCV that enables the detection of signs, matching of the recognized sign to the voice templates follows. The last step is to test and evaluate all of the processes.

3.2. Predevelopment and Development stage:

Before this research is put into implementation, the first step is to collect important information and research papers related to the field of the study. Analysis and determining the processes involved in the implementation of this research comes next. The gathering of data for the research will determine which processes or methods will be applied. Also, one of the important data of the research is the results from the testing and evaluation. The research must determine the kind of testing methods and what are the testing criteria to be used in order to make sure that all process work well.

After the compilation of information from the predevelopment stage, the research takes a step into the development stage. This stage will be the analyzing of compiled data, as well as determining the hardware and software that will be used in the implementation. It is important that the hardware and software to be used is analyzed in order to determine the advantages and disadvantages of these hardware and software.

3.3. Software:

OpenCV is an open source computer vision library originally developed by Intel. Currently, this library supports: real-time capture, video file import, basic image, treatment (brightness, contrast, threshold, and etc.), object detection, blob detection. Since OpenCV have capability to recognize objects, this research will formulate an algorithm that will recognize traffic signs, road signs, and etc.

EmguCV is a cross platform .Net wrapper to the OpenCV image processing library. Allowing OpenCV functions to be called from .NET compatible languages such as C#, VB, VC++, IronPython etc.

3.4. Implementation

The application is made with Microsoft Visual Studio in .NET platform with the EmguCV and OpenCV libraries. The program is coded in the C# language. The video feed will undergo shape detection algorithm followed by the contour detection for sign identification. Once the sign is identified, the application will notify the user via a voice playback.

a. Shape Detection Algorithm

The image frames are first thresholded, edges are detected by Canny Edge Detection, and then binarized. The shapes the algorithm will detect are triangles, rectangles, and circles. The detected shapes are cropped afterwards for the contour detection algorithm.

b. Contour Extraction

Figure 3.a (a) Image from scene. (b) Binarized Image. (c) Image with Contours shown.

The image resulting from the shape detection algorithm is subject to contour analysis. The image is again thresholded to clear out unnecessary noise and artifacts in the image. The outlines are drawn before the area of the contour is analysed. Contours with an area less than the specified are disregarded.

c. Template Matching

Extracted contours are then matched in the template database. If the contour is recognized in the database, the voice playback matching the template will be played.

8. Testing and evaluation

If application design is done, the research will be subject to testing and evaluation. The application must be able to detect and recognize the signs in different angles. If the technology detects a sign, it will display the textual form of the recognized sign and the voice notification will play. If the process is well executed without bugs and errors, the testing part of the research is done.

4. TECHNOLOGY BACKGROUND

In order to successfully achieve the goals of this project, the TSR technology is used for the recognition of traffic/street signs. EmguCV will be used as the computer vision library needed for the detection of interest points in the video frame. TSR technology is used by car companies (e.g. BMW, Mercedes Benz, and Volkswagen) as a system which detects speed limits and overtaking restrictions. The EmguCV library holds tools useful in conducting this research, such as object detection, blob detection, and real-time capture.

5. RESULTS AND DISCUSSIONS

This paper aims to create an application that can detect and recognize traffic signs in varying conditions. These conditions are: varying angles, varying speeds (vehicle speed in KPH), and weather conditions. From these different conditions, the application should be able to detect and recognize traffic signs if the conditions pass.

This research considers the desirable position of the camera in order to get the angle at which the application can successfully detect and identify the traffic signs. The proponents observed that there are differences between the drivers' perspectives, passengers' perspectives and central perspectives, which would be in the same position of the rear view mirror. If the camera's position is in the driver's perspective (left side of the car), the distance will be further from the nearest traffic sign which would most likely be located at the right side of road. There would be an estimated increase of 1 to 2 meters in distance depending on the width and height of the car. The average detection distance between the camera and the traffic sign, is approximately 8 to 10 meters considering the camera position. If the camera is positioned in the passenger side of the car, there is the possibility that the traffic sign appears at a blind spot of the camera. The proponents identified that the best position of the camera is at the central perspective as the proponents observed that there is higher accuracy in detecting signs as the chances of encountering blind spots are lower, and with only a minimal increase in distance from the camera and the nearest traffic sign.

Focusing in varying speeds, part of the limitation of this research is video capture during high speed driving, as capture quality depends on the specifications of the camera and the processing capability highly depends on the specifications of the computer used. The proponents of this research tested using an A4Tech 1080P Full-HD Webcam. The proponents observed the average vehicle speed in which the application is successfully able to detect and recognize a sign is at most 35-40 KPH. Furthermore, if the vehicle speed goes beyond 30 KPH some frames become blurred, which will decrease the detection rate of the application. Considering the limitations of the camera and computer used, the proponents consider 30 KPH as the average speed for proper detection and recognition of the traffic signs.

Focusing in varying weather conditions, the scope of this research only includes daylight with good weather conditions. If the weather condition is rainy, the vision of the camera will be obstructed by the rain falling on the windshield. Furthermore, in night conditions, it becomes hard for the camera to capture the signs. So the best weather condition for this research is during daylight, when the traffic signs have clear lighting and there are no obstructions present on the windshield.

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Media;
using System.Runtime.Serialization.Formatters.Binary;
using System.Windows.Forms;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;
using TSRLibrary;
using System.Threading.Tasks;
```

Figure 5.a Imported Libraries

These are the libraries required for the application to run.

```
public void Process_Frame(Image<Bgr, byte> frame {
while (frame != null){
```

```
Image<Gray, byte> grayFrame = frame.Convert<Gray, byte>();
Image<Gray, byte> smoothedGrayFrame = grayFrame.PyrDown();
smoothedGrayFrame = smoothedGrayFrame.PyrUp();
grayFrame = smoothedGrayFrame;
Image<Gray, byte> cannyFrame = null;
cannyFrame = smoothedGrayFrame.Canny(175.0, 175.0);
CvInvoke.cvAdaptiveThreshold(grayFrame, grayFrame, 255,
Emgu.CV.CvEnum.ADAPTIVE_THRESHOLD_TYPE.CV_ADAPTIVE_THRESH_MEAN_C,
Emgu.CV.CvEnum.THRESH.CV_THRESH_BINARY, 10 + 10 % 2 + 1, 1.2d);
grayFrame._Not();
if (cannyFrame != null)
cannyFrame = cannyFrame.Dilate(3);
var sourceContours = grayFrame.FindContours
(Emgu.CV.CvEnum.CHAIN_APPROX_METHOD.CV_CHAIN_APPROX_NONE,
Emgu.CV.CvEnum.RETR_TYPE.CV_RETR_LIST);
int maxArea = grayFrame.Width * grayFrame.Height / 5;
contours = Clean_Contours(sourceContours, cannyFrame, maxArea);
```

Figure 5.b Processing of captured frames

Captured frames are processed in the function “Process_Frame”. Here they are converted to grayscale and undergo image processing methods. The frames are smoothened by downscaling the image by one level, and upscaling it again another level. This produces a blurred effect on the image, thus smoothening it. Next, Canny-Edge detection is applied to the smoothened frame. This filters out objects so that there are less unwanted artifacts. The smoothened frame is thresholded or binarized to make contour extraction more efficient.

Figure 5.c Original Scene

Figure 5.d Conversion to Gray

Figure 5.e Application of Canny Edge

Figure 5.f Thresholding of Image

```
private List<Contour<Point>> Clean_Contours(Contour<Point> contours,
Image<Gray, byte> cannyFrame, int area)
{
List<Contour<Point>> result = new List<Contour<Point>>();
while (contours != null){
if (contours.Total < 55 ||contours.Area < 60 || contours.Area > area ||
contours.Area / contours.Total <= .5)
goto next;
Point p1 = contours[0], p2 = contours[(contours.Total / 2) % contours.Total];
if (cannyFrame[p1].Intensity <= double.Epsilon && cannyFrame[p2].Intensity <=
double.Epsilon)
goto next;            result.Add(contours);
next:
contours = contours.HNext;}
return result;}
```

Figure 5.g Cleaning out Unwanted Contours

The canny-edged frame is passed into this function. This filters out irrelevant contours produced in the application. Only the contours with an area above the specified value will be processed. Epsilon represents the smallest positive System.Double value greater than zero. This is constant. If the gray intensity of the first contour point is less or equal to Epsilon, and the gray intensity of the midpoint of the contour is less or equal to Epsilon, it proceeds to the next set of points. These points are the "noise" in the current frame, and are left out. The rest of the points are added to a list, which are returned to the main program.

Figure 5.h Before Cleaning

Figure 5.i After Cleaning

```
if ((prev.Equals("Triangle") && !current.Equals("Triangle")) || (!prev.Equals
("Triangle") && current.Equals("Triangle")))
if (prev.Equals("Triangle") && !current.Equals("Triangle")) {
if(trafficSign.template.name.Equals("Roundabout A") ||
trafficSign.template.name.Equals("Roundabout B") ||
trafficSign.template.name.Equals("Roundabout C")) {
text = "Round About Ahead";
playSound(text);}
else { text = current; }
playSound(text);}
if ((prev.Equals("Pentagon") && !current.Equals("Pentagon")) || (!prev.Equals
("Pentagon") && current.Equals("Pentagon")))
if (prev.Equals("Pentagon") && !current.Equals("Pentagon")){
text = current;
playSound(text);}
```

Figure 5.j Shape Detection

After having filtered the frames, We then check for shapes in the current frame. This is necessary as Traffic Signs have a fixed border around them. This will improve accuracy of the program by detecting only those contours inside the specified shape.

Figure 5.k Detection of Shape

```
if ((prev.Equals("Triangle") && !current.Equals("Triangle")) || (!prev.Equals
("Triangle") && current.Equals("Triangle")))
if (prev.Equals("Triangle") && !current.Equals("Triangle")){
if(trafficSign.template.name.Equals("Roundabout A") ||
trafficSign.template.name.Equals("Roundabout B")   ||
trafficSign.template.name.Equals("Roundabout C")){
text = "Round About Ahead";
playSound(text);}
else { text = current; }
playSound(text);}
if ((prev.Equals("Pentagon") && !current.Equals("Pentagon")) || (!prev.Equals
("Pentagon") && current.Equals("Pentagon")))
if (prev.Equals("Pentagon") && !current.Equals("Pentagon")){
text = current;                              playSound(text);}
```

Figure 5.l Detecting templates inside Shapes

The program scans the inside of the shapes, to look for contours. Since the research uses contours, some signs have to be broken into parts.

Figure 5.m Template Found Post-Shape Detection

```
foreach (var model in models){
if (Coordinate_validator(sample, model))
continue;
comparison_rate = model.normalizedProportion.NormDot
(sample.normalizedProportion).Norma;
if (comparison_rate < 0.98d)
continue;
iCorr = model.polygon.InterCorrelation(sample.polygon).FindMaxNorma();
comparison_rate = iCorr.Norma / (model.normalize_value *
sample.normalize_value);
if (comparison_rate < 0.96d)
continue;
if (Math.Abs(iCorr.Angle) > Math.PI)
continue;
if (comparison_rate >= qualified_rate){
qualified_rate = comparison_rate;
match_templates = model;
angle = iCorr.Angle;}
}
```

Figure 5.n Template Matching

After scanning for contours inside the shape, We check for possible template matches. The criteria used for a match is the comparison rate, which checks the similarities of the detected contour from the template file. If the similarity for the ACF is less than 98%, it proceeds to check for the ICF similarity rate. Given the condition, the ICF similarity of the sample found should be at least 96%. Computations for ACF and ICF are taken from the Contour Analysis by P. Torgashov.

Figure 5.o Extracted Contour From Scene

Figure 5.p Contours in Template

```
voice = new SoundPlayer(AppDomain.CurrentDomain.BaseDirectory + "\\" +
text + ".wav");
if (prevDetected != text){
try{                voice.PlaySync();}
catch (Exception ex){                   Console.WriteLine(ex.Message);}
}
prevDetected = text;
```

Figure 5.q Audio Player

After having found a match, the program proceeds to the voice player. In order to prevent an infinite loop of the audio playback, the previously detected sign must be different than the currently detected sign. The problem with this is that if the next traffic sign is the same as the previous traffic sign, the program will not play the audio.

Figure 5.r Traffic Sign Templates

3.5. Testing Results

A total of 21 videos were used as samples for testing. There are 47 signs in total contained in the videos. Among the 21 videos, 3 are random videos containing no traffic signs. This is to test whether or not the application could detect objects unnecessarily. If there is no traffic sign present in the test video, the application should make no detections.

Test Case	Signs Present	Correct Signs Detected	# False Detections	# Correct
1	No U-Turn 30 Speed Limit No Parking	No U-Turn 30 Speed limit	1	2
2	No U-Turn School Crossing 30 Speed Limit Crossing	No U-Turn 30 Speed Limit	3	2
3	No Parking Traffic Signals Ahead	No Parking	1	1
4	30 Speed Limit No Parking	None	1	0
5	No Parking No U-Turn No Crossing	No Parking No U-Turn	2	2
6	No Right Turn No U-Turn No Parking	No Parking No Right Turn	0	2
7	School Crossing Pedestrian Crossing No Parking	No parking	3	1
8	Pedestrian Crossing 30 Speed Limit	Pedestrian Crossing 30 Speed Limit	1	3
9	No Parking No U-Turn No Left&U-Turn Pedestrian Crossing Intersection	Intersection No U-Turn No Parking	3	3
10	No Parking Crossing 30 Speed Limit	None	1	0
11	School Crossing Intersection Pedestrian Crossing	None	2	0
12	No Parking No Right Turn Straight and Left-Turn Only	No Parking Straight and Left-turn only No Right-Turn	3	3

13	No Parking	No Parking	2	1
14	30 Speed Limit No Parking	30 Speed Limit No Parking	1	2
15	Approaching Intersection	Approaching Intersection	0	1
16	No U-Turn 30 Speed Limit	30 Speed Limit	0	1
17	Crossing	Crossing	0	1
18	Intersection	Intersection	1	1
19	None	None	0	1
20	None	None	0	1
21	None	None	1	0

Figure 5.s Test Results

Figure 5.t Accuracy Diagram

3.6. Conclusion

The study developed an application for the detection and recognition of traffic signs. Based on the tests made, the accuracy of detecting signs correctly is at 38%. False detection rate is at 36%, and No-detection rate is at 26%. The tests yield this result with a total of 47 samples, with 6 samples having a negative result due to sun glare, since our RRL's which uses Neural Networks obtained an accuracy of 54% from 7 samples. At the end, the research is implemented with a different algorithm. This research developed a novelty in traffic sign detection and recognition using EmguCV with Shape Detection and Contour Extraction, without the use of neural networks.

6. Recommendations

The following recommendations have been proposed for future studies similar to this paper:

- **Minor obstructions found on the signs are a problem yet to be addressed in this paper. The proponents suggest that future studies will step on image reconstruction so that obstructed traffic signs can still be detected and identified.**

- **The range for detecting traffic signs is still relatively low. The proponents suggest increasing the range of detection in future studies.**

- **The application is programmed to work only on daylight conditions. The proponents suggest that future studies implement algorithms that enable traffic sign detection even during night conditions.**

- **The proponents also suggest the implementation of a timer, so that audio playback will not run continuously if signs are detected.**

DEFINITION OF TERMS

TSR – Traffic Sign Recognition; a technology used by modern vehicles to recognize traffic signs on the road.

Thresholding – The isolation of objects or other relevant information in digital images.

Contour – The outlines or silhouettes in an image; outline representing or bounding the shape or form of an object.

Grayscale – Images composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest.

Noise – Unwanted or unnecessary artifacts found in the image.

BIBLIOGRAPHY

Foong, O. M., Sulaiman, S., & Ling, K. K. (2013). Text signage recognition in Android mobile devices. Journal of Computer Science, 9(12), 1793.

Lorsakul, A., & Suthakorn, J. (2007). Traffic sign recognition for intelligent vehicle/driver assistance system using neural network on OpenCV. In Proceedings of the 4th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2007) (pp. 22-24).

ÖZGEN, C. (2009). TRAFFIC SIGN RECOGNITION (Doctoral dissertation, MIDDLE EAST TECHNICAL UNIVERSITY).

Brkic, K. (2010). An overview of traffic sign detection methods. Department of Electronics, Microelectronics, Computer and Intelligent Systems Faculty of Electrical Engineering and Computing Unska, 3, 10000.

Torgashov, P. (2011). Contour Analysis for Image Recognition in C#.