ADES: AUTOMATIC DRIVER EVALUATION SYSTEM

by

Kemal Kaplan

B.S. in Computer Engineering, Boğaziçi University, 2000

M.S. in Computer Engineering, Boğaziçi University, 2003

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2011

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Professor H. Levent Akın, who has consistently inspired me in this study and provided me precious advice and suggestions. Without his guidance and never ending patience, this thesis would not have been possible to accomplish.

I am also indebted to the members of my committee, Assoc. Prof. Tankut Acarman, Prof. Gökmen Ergün, Prof. Cem Say, and Prof. Oğuz Tosun. I would like to express my great appreciation for their insightful comments and guidance throughout the development of my dissertation.

I would like to thank to the past and present members of the Robotics Group of Boğaziçi University Artificial Intelligence Laboratory for their contributions to this thesis. It has been a great honor and privilege for me to work with them.

I dedicate my thesis to my family, especially my wife, Meltem, for her consistent support, love and for the life we experience together, both in our good times and hard times. I also thank to my parents and brother for their endless patience and support during this long journey.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the project.

ABSTRACT

ADES: AUTOMATIC DRIVER EVALUATION SYSTEM

Most of the traffic accidents occurred in the world and in our country are caused by the drivers. ADES (Automatic Driver Evaluation System) project targets to present a framework for integrating different applications for driver evaluation purpose. The proposed system can be divided into two main modules. The first one, which is the data acquisition and processing module, acquires the sensor information from the outside world and processed this data to present valuable information to the decision system. The system may benefit from built-in sensors like cameras or GPS (Global Positioning System) systems as well as non standard devices like RFID (Radio Frequency Identification) readers. The second module is the inference engine, which processes the information provided by the first module and makes judgments about the actions of the driver. Two sample expert system designs are proposed in the project. The developed solution is tested in simulation environment and by using real video recordings.

ÖZET

ADES: OTOMATİK SÜRÜCÜ DEĞERLENDİRME SİSTEMİ

Günümüzde ülkemizde ve dünyada meydana gelen trafik kazalarının büyük bir kısmı sürücü hatalarından kaynaklanmaktadır. ADES (Otomatik Sürücü Değerlendirme Projesi) projesinin amacı, sürücüler tarafından yapılan trafik kural ihlalerinin araç içerisinde bulunan bir cihaz tarafından algılanmasını sağlamak amacı ile geliştirilecek olan uygulamalar için bir altyapı oluşturmaktır. Tasarlanan sistem temel olarak iki kısımda incelenebilir. İlk kısım araç içerisinde bulunan çeşitli algılayıcılar tarafından edinilen verileri işleyip bu verilerden değerli bilgiler çıkaran uygulamaları kapsamaktadır. Bu algılayıcılar GPS, kamera gibi yeni nesil araçlarda bulunan cihazların yanı sıra RFID okuyucuları gibi araç içerisine sonradan eklenebilecek cihazlardan da oluşabilmektedir. İkinci kısım ise bu algılayıcılardan gelen bilgileri kullanarak sürücünün davranışlarını değerlendirmekle görevli olan çıkarım motorudur. Proje kapsamında bu amaç için tasarlanmış iki uzman sistem örneği verilmiştir. Tasarlanan çözüm gerçek kamera kayıtları üzerinde ve gelişmiş bir benzetim ortamında test edilmiş ve sonuçlar incelenmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATIONS

AAAI	Association for the Advancement of Artificial Intelligence
ADAS	Advanced Driver Assistance Systems
ADES	Automatic Driver Evaluation System
BN	Belief Network
CAN	Controller Area Network
DfC	Distance from Center
DtB	Distance to Borders
ES	Evolution Strategies
EU	European Union
FPS	Frames per Second
GA	Genetic Algorithm
GIS	Geographic Information System
GLONASS	The Global Navigation Satellite System
GNP	Gross National Product
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
HMM	Hidden Markov Process
HOV	High Occupancy Vehicle
HSL	Hue-Saturation-Luminance
IEEE	Institute of Electrical and Electronics Engineers
LDA	Linear Discriminant Analysis
LIDAR	Light Detection and Ranging
MHT	Multiresolution Hough Transformation
MP	Matching pursuit
MSBNx	Microsoft Bayesian Network Editor
NN	Neural Network
RADAR	Radio Detection and Ranging
RBF	radial basis kernel function
RF	Radio Frequency

RFID	Radio Frequency Identification
SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features
SVM	Support Vector Machines
TCP/IP	Transmission Control Protocol/Internet Protocol
UE	Unreal Engine
UPIS	USARSim Image Server
USARSim	Unified System for Automation and Robot Simulation
XML	eXtensible Markup Language

1. INTRODUCTION

Traffic accidents are one of the main causes of death and economic loss in the world. Road traffic crashes kill 1.2 million people a year or on the average more than 3000 people every day. In addition, traffic accidents injure or disable between 20 million and 50 million people a year [1]. In Europe, according to the *Road Safety Action Programme* of European Commission, more than one million accidents a year cause more than 40 000 deaths and nearly two million injuries on the roads. In addition, the direct and indirect cost has been estimated at 160 billion euros, which is nearly two percent of the EU's GNP [2]. The number of road accidents in Turkey was approximately half million in 1998, where this number exceeds 720,000 in 2006. In those accidents, more than 150,000 people injured, and unfortunately, nearly 5000 people lost their lives [3]. Unfortunately, the number of accidents and the number of people suffered from these accidents are increasing every year as shown in Table 1.1 [4, 5]. Worse still, the increasing number of permanently or temporarily suspended driver licenses failed to reduce these numbers.

A previous study about the causes of the traffic accidents states that most of the accidents are caused by driver actions [6]. The other causes are the roadway and the vehicle conditions as shown in Figure 1.1.

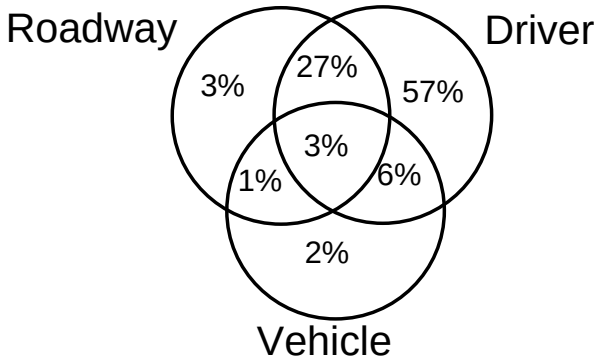


Figure 1.1. Causes of traffic accidents.

Table 1.1. The number of traffic accident and casualties in Turkey for the last decade.

Year	Accidents	Deaths	Injuries	License Suspensions
2001	442.960	4.386	116.202	105.828
2002	439.958	4.169	116.045	88.619
2003	455.637	3.959	117.551	86.084
2004	537.352	4.427	136.437	95.932
2005	620.789	4.505	154.086	78.170
2006	728.755	4.633	169.080	86.178
2007	825.561	5.007	189.057	110.560
2008	950.120	4.236	184.468	139.432
2009	1.053.346	4.324	201.380	138.619
2010	1.104.388	4.045	211.496	163.572

1.1. Motivation

Although the vehicle manufacturers deploy more intelligence in their newest models, the current applications are usually focused on driver assistance and early warning systems. However, in the near future, it is expected that, intelligent vehicles will also enforce the traffic regulations. For example, speed limit and traffic signal violations are going to be detected by cars. With this motivation, the ADES project is designed to be a framework for evaluating the drivers' obedience to the traffic rules. The applications of the resulting system include but not limited to the following items;

- Assist drivers in driving more safely: The artifacts of the proposed system can be used to inform the driver about the current traffic regulations that exist on the way.
- Report to the traffic central: The detected violations can be reported to the traffic central for further inspection.
- Automate driving license examinations: The system may give an evaluation result for the driver after a specific test period, which can be planned as a part of a driving license examination.
- Maintain inventory of the traffic signs on highways: The sign detection and recognition

module of the proposed system may be used to maintain the inventory of the traffic signs.

- Provide supervision for the development of the autonomous urban driving systems: The proposed system can give supervisory feedback for autonomous driving applications especially in urban environments.
- Augmented reality applications based on traffic signs or lanes: The detected road conditions can be integrated to the head up display of the vehicle for providing more information about the road to the driver.

1.2. Problem Statement

The focus of this work is in developing a system which can be used for evaluating the actions of vehicle drivers especially in urban environments. Most of the accidents can be avoided if the users become more sensitive about the traffic rules. However, it is not feasible to place a police at each corner of the smallest city. Therefore, automation of driver evaluation is inevitable for enforcing the obedience to the traffic rules.

Similar to the structure of the designed solution, the statement of the problem can be divided into two major categories. The first category is about the acquisition of the related data as quickly, cheaply and accurately as possible. However, there are numerous difficulties even for the basic sensors. Naturally, the more advanced the sensors are, the more complicated the problems become. For example, the RFID readers may suffer from tag collision (or data collision, i.e. multiple RF tags exist in the reading distance of the RF reader), which is relatively easy to resolve [7]. However, in vision, the system may suffer from lightning conditions or unrecognizable objects which requires more complicated solutions [8].

The second category of the problem is related with designing the inference engine. Even if the data acquisition modules provide the most accurate knowledge, the system may become too complicated to manage with simple if-then rules. Moreover, usually the sensors and data interpreting modules usually introduce considerably large amount of uncertainty to their outputs. As a result, the inference engine should be able to cope with both the complexity and the uncertainty of the environment [9].

In addition to these scientific difficulties, there are also some engineering problems. The system should be able to process a large amount of data with limited computational power. Additionally, the quality of the acquired data is closely related with the selected equipment. Moreover, the proper placement of the sensors on the vehicle may affect the system performance dramatically. The system may also require integration with different systems like existing traffic control agents or the vehicle itself. The cost of the sensors and additional peripherals, like RFID tags, should also be considered carefully. Although the expected cost of a single RFID tag is relatively small, thousands of RFID tags may become the most expensive part of the entire system. If the RFID tags are placed on vehicles, then the readers should be placed near the traffic signs. Otherwise all vehicles should be equipped with RF readers and traffic signs should have RFID tags which identifies them. In each case the number of required RF readers and RFID tags are quite large even for deploying the system in a small town. Since the RF readers are more expensive and require more maintenance, they may be placed in cars where it is relatively easy to obtain power source and keep the device under control. In addition, the cost of the reader may be charged to the vehicle manufacturers and can be subsidized by tax reductions.

1.3. Scope

The aim of this work is to develop a framework to reduce the rate of the accidents based on driver faults by detecting and reporting the traffic violations. In order to be more specific about the goals of the project, the scope of the proposed solution is defined for different aspects as follows.

- Targeted traffic rules: Only the subset of the traffic violations, which can be detected with the available sensors, are considered.
- Sensors: Although there are many sensors available in the market, the proposed solution is based on camera, RFID reader, GPS and the information provided by the vehicle itself.
- Implementations: The number of possible implementations for the proposed modules are numerous. Therefore, we have to limit the scope of this study with specific implementations for proposed modules. For example, we implemented the most common

used methods in the literature, which are neural networks (NN) and support vector machines (SVM), for sign recognition. Similarly, we presented two inference engine implementations, which are Prolog based and Belief Network based expert systems for comparing the rule based and probabilistic models.

- Interactions: The interactions between the actors of the traffic, like vehicle to vehicle, or vehicle to infrastructure communication, is left out of the scope of this study.

The end-to-end application presented at the end of the thesis encapsulates the main idea of the project and demonstrates the usage of the proposed framework. Most of the improvements for these limitations are discussed as the recommendations for future work, because of the available resources of this thesis work.

1.4. Thesis Outline

In the first section of Chapter 2, the ADAS technologies are introduced and related works are presented. The expert system implementations used in the intelligent vehicles are investigated in the second section.

The system overview for the solution design is given in the first section of Chapter 3. The proposed approach contains two main sections. The first section discusses the data acquisition from different types of sensors, and the next section examines the requirements for the expert systems which will be developed for the inference engine.

In Chapter 4 the details of the solution for the problem are given. The first section of this chapter focuses on the implementation of the vision module which is responsible for gathering information from raw camera images. The next chapter explains the basics and challenges of the radio frequency identification technology and the implementation used in the project. The following section about the GPS/GIS and vehicle integration details the employment of these technologies in the ADES project. The details of the expert system designs are also given in the following section. Finally the driver aggressiveness and its application in the project are discussed in the last section. In Chapter 5, the computer applications developed for the ADES project are introduced. In Section 5.1 the details of ADES Detector, which is used for training and testing vision algorithms by using different kind of image sources, are given. The ADES

simulation environment which is build on Unreal Engine and USARSim are explained in the subsequent section.

In Chapter 6, the evaluation of each module is discussed and comparisons with alternative methods are presented. The conclusions and the improvement opportunities are given in Chapter 7.

2. RELATED WORK

Although, to the best of our knowledge, there is no significant study available concerning the autonomous driver evaluation by the means of on vehicle systems, there are numerous works regarding autonomous driving, driver assistant systems, and autonomous traffic control systems. These studies dealt with many common problems and have proposed several remarkable solutions which are also applicable in this problem domain.

2.1. Current Advanced Driver Assistance Systems Technologies

The proposed system will benefit from different kinds of Advanced Driver Assistance Systems (ADAS) to detect and report the traffic violations. The state-of-the-art ADAS implementations include the following;

- Adaptive cruise control [10]
- Forward collision warning [11]
- Lane Departure Warning and Blind Spot Detection [12, 13]
- Speed Limit Monitoring [14]
- Driver Drowsiness Detection [15]

and there are many more systems on development like vehicle to vehicle or vehicle to infrastructure interaction systems [16]. ADES project is a framework where these driving assisting systems can be utilized to evaluate the drivers' traffic violations. A lane detection system and a traffic sign recognition module are developed in the proposed solution.

2.1.1. Lane Detection Techniques

For lane detection, Hough Transform [17] is one of the most common techniques [18, 19]. However, there are many other techniques in the literature for lane detection. Pomerleau *et al* [20] used neural networks in their ALVIN system. Dynamic programming is used for eliminating outliers from detected line segments by Kang et al [21]. In addition, Wang *et al* [22] used B-splines in order to fit lane markings. Different techniques have also been

proposed for tracking the detected lanes and modeling the road. Kalman filtering [23] and particle filtering [24, 25] are the two most common tracking techniques used in lane tracking methods. A more detailed survey for lane detection strategies are provided by McCall and Trivedi [26].

2.1.2. Methods for the Detection of Traffic Signs

Numerous methods for the detection of traffic signs are proposed. Escalera *et al* [27] used color thresholding and shape analysis for sign detection. Fang *et al* [28] developed a neural network based approach for detection and Kalman Filter for tracking the signs. Hsu *et al* [29] proposed a template matching based for detection and matching pursuit filters, which decompose patterns into two dimensional wavelet expansions, for recognition. Bahlmann *et al* [30] used Haar wavelet features obtained from Ada-Boost training for detection and LDA followed by maximum likelihood approach for recognition. Loy *et al* [31] modified the radial symmetry transform for sign detection. Bascon *et al* [32] used shape classification based on linear SVMs for the detection phase. H. Fleyeh [33] proposed a fuzzy approach for color detection and segmentation of traffic signs. Parada-Loira *et al* [34] defined a new operator named as Local Contour Patterns and used it in fast Hough-Transform-based approaches for circle and line detectors. A Scale-Invariant Feature Transform (SIFT) based approach is proposed by Hoferlin *et al* [35]. Local SIFT features are used for content-based traffic sign detection along with widely applied shape-based approaches. Escalera *et al* [36] and Soetedjo *et al* proposed Genetic Algorithm (GA) based methods for sign detection. Although the sign detection phase in this thesis also uses GA, a novel approach for the encoding of individuals and a suitable fitness function are proposed.

2.1.3. Methods for Traffic Sign Recognition

One of the early studies on the topic of traffic sign recognition was introduced by Escalera *et al* [27] in 1997 where classification was done by neural networks. Two separate multilayer perceptron NNs have been trained for triangular and circular signs. The size of the input layer corresponds to an image of 30x30 pixels, and the output layer is of size ten, i.e., nine sign types plus one output that shows that the sign is not one of the nine. Hsu and

Huang [29] used matching pursuit (MP) filter [37], which decompose patterns into two dimensional wavelet expansions, to recognize the road signs effectively. MP algorithm uses a greedy heuristic to iteratively decompose any signal into a linear expansion of waveforms that are selected from a redundant dictionary of functions. Matching pursuits are general procedures to compute adaptive signal representations. An SVM-based study introduced by Maldonado *et al* [32] employed Gaussian-kernel SVMs where the input to the recognition stage is a block of 31x31 pixels in grayscale image for every candidate blob. In order to reduce the feature vectors, only those pixels that must be a part of the sign (pixels of interest) are used. Another SVM-based solution by Kiran *et al* [38] introduces an SVM Learning technique for traffic sign classification where the classification performance is improved by training the SVM using novel features called *distance from center* (DfC) and *distance to borders* (DtB). These features generates blob relative vectors which make the system invariant of translation, rotation and scale factors. Soetedjo and Yamada [39] have focused on traffic sign classification using Ring Partitioned Method on grayscale images which uses partitioned histograms to classify signs. Miura *et al* [40] have used two cameras to recognize the traffic signs. A telephoto camera, which can change the viewing direction, is directed to the region of interest and it captures a closer view of the candidate signs. The classification is achieved by a normalized correlation-based pattern matching technique using a traffic sign image database. Another work by Garcia-Garrido *et al* [41] also used neural network for classification where the input is a 32x32 pixel-size normalized image of the candidate sign. Finally, in a very recent study Ruta *et al* [42] introduced a novel feature selection algorithm that extracts for each sign a small number of critical local image regions having the highest dissimilarity between the candidate and the other signs.

2.2. Expert Systems Used in Intelligent Vehicles

In order to propose a suitable reasoning method for the ADES project, different planning techniques used in autonomous vehicles have been investigated. Sukthankar *et al* [43] presented a distributed solution, which consists of a collection of reasoning objects that votes upon a set of possible actions for dealing with the complexity of tactical reasoning. Rosa *et al* [44] implemented a fuzzy expert system to organize traffic regulations in town area. Al-Shihabi *et al* [45] proposed a framework for modeling driver behavior which has perception,

emotions, decision-making, and the decision-implementation units formed of fuzzy variables and if-then rules. Gao and Zhou [46] developed a fuzzy rule based control strategy selection approach for mobile navigation which utilizes Fuzzy Reasoning Petri Nets for system modeling and parallel reasoning. Lattner *et al* [47] introduced a knowledge-based approach to behavior decision for intelligent vehicles which uses qualitative representation of the information perceived by the sensors. Long *et al* [48] presented a review of software platforms for autonomous vehicles and their artificial intelligence development capabilities. Ferguson *et al* [49] presented a reasoning framework, which consists of the mission, behavioral, and motion planning components perform the reasoning for an autonomous vehicle navigating through urban environments. Vacek *et al* [50] used case-based reasoning in order to predict the progress of the current situation and to select the appropriate behavior. Toit *et al* [51] described a finite state machine model to manage the complexity of the situational reasoning subsystem which recognizes the current situation to impose the correct traffic rules.

The BN's have been studied in researches related with autonomous driving or driver assistant systems for more than a decade. In 1995, Forbes *et al* [52] proposed the *Bayesian Automated Taxi (BATmobile)* which is a decision-theoretic architecture using dynamic probabilistic networks for autonomous driving. The BATmobile used dynamic Bayesian networks where the states of the variables are also introduced to the decision process. In addition to having Markovian property, the proposed network also modifies its internal structure by adding or removing the time slices. In 2002, Ross *et al* [53] described a distributed information fusion system based on hierarchical BN's using D'Agent mobile agent system. In the proposed system the structure of the belief network can be rearranged according to the gathered information. Kumagai *et al* [54] also used dynamic Bayesian networks to predict the driver's stop behavior. The structure and the parameters of the network are calculated by using the pedal strokes and the speed of the vehicle during the learning phase. Dagli *et al* [55] also used a similar technique for predicting the lane departure behaviors of the drivers. In 2008, Petrovskaya *et al* [56] used a dynamic Bayesian network model for tracking the vehicles during DARPA urban challenge. The positions and the orientations of the vehicles are used for predicting the future location of the vehicles. Numerous examples of BN's in autonomous driving can be presented since the idea behind the time dependent dynamic Bayesian networks is quite similar with Markovian Models and Kalman Filters.

3. PROPOSED APPROACH

Unlike the basic definition of the problem, which is revealing the faulty driver, it is not very easy to propose a fully detailed approach for this purpose. It is clear that any proposed solution should be aware of the actions of the driver and be able to judge the decisions of the driver. However, there are many alternative ways to accomplish every single sub problem some of which are summarized in the previous chapter.

3.1. System Overview

In the ADES project, the task is divided into two major parts. The first part acquires the sensor data and processes it for the use of the inference engine. The second part, which is the inference engine, uses the sensor values as facts, and comes up with the final decision according to its rules in the knowledge base. Since the rules in the knowledge base will be derived from common traffic rules for this application, they are going to be introduced by human experts. As a result, the inputs of the system are sensor values and predefined expert rules, and the output is the evaluation of the driver's moves as shown in Figure 3.1.

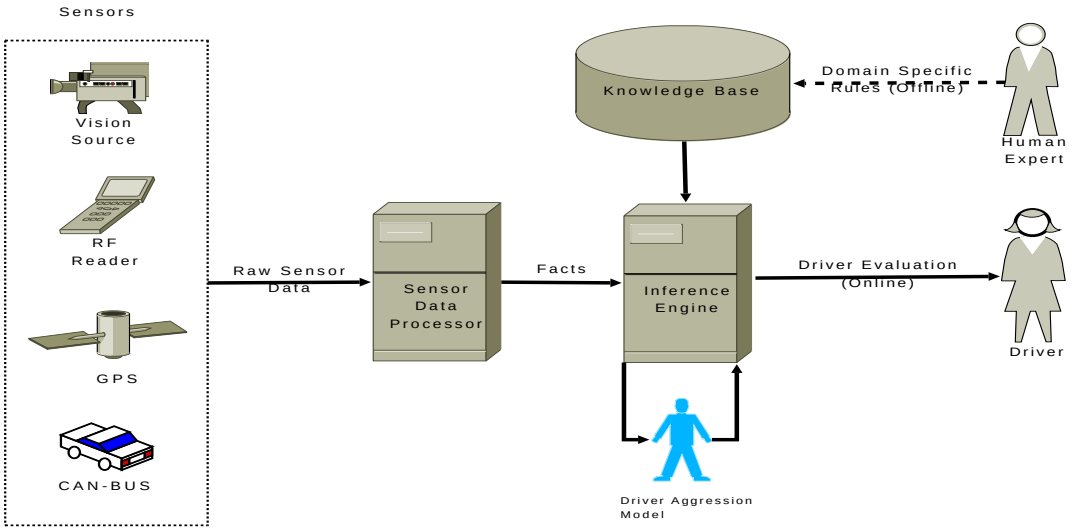


Figure 3.1. Basic system architecture of ADES project.

3.2. Data Acquisition and Processing

The most common sensors for mobile agents are video cameras, global navigation satellite systems, light and radio waves based detection and ranging systems. The proposed approach utilizes a single video camera which is placed in the front of the vehicle where the road and other important objects can be seen easily. The RFID technology also inspected in order to reduce the possibility of the errors and increase the reliability of the system. In addition to these modules, there are other commonly used devices for gathering position data and communicating with the vehicle itself.

3.2.1. Vision Module

Since the vision sensors usually provide valuable information about the environment, they are inevitable for most of the autonomous systems. ADES project based on a mono vision sensor, which is a video camera placed on the vehicle. The goal of the vision module is not only capture the image of the environment, but also process this image in order to provide valuable information for the inference engine. The proposed vision processing subsystem will be able to detect and track lane markings and traffic signs. However there are ongoing research projects for detecting traffic lights and various obstacles like other vehicles and pedestrians [57, 58].

The set of images includes the defined traffic signs shown in Figure 3.2. The circular ones are regulatory and the triangular ones are the warning signs in Europe [59]. The proposed approach benefits from the red borders which is the most distinct common property of these signs. However, the proposed detection method can be used for different traffic signs in other countries with appropriate fitness functions.

Figure 3.2. Traffic signs recognized by the proposed system.

3.2.2. RF Module

In the proposed solution the RF reader is placed within the vehicle due to power requirements. Since the required know-how and resources are beyond the scope of this project, the physical implementation is left as a future work. However, the RF reader and RFID tags are successfully modeled in the simulation environment.

3.2.3. Other Sensor Technologies

There are also other sensors in autonomous driving. One of the most common sensor is RADAR which is a system that uses electromagnetic waves to identify the range, altitude, direction, or speed of both moving and fixed objects. However, more recent applications use the laser scanning radar sensor LIDAR, which measures the relative position of the controlled vehicle with respect to its preceding vehicle. In addition to ranging sensors there are Global Navigation Satellite Systems (GNSS) which uses a version of triangulation to locate the receiver, through calculations involving information from a number of satellites. The most popular GNSS is probably the NAVSTAR GPS, maintained in the United States. There are other systems using satellites as references in the navigation process. Galileo is one GNSS system under development by the European Union (EU). Furthermore, Russia has a system under development called GLONASS [60].

3.3. Inference Engine

The proposed inference system is designed to reason for specific traffic violations. The targeted rules and their fines [61] are given Table 3.1. Although the proposed system is focused on the previous rules, it can also be used for detecting the violations given in Table 3.2 with necessary improvements like obstacle recognition via vision module or cross-road awareness by GPS.

In the current state of the project, there are two expert system implementations which are based on Prolog and Belief Networks. However, any expert system implementation which satisfies specific functionalities, which are detailed in Section 4.4.1, can be integrated easily

Table 3.1. Targeted traffic rules.

Violation	**Fine**
Violating the red traffic light	140 TL
Exceeding speed limits from 10 up to 30 percent	140 TL
Exceeding speed limits more than 30 percent	290 TL
Entering forbidden roads.	140 TL
Not obeying the signs and land markings	66 TL

to the system.

Table 3.2. Other relevant traffic rules.

Violation	**Fine**
Not following the rightmost lane unless there is an overriding sign	140 TL
Following the leading vehicle from an unsafe distance	66 TL
Not slowing down while entering crossroads, passing hills, crosswalks	66 TL
Driving too slow or decelerating unexpectedly	66 TL
Entering the crossroad and blocking the traffic	66 TL
Not obeying the STOP sign on the school buses	140 TL

4. METHODOLOGY

The realization of the proposed solution requires implementation and integration of several modules as described in the following sections.

4.1. Vision Module

Vision module is responsible for acquiring raw image data from the camera and processing it to provide information to the expert system implementation. The current implementation is capable of detecting lane markings and specific traffic signs. The overall process is shown in Figure 4.1. The entire process is triggered by the arrival of the raw RGB image data. The received image bytes are duplicated for parallel running processes. The first image is preprocessed by a color remapping function to indicate the white and yellow lane markings in the binarized image. The lane detection process then uses this binarized image to find the road lanes. The image binarization process for the sign detection process, however considers the red pixels in the image. The binarized image is duplicated again for the circular and triangular file detection processes. After the location of the traffic sign is detected, the clipped portion of the image is transferred to the predefined sign recognition processor. Currently two sign classification methods are implemented. One of the SVM and NN based sign recognition processors can be selected as desired. The performance of both methods are discussed in Chapter 6. Although the current implementation of these classifiers are using grid based features, a Speeded Up Robust Features (SURF) [62] based feature extraction mechanism is also implemented. The performances of these systems are discussed in the evaluation section.

4.1.1. Image Binarization

The performance of the image binarization process has a dramatic effect on the overall system performance since the remaining processes run on the selected white pixels. If these pixels are not selected properly or incorrectly mapped, the failure of the entire system is guaranteed. In addition, due to the complexity of the detection processes, the binarization process should be very fast for real time applications. Therefore, fast binarization processes

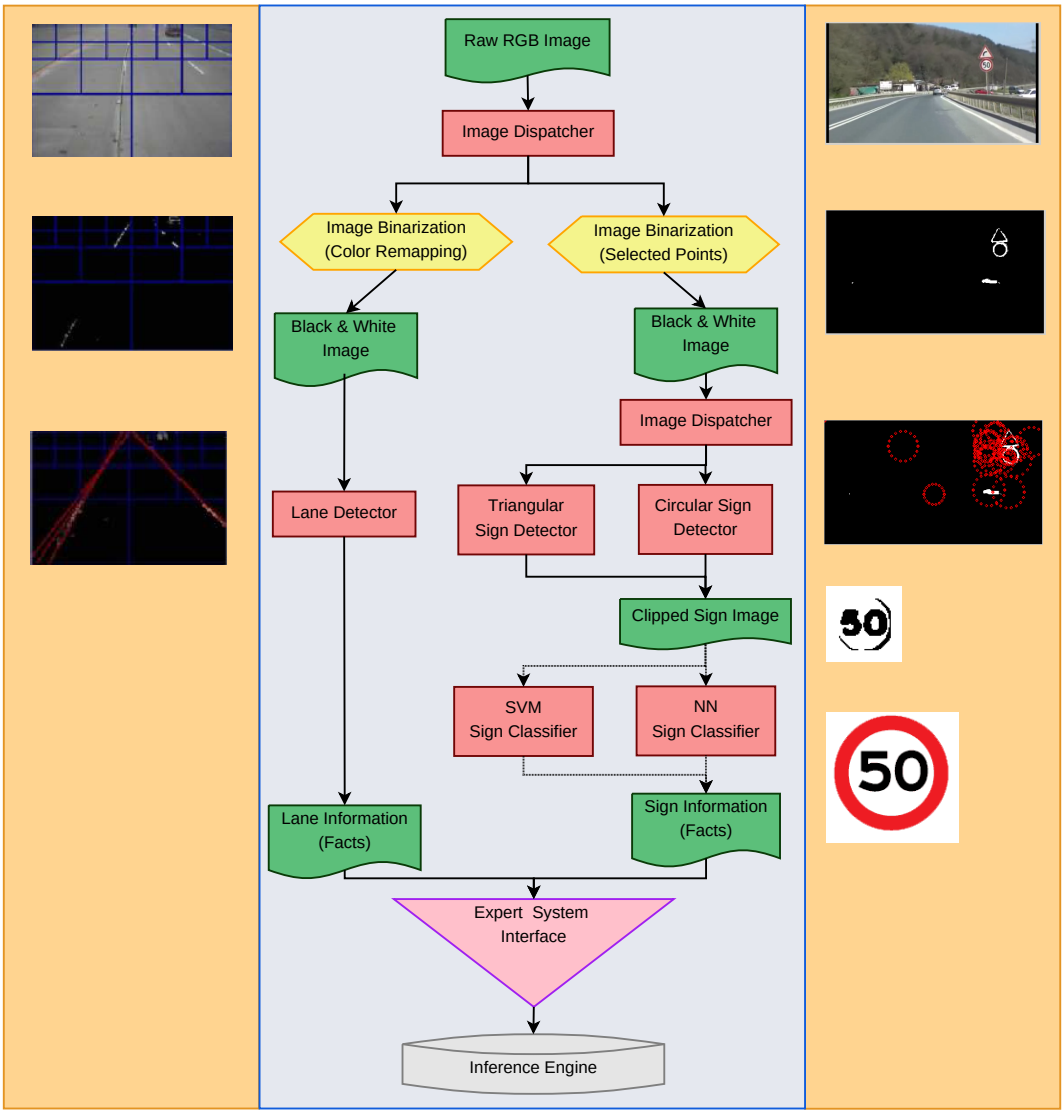


Figure 4.1. Vision processing.

with simple but effective mapping algorithms are proposed in this study.

4.1.1.1. Color Remapping. As the first step of the lane detection process, the image is converted to a binary image by using a color remapping function. The mapping for each pixel from the 24bit RGB value to a binary value is given in Table 4.1. If each mapped value of the red, green, and blue components of the selected pixel is one, then this pixel is a white one in the resulting image. This binarization favors the white and yellow parts of the images. The values have been empirically determined for the video sample. However, for realistic applications an adaptive scenario should be used to accommodate changing lightning conditions.

Table 4.1. Color remapping.

Pixel Value	Red	Green	Blue
0-176	0	0	0
176-196	1	1	0
196-255	1	1	1

4.1.1.2. Adaptive Image Binarization. There are two main image binarization methods proposed for improving the sign detection performance. Both schema try to differentiate the red regions of the image by assigning the most proper values for the coefficients α and β in Equation 4.1.

$$\begin{aligned} f(r,g,b) &= \begin{cases} 1 \rightarrow r > \alpha \times g, r > \beta \times b \\ 0 \rightarrow o/w \end{cases} \\ \alpha &> 1 \\ \beta &> 1 \end{aligned} \tag{4.1}$$

The mean based image binarization uses the histogram calculations of the acquired frame. Sample scenes with corresponding red, green, blue histograms for 24bit RGB images are given in Figure 4.2. The values of α and β in Equation 4.1 are calculated according to Equation 4.2 where HSL denotes the hue, saturation and lightness histogram arrays.

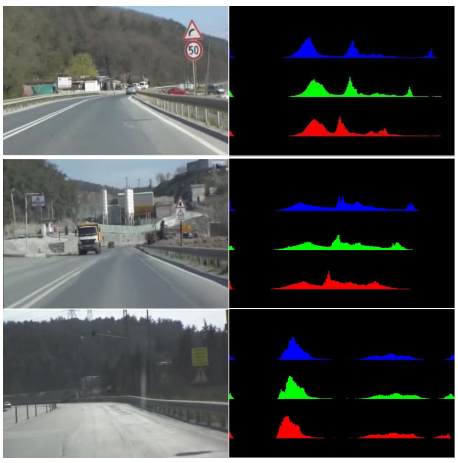


Figure 4.2. Good, medium and poor conditions for traffic sign detection with corresponding red, green, and blue histograms.

$$\begin{aligned} HIST &= Histogram(HSL_{rgb}) \\ \alpha = \beta &= 1 + \frac{L_{mean}}{2} \end{aligned} \tag{4.2}$$

However, for sign detection process, the red pixels should be considered. Therefore, in the second approach, the coefficients' values are adjusted according to the total number of white pixels in the binarized image. If the number of the white pixels is less than a predefined value then the coefficients are slightly relaxed. On the opposite side, if there are too many white pixels, then the coefficients are tightened. Since the number of white pixels in the binarized image represents the red-most pixels in the original image, this method is focused on the red pixels of the image.

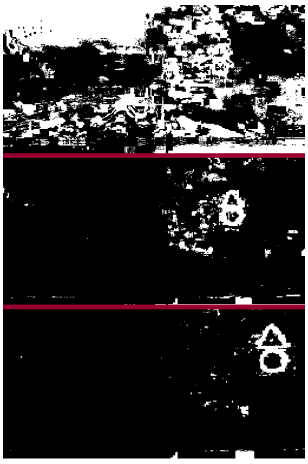


Figure 4.3. Effect of selected point count on image binarization.

The process can be formulized by Equation 4.3 where p is a pixel on the binarized image. The values of α' and β' are usually assigned slightly higher than one (e.g 1.05) to prevent major changes in the consecutive frames. However, the effect of the adaptation is not reduced because of the high frame rate of the streaming media. In Figure 4.3 the sign becomes clearer as the α and β values become 1.02, 1.04, and 1.104 respectively.

$$
\begin{aligned}
p_i &= \begin{cases} 1 \rightarrow p = white \\ 0 \rightarrow p = black \end{cases} \\
p_{white} &= \sum p_i, \forall p \in image \\
\alpha &= \begin{cases} \alpha \times \alpha' \rightarrow p_{white} > max_{white} \\ \frac{\alpha}{\alpha'} \rightarrow p_{white} < min_{white} & , \alpha' > 1 \\ \alpha \rightarrow o/w \end{cases} \\
\beta &= \begin{cases} \beta \times \beta' \rightarrow p_{white} > max_{white} \\ \frac{\beta}{\beta'} \rightarrow p_{white} < min_{white} & , \beta' > 1 \\ \beta \rightarrow o/w \end{cases}
\end{aligned}
\tag{4.3}
$$

4.1.2. Road Lane Detection and Tracking

The classical Hough transformation approach processes the entire vision data in order to detect the lines. This scenario has two main drawbacks. First, the occluded lines (i.e. another car passing through the line) become noisy since the transformed relative intensity of the line decreases. Second, the relative intensity of the lines also decreases at the curves in the road. The proposed solution divides the road image into partitions, where the sizes of the partitions are inversely proportional to the distance of the partition to the vehicle.

The proposed approach employs Multiresolution Hough Transformation (MHT) for lane detection, followed by two Hidden Markov Process (HMM) models for radius and orientation of the candidate lanes [8]. After the image is partitioned, a separate Hough transformation is applied to each single partition as shown in Figure 4.4. The most intense line in each partition, which is the candidate line segment, is taken into consideration in order to find the global lanes in the image. Since the Hough lines are represented in polar coordinates *(r, θ)* instead of rectangular coordinates *(x, y)*, the candidate lines are grouped according to their slopes and distances to the center of the image as well as their intensities. The center of the frame is chosen as the bottom most pixel. The transformation of the lines basically changes the center

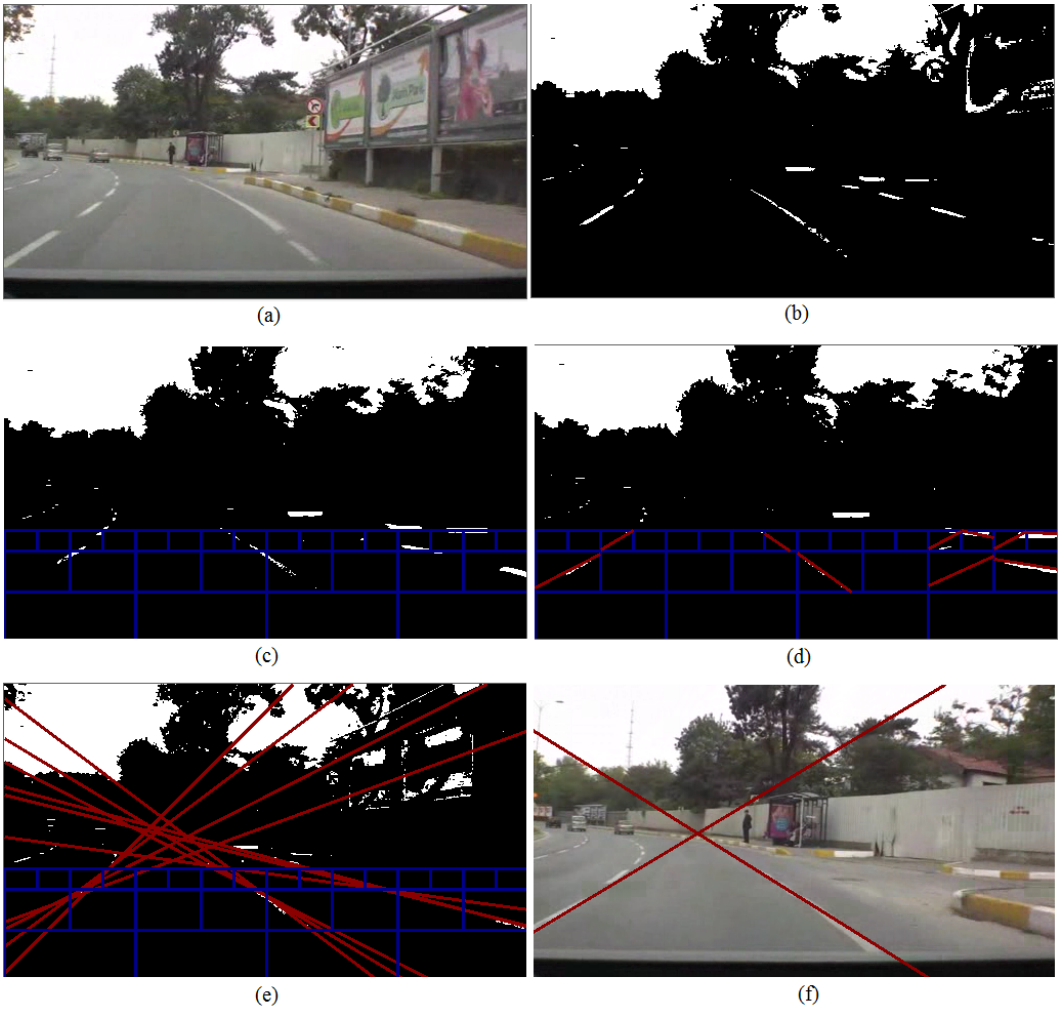


Figure 4.4. (a) Original image, (b) Binarized image, (c) Partitions, (d) Hough line segments in partitions, (e) Transformed candidates, (f) Detected lane markings.

point of the polar coordinates for each transformed line which is achieved by Equation 4.4

$$\begin{aligned} r' &= r + (x - x')\cos(\theta) + (y - y')\sin(\theta) \\ \theta' &= \theta \end{aligned} \tag{4.4}$$

where *(r', θ')* is the polar coordinates of the transformed Hough line *(r, θ)*. Note that the translation of the center of the Hough transformation is from *(x, y)* to *(x', y')*. After the lines are grouped, the most intense three clusters are assigned as the lanes. However, there may be less than three lanes if the sum of the intensities of the candidate lines is less than a threshold value.

For lane tracking, HMM [63] is used to represent the relation between the current

Table 4.2. Properties of video sequence.

Camera Position:	Front console of the car.
Resolution:	512x288
Frame Rate:	29.97
Length:	34 sec.

frame and its successor. Each line in a specific frame is represented by an individual *(r, θ)* pair. In the succeeding frame, the process will most probably observe the same line at *(r', θ')* which is not very far from the position of the line in the previous frame. The probability of observing *(r', θ')* pair in the next frame is modeled as an HMM problem. In addition, θ and r values are modeled by two different HMM. The θ value is discretized as (0, 1, 2, 3...178, 179) where the *r* value is discretized at the pixel level. This discretization schema is used for the both transmission and emission matrices. The emission probability matrix shows the probability of observing θ' (or *r'*) in the next frame, having observed θ (or *r*) in the current frame. In our implementation, the observation and state transition matrix values are derived from two Gaussian distributions with different deviations. The deviation of the transition matrix is assigned to a smaller value than the observation matrix, which means, the state transition matrix aims to preserve the current state where the observation matrix promotes the exploration behavior.

4.1.2.1. Implementation Details. The first step of the implementation is for determining the partitions of the image on which the Hough transforms will be applied. Although the processed sample image is 288 pixels high, only the bottommost 116 pixels are used since the road remains in this lower part of the image. The accuracy of this assumption may slightly differ depending on the slope. The proposed approach is implemented and tested on a relatively short video sequence of an urban drive. In addition, the new approach is compared with the classical Hough transform where the entire image is processed and the most intense lines are accepted as candidate lines. The properties of the video sequence are given in Table 4.2.

The widths of the partitions are 32, 64, and 128 pixels from top to bottom, and the heights are 32, 42, and 42 pixels respectively. These values are assigned according to the position of the camera. After the partitions are calculated, Hough transformation is applied

Table 4.3. (a) Transmission matrix for r, (b) Transmission matrix for θ.

r	**0**	**1**	**2**	**3**	...	**279**	**280**	**281**	**282**
0	0,3989	0,2420	0,0540	0,0044	...	0,0000	0,0000	0,0000	0,0000
1	0,2420	0,3989	0,2420	0,0540	...	0,0000	0,0000	0,0000	0,0000
2	0,0540	0,2420	0,3989	0,2420	...	0,0000	0,0000	0,0000	0,0000
...	...	...	...	...	...	...	...	...	...
280	0,0000	0,0000	0,0000	0,0000	...	0,2420	0,3989	0,2420	0,0540
281	0,0000	0,0000	0,0000	0,0000	...	0,0540	0,2420	0,3989	0,2420
282	0,0000	0,0000	0,0000	0,0000	...	0,0044	0,0540	0,2420	0,3989

θ	**0**	**1**	**2**	**3**	...	**176**	**177**	**178**	**179**
0	0,3989	0,2420	0,0540	0,0044	...	0,0001	0,0044	0,0540	0,2420
1	0,2420	0,3989	0,2420	0,0540	...	0,0000	0,0001	0,0044	0,0540
2	0,0540	0,2420	0,3989	0,2420	...	0,0000	0,0000	0,0001	0,0044
...	...	...	...	...	...	...	...	...	...
177	0,0044	0,0001	0,0000	0,0000	...	0,2420	0,3989	0,2420	0,0540
178	0,0540	0,0044	0,0001	0,0000	...	0,0540	0,2420	0,3989	0,2420
179	0,2420	0,0540	0,0044	0,0001	...	0,0044	0,0540	0,2420	0,3989

to each partition as described in the previous section. The most promising three lines are assigned as the candidate lane markers. But there may be less than three lines if the intensity of the calculated lines are less than an empirically assigned threshold. The experiments show that the proposed approach usually detects only two lines most of the time.

After finding the lane markers, HMM method is used to track the lanes. The values of the emission and transition matrices are derived using Gaussian assumption. The deviation of the transition matrix is assigned as 1 and the deviation of the emission matrix is taken as 2. Two separate models are prepared for the θ and r values of the candidate lane markers. The transition and emission matrices are given in Tables 4.3 and 4.4. Since the θ values 0 and 179 are actually very close, the emission and transmission values are the same for 1 and 179 in θ matrices. In addition, the range of the r matrices is (0, 282) because the maximum possible distance for any detected line is 282 pixels where the height of the processed part of the image is 116 and width of the image is 512.

Table 4.4. (a) Emission matrix for *r*, (b) Emission matrix for θ.

r	**0**	**1**	**2**	**3**	**4**	**5**	**...**	**281**	**282**
0	0,1995	0,1760	0,1210	0,0648	0,0270	0,0088	...	0,0000	0,0000
1	0,1760	0,1995	0,1760	0,1210	0,0648	0,0270	...	0,0000	0,0000
2	0,1210	0,1760	0,1995	0,1760	0,1210	0,0648	...	0,0000	0,0000
3	0,0648	0,1210	0,1760	0,1995	0,1760	0,1210	...	0,0000	0,0000
4	0,0270	0,0648	0,1210	0,1760	0,1995	0,1760	...	0,0000	0,0000
...	...	...	...	...	...	...	...	...	...
281	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	...	0,1995	0,1760
282	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000	...	0,1760	0,1995

θ	**0**	**1**	**2**	**3**	**4**	**5**	**...**	**178**	**179**
0	0,1995	0,1760	0,1210	0,0648	0,0270	0,0088	...	0,1210	0,1760
1	0,1760	0,1995	0,1760	0,1210	0,0648	0,0270	...	0,0648	0,1210
2	0,1210	0,1760	0,1995	0,1760	0,1210	0,0648	...	0,0270	0,0648
3	0,0648	0,1210	0,1760	0,1995	0,1760	0,1210	...	0,0088	0,0270
4	0,0270	0,0648	0,1210	0,1760	0,1995	0,1760	...	0,0022	0,0088
...	...	...	...	...	...	...	...	...	...
178	0,1210	0,0648	0,0270	0,0088	0,0022	0,0004	...	0,1995	0,1760
179	0,1760	0,1210	0,0648	0,0270	0,0088	0,0022	...	0,1760	0,1995

4.1.3. Traffic Sign Detection and Tracking

The proposed approach for sign detection and tracking in the ADES project is based on GA and a modified version of radial symmetric transform after an image binarization.

The GA implementation uses the coefficients of a geometric transformation applied to a set of points which describes the characteristics of any searched template. The formulation of geometric transformation, which includes affine and perspective transformations is given in Equation 4.5

$$\begin{vmatrix} u' \\ v' \\ w \end{vmatrix} = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ 1 \end{vmatrix} \tag{4.5}$$

$$u = u'/w$$

$$v = v'/w$$

where x and y are the coordinates of the sample point from the template describing set of points. u and v are the transformed point on the image. a, b, d, e provides rotation, scaling and shearing where c and f used for translation. In addition, g and h provides perspective transformation in two dimensions. These coefficient values or a subset of them can be used in the chromosome encoding of the GA.

The effect of the transformation can be visualized better with a simple example. Assume that a, b, c, d, e, and f coefficients are used in the encoding of the GA chromosome. In addition, g and h are left zero for simplicity. Also assume that 12 points are used in representing the characteristics of a circular object. For this scenario, we can conclude that a chromosome with the transformation coefficients in Equation 4.6 can yield to the transformed circle points in Figure 4.5. The points on the left hand side of the figure are the characteristic points of the circular template, which are 12 equidistant points on the unit circle, and the right points are the translated, scaled, and rotated counter parties in the transformed domain.

$$\begin{vmatrix} u' \\ v' \\ 1 \end{vmatrix} = \begin{vmatrix} 2 & 1 & 100 \\ 1 & 2 & 50 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ 1 \end{vmatrix} \tag{4.6}$$

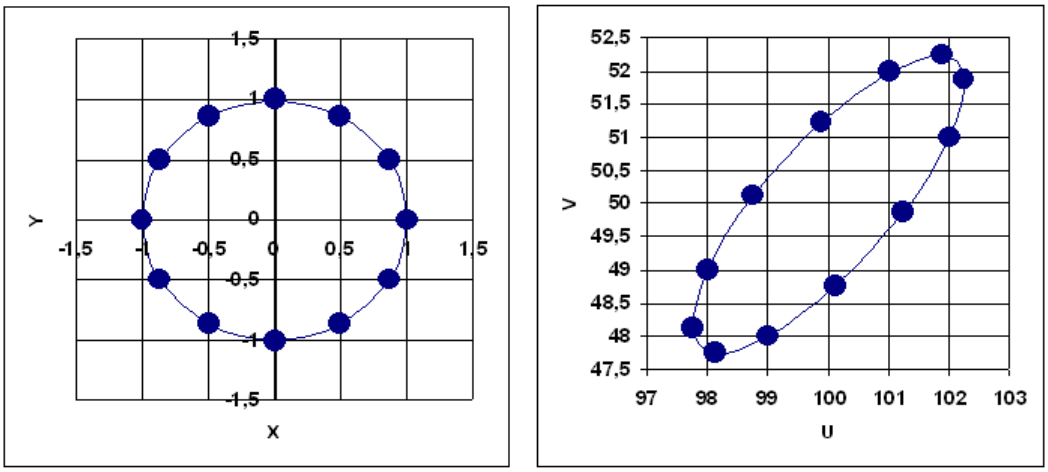


Figure 4.5. Template characteristic points in (x, y) domain, and (u, v) domain after the geometric transformation.

These new points are used to calculate the fitness of the candidate traffic sign location. The fitness value is formed as a function of the colors of the underlying pixels for each transformed point in the processed snapshot.

4.1.3.1. Implementation Details. Similar to the lane detection, the sign detection process also starts with an image binarization process. The second step is creating the population for the GA process. The chromosome encoding is based on the coefficients of the geometric transformation matrix as defined in problem statement section. However, for simplicity, only the two translation and one scaling coefficients are included in the chromosome. The resulting transition matrix is given in Equation 4.7.

$$\begin{vmatrix} u \\ v \\ 1 \end{vmatrix} = \begin{vmatrix} a & 0 & c \\ 0 & a & f \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ 1 \end{vmatrix} \tag{4.7}$$

The crossover process is also a function of these coefficient as given in Equation 4.8.

$$\begin{aligned} a_{newchromosome} &= \alpha.a_{chromosome1} + \beta.a_{chromosome2} \\ c_{newchromosome} &= \alpha.c_{chromosome1} + \beta.c_{chromosome2} \\ f_{newchromosome} &= \alpha.f_{chromosome1} + \beta.f_{chromosome2} \\ 1 &= \alpha + \beta \end{aligned} \tag{4.8}$$

Empirically determined parameters of the GA process are as follows,

- Population Size: 100
- Number of Iterations: 5
- Mutation Rate: 0.05
- Selection Rate: 0.9
- Selection Method: Elitist

In addition to these properties, half of the best chromosomes in a frame is transferred to the

next frame in order to provide an initial knowledge about the image. Therefore the number of iterations is kept small. The fitness of the chromosome is evaluated according to the color of the transformed point (u, v) on the binary image. If the value of the pixel is one which means it is a red point on the original image, the fitness of the chromosome is increased. However, this method fails for completely red regions, therefore another set of template points are introduced in order to indicate the non-red points on the template. These points are also subject to the transformation. In this implementation these non-red points are selected to place in the inside of the unit circle. For the example in the problem statement section, if we add the non-red points to the Figure 4.5 we end up with Figure 4.6 where the red points increase the fitness values when they are white in the binary image, and the black points increase the fitness value when they are black in the binary image. If the expected color cannot be found then the fitness is decreased for each failed point.

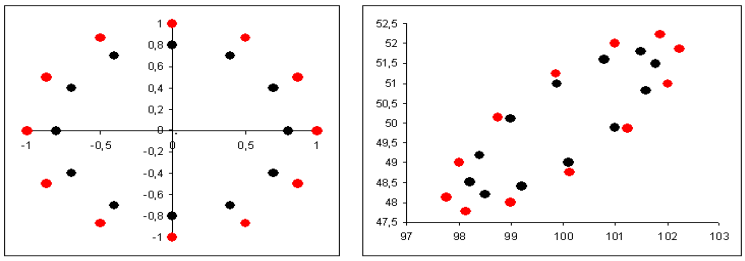


Figure 4.6. Red and non-red template points.

At each iteration the fitness values are calculated for each chromosome and at the end of the process the chromosomes are expected to converge around the circular sign as shown in Figure 4.7

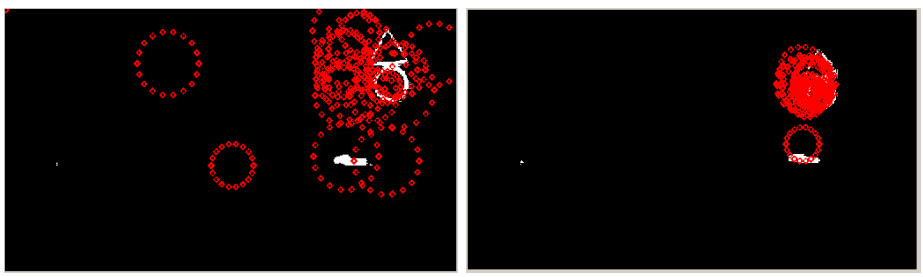


Figure 4.7. Initial and converged chromosomes.

Finally if the best chromosome has a fitness value greater than a threshold value, then

the points of this chromosome should be on a circular traffic sign on the original image as shown in Figure 4.8. This complete process takes nearly 50 milliseconds when it is executed on the same hardware specified in Chapter 6.

Figure 4.8. Detected traffic sign.

4.1.4. Traffic Sign Classification System

After the detection phase, the detected image should be classified to provide the sign information to the inference engine. In the ADES project, three different methods are proposed for this purpose.

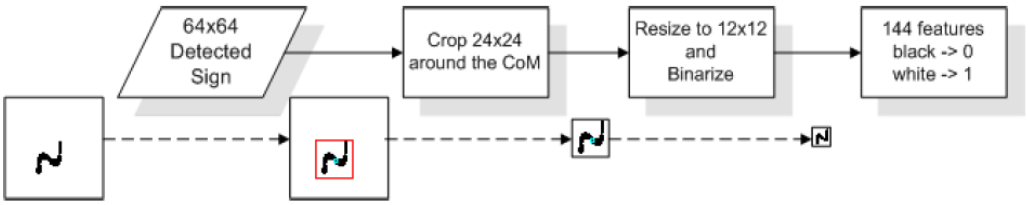


Figure 4.9. Input array formation for Grid based NN and SVM implementations.

4.1.4.1. Grid Based Neural Network Implementation. The detected sign is serialized to a binary input array with 144 items where each item is an input for the classifier network as shown in Figure 4.9. In the current implementation, each sign has its own network, i.e. the network trained in order to classify that sign, with 144 inputs, 4 hidden layers and one output layer. The networks are trained by using Levenberg-Marquardt learning technique [64] which based on solving Equation 4.9

$$(J^t J + \lambda I)\phi \quad = \quad J^t E \tag{4.9}$$

where J is the Jacobian matrix for the system, λ is the Levenberg's damping factor. ϕ is the weight update vector which is usually found by solving this equation using LU Decomposition. E is the error vector containing the output errors for each input vector used on training the network. The ϕ tell us by how much we should change our network weights. The J^tJ matrix can also be known as the approximated Hessian. The λ damping factor is adjusted at each iteration according to the difference in error.

4.1.4.2. SURF Based Neural Network Implementation. SURF [62] is a scale and rotation invariant feature detector approach. It is basically derived from SIFT [65] but outperforms it in terms of speed, robustness and distinctiveness. SURF has several parameters that may affect its output:

- Upright: This parameter determines whether to run Upright SURF (U-SURF) or not. U-SURF better fits the horizontal camera cases. It runs invariant of image rotation and therefore consumes less CPU time.
- Octaves: The scale space is divided into a number of octaves. The filter size is affected by the octave levels.
- Intervals: This is the sampling interval. Together with the number of octaves, it determines the number of filters to be applied. (Number of filters = octaves $\times$ intervals)
- Threshold: A threshold value used to control the accuracy of the results. Increasing the threshold value will decrease the number of detected interest points.

The SURF process finds a set of interest points, which have the following properties.

- (x, y): The center of the SURF interest point.
- Orientation: This is the orientation of the detected feature in radians.
- Scale: Scale takes values from 1 up to the number of octaves.
- Laplacian: A value of 1 indicates bright blobs on dark backgrounds, and -1 indicates just the reverse case.

In the early implementations of the project, the neural network was fed by the properties of a constant number of SURF interest points. However, this schema suffers from the ordering of

the inputs of the network. Therefore a new schema, which uses the number of SURF points found in predefined parts of the detected image as the input of the network, was developed. [66].

4.1.4.3. SVM Implementation. SVM [67] models were originally defined for the classification of linearly separable classes of objects. For any particular set of two-class objects, an SVM finds the unique hyperplane having the maximum margin. The solution to a classification problem is represented by the support vectors that determine the maximum margin hyperplane. SVM can also be used to separate classes that cannot be separated with a linear classifier. In such cases, the input values are mapped to a high-dimensional feature space in which the two classes can be separated with a linear classifier. Since the feature space is high dimensional, the nonlinear mapping is computed with special nonlinear functions called kernel. Kernels have the advantage of operating in the input space. The solution of the classification problem is a weighted sum of kernel functions evaluated at the support vectors.

In our implementation, we used the same input values, which are generated according to the process in Figure 4.9 to train the SVM with radial basis kernel function (RBF). The RBF kernel function is given in Equation 4.10

$$k(u,v) \quad = \quad e^{-\gamma \times |u-v|^2} \tag{4.10}$$

where k is the kernel function which gives the distance between the support vector u and the test point v. The coefficient γ is found by trying sample values at different exponential scales.

4.2. RF Module

The RF module provides information gathered from the RFID tags which are placed special points on the roads [68]. An RFID system consists of an antenna and a transceiver which can read the radio frequency and transfer the information to a processing device (reader) and a transponder, or RF tag, which contains the RF circuitry and information to be transmitted. The antenna provides the means for the integrated circuit to transmit its information to the reader that converts the radio waves reflected back from the RFID tag into digital information

that can then be passed on to computers that can analyze the data. The most common types of RFID modules and specific properties are given in Table 4.5 [7].

Table 4.5. Properties of most common RFID systems.

Band	**Low Frequency**	**High Frequency**	**Ultra-High Frequency**	**Microwave**
Typical RFID Frequencies	125-134 kHz	13.56 MHz	433, 865-956 MHz	2.45 MHz
Read Range	<0.5m	<1.5m	<5m	<10m
Data Transfer Rate	1 kbit/s	25 kbit/s	30-100 kbits/s	<100 kbit/s
Sample Application	Car Immobilizer	Contact-less travel cards	Logistics	Electronic toll collection

4.2.1. Challenges of RFID Technology

Although the RFID technology is a commonly used technology especially when fast identification is necessary, it has several challenges for the proposed application [7].

- Power consumption: The most important constraint of the RFID systems is the required power. If the tags are passive, then only the RF reader requires the power. However, for active tags, both the reader and the tags should be connected to a power source. Since the power requirement of the tags are relatively less, they can be equipped with small size batteries. On the other hand, the reader usually requires more power because it provides the current for the antenna.
- Reading range: After the power is supplied to the system, the reading distance of the reader system is calculated according to the structure and positioning of the antenna. Since the proposed application requires omnidirectional reading within two or three meters, the system may require additional amplifiers and power source. In addition, the passive tag solutions are not applicable to the proposed system because of their very low reading distances.
- Antenna: The antennas in the current active tag applications are usually designed for static implementations. Therefore the size and the shape of these antennas may result in more complicated designs for the RFID system on the vehicles.
- Metal interference: Although there are RFID systems which are not affected from rain or other weather conditions, most of these systems are sensitive to metal interference. The reason of this interference is the reflection and cancellation of the reader's electro-magnetic field by metallic objects other than the transponder within the tag.

- Passive tag limitations: The passive tags are cheap, easy to place and replace, and require no power source. They are the best candidates for identifying the traffic signs and other important objects. However, the required reading distance by the proposed solution is not available with the current readers without additional equipments. In addition, the limited power source on the vehicle also reduces the performance of the reader system.

4.2.2. Tag ID Assignment

The implementation of the RFID technology for the proposed application will contain a reader which is capable of reading tags up to two meters. If the RF reader can be placed somewhere near the front bumper, this distance is sufficient for reading the necessary markings. If we consider the directional restrictions, we place an RFID tag very close to the related regulation sign. In order to detect the violation of the restriction, another RFID tag is also placed a few meters ahead the crossroad in the restricted direction. According to this setup, the system can easily conclude that the driver violates the traffic rule if the RF reader on the car detects the first RFID tag and then the second tag in this specific order.

A slightly different scenario can be applied for the traffic lamps. However this time the first RFID tag should be an active one which is activated only when the red light is on. The violation of traffic rule can be detected in the same way as it is done in the previous scenario. The easiness of producing similar scenarios for different traffic rules makes the RFID technology very promising for the ADES project.

However, to accomplish these scenarios a predefined tag identification number assignment strategy should be designed. In Table 4.6 the tag ID assignment strategy for ADES project is given. In this strategy the RF tags are identified by their global positions, referring traffic rule ids. Finally a redundancy check field is also added for robustness. Since the total number of byte required for indicating a specific rule is eight, several rules can be included within a single tag. This strategy reduces the required number of total tags and the cost of the system.

Table 4.6. Tag ID assignment strategy.

Tag Part	Length (Bytes)	Explanation	Example
UniqueID	4	Latitude and Longitude	41.013144-29.081244
RuleID	1	Predefined Rule ID	13 (no trun left)
Pre/Post/Single	1	Type of Tag	Pre - no turn left
Matching Tag ID	1	Matching Tag for pre/post tags	UniqueID of pre tag for a post tag
CRC	1	Redundancy Check	01010101
	Total: 8 bytes		

4.3. Usage of GPS/GIS and CAN Bus Integration

The ADES system includes a simple Geographic Information System (GIS) system design and uses this to retrieve the existing traffic rules at the global position of the vehicle [69]. The proposed GIS database holds the rules as an XML file. A simple example of this XML file is given in Figure 4.10.

```
<?xml version="1.0" encoding="utf-8" ?>
<GIS>
  <NodeProp lat="0,0.04,N" lon="0,0.00,W" rule="1"/>
  <NodeProp lat="0,0.12,N" lon="0,0.00,W" rule="13"
            type="pre" matching="0,0.12,N:0,0.00,W"/>
  <NodeProp lat="0,0.13,N" lon="0,0.02,W" rule="13"
            type="post" matching="0,0.12,N:0,0.00,W"/>
</GIS>
```

Figure 4.10. Simple GIS XML for ADES.

The attribute properties are similar to the RF Tag ID assignment strategy given in Section 4.2.2. The lat attribute gives the latitude of the definition at a predefined resolution. Similarly the lon attribute presents the longitude. The $rule$ presents the existing rule id for the global position defined by the previous attribute. Since there may be only one definition for a specific rule in the defined position, the combination of the lat, lon, and $rule$ attributes can be considered as the unique ID of the GIS data. The resolution of the system depends on the precision of the lat and lon attributes. In the given example, each rule is effective in nearly 25 square meters in the simulated environment. The $type$ attribute provides additional information about the rule. Finally the $matching$ attribute provides corresponding rule for

pre/post rule pairs.

One other tool which can be used to obtain valuable information from the vehicle is the Control Area Network (CAN) bus integration. The CAN is a serial communications protocol developed by Robert Bosch GmbH. It supports distributed real-time control with a high level of security [70]. There are many ADAS applications which uses CAN bus integration for obtaining the sensor readings from the vehicle [71]. Although advance statistical analyses can be performed on the information obtained from the CAN bus, it also provides simple information like the speed of the vehicle and sensor readings from throttle, break, and steering control units [72, 73].

In the ADES simulation environment the speed of the vehicle is provided to the system via the vehicle itself. This scenario can be realized in real life examples by using the CAN bus integration.

4.4. Inference Engine

Expert system is a computer program that uses knowledge and inference functions to solve problems which requires human expertise for their solution [74]. The knowledge base and inference rules are either hand coded or learned by the system by observing another expert, possibly a human expert. The most important difference of an expert system from a conventional computer application is its inference engine. As an example, a dictionary may be a conventional computer program where a successful translator can be addressed as an expert system.

The knowledge base consist of the information encoded in a specific way and the basic functions to deal with this data. The inference engine is the main part of the expert system which uses these functions on available information to come up with a solution. However, an expert system has to explain its solution in a way that an human observer can understand. Moreover, an expert system may also utilize a learning subsystem to update its information and inference capabilities.

The expert system models can be categorized into two categories which are probabilistic

and rule based models [75]. The major properties of these models and their comparison are shown in Table 4.7 and Table 4.8.

Table 4.7. Properties of expert system models.

	Probabilistic Model	**Rule Based Model**
Knowledge Base	Probabilistic Structure, Facts	Rules, Facts
Inference Engine	Conditional probability evaluation (Bayes theorem)	Backward chaining, Forward chaining
Explanation Subsystem	Based on conditional probabilities	Based on triggered rules
Learning Subsystem	Change in probabilistic structure and probabilities	Adding, removing rules

Table 4.8. Comparison of expert systems models.

	Probabilistic Model	**Rule Based Model**
Advantages	Easy learning, Easy probability propagation	Easy explanation, Easy modification
Shortcomings	High number of parameters	Performance issues, Certainty implementation

There are advantages and disadvantages of both models. The probabilistic model provides a proper handling of uncertainty with easy learning capabilities. Statistical information can be used directly to form a probabilistic model. However, high number of parameters even in small models makes the model more complicated to understand and manage. On the other hand, the rule based systems are easier to maintain. However, this advantage usually causes performance issues when extensive rule chaining is required for complex problems.

4.4.1. Expert System Interface

In order to integrate various expert system implementations into the ADES project, we defined an interface as shown in Fig. 4.11.

```
public interface ExpertSystems
        {
            string init(params object[] esParams);
            string assertFact(params object[] esParams);
            string retractFact(params object[] esParams);
            string[] query(params object[] esParams);
            void setThreshold(double threshold);
            double getThreshold();
        }
```

Figure 4.11. ADES Interface for Expert Systems.

The $init$ function initializes the expert system with various parameters like network definition files for BN based expert systems or the built in predicates of the Prolog based expert system. The $assertFact$ function provides the high level information to the expert system. This information is formed by processing the data acquired from different sensors. The $retractFact$ function is used when a previously introduced information is invalidated. However, the expert systems are free to retract their facts according to their internal operations. The $query$ function is the main procedure triggered right after a fact is asserted or retracted. This function is expected to return the kind of violation and its probability if there is a traffic violation. Finally, the $setThreshold$ and $getThreshold$ functions are used for arranging the threshold level of the expert system for deciding a violation. In the current state of the projects there are two implementations of this interface which are the Prolog based and Belief Network's based expert systems.

4.4.2. Implementation with Prolog

Prolog is a programming language for symbolic computation [76]. It is suitable for solving problems which contains objects and relations between these objects. The relations with variables are called the *rules*. Rules have condition and conclusion parts. On the other hand, the *facts* are means of stating that relations exist between objects. Therefore, the relations of the facts only contains atoms [76, 77]. The power of Prolog comes from three major features of the language which are rule-based programming, built-in pattern matching and backtracking execution. Pattern matching and backtracking features provide automatic control of the flow in the program which make it possible to realize many types of expert systems [74].

Writing a Prolog program is not similar to writing a program in a conventional programming language. The Prolog programmer should consider the formal relationships exist in the problem domain. In other words, Prolog approach is more like describing the facts and rules about the problem domain instead of prescribing a procedural algorithm [78].

4.4.2.1. Probabilistic Prolog Enhancement. In real life problems, the relations between the objects are not always very clear. Therefore, an expert system should also consider the uncertainties associated with the supplied data. There are different ways to provide certainty factors or probabilities to the facts in Prolog. In the simplest way the facts can be coupled with their probabilities as shown in Figure 4.12.

Rules and Facts

```
weather(fine,P) :- sky(sunny,PS),temp(high,PT),P is PS*PT.
sky(sunny,0.8).
temp(high,0.9).
```

Query

```
?- weather(X,P).
X = fine
P = 0.72 (0,016 sec)
```

Figure 4.12. Probabilistic Prolog example.

In this example, the probability of a fine weather condition is calculated according to the probabilities of the sky and temperature facts with given probability values. This method works well for small examples, however, unnecessary probability calculation overhead in the rules and the additional parameters in the fact definitions can increase the complexity in more complicated applications.

The second approach use operators to give probabilities to the facts. Unlike the previous examples, the rules do not have additional parameters for probability calculations. We can modify the previous example as shown in Figure 4.13.

In this project this method is preferred and the probability operator '::' is implemented within the Prolog engine. However, there are other implementations where the probability processing extensions can be consulted as libraries without changing the engine itself [79].

Rules and Facts

```
weather(fine):-sky_sunny,temp_high.
"0.8::sky_sunny".
"0.9::temp_high".
```

Query

```
?- weather(X).
Prob for sky_sunny is 0,8
Prob for temp_high is 0,9
Prob for this answer:0,72
X = fine (0,000 sec)
```

Figure 4.13. Facts with operators.

Another common example for visualizing the operation of the probabilistic Prolog is the path finding application. In this example a small Prolog application is presented to find a path on a graph. The probability of the achieved path is calculated from the probability values assigned to the edges of a graph. If the directed graph in Figure 4.14 is considered, there are three acyclic path from node one to node five.

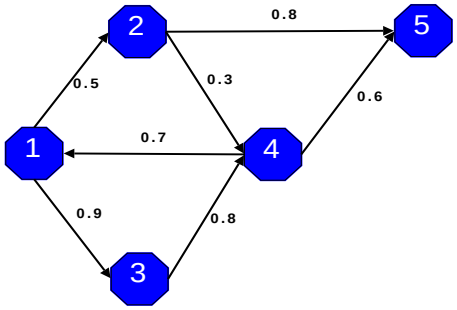


Figure 4.14. Directed graph with probability values.

The representation of the graph by facts, the Prolog program which returns the paths between two nodes, and the result of a sample query are given in Figure 4.15.

This program finds a path without traversing a node twice and returns the list of nodes on the path. The algorithm keeps a list of visited nodes which prevents infinite loops on cyclic links. As mentioned before, the rule definitions do not deal with the probability issues, which means they do not have any object, or parameter related with probability calculations. The probability values of the edges are given within the fact definitions.

Rules and Facts

```
path(X,Y,[X|NL]) :- pathX(X,Y,L), reverse(L,[],NL), notvisited(X,NL).

pathX(X,Y,L) :- pathX(X,Y,_,L).
pathX(X,X,A,A).
pathX(X,Y,A,R) :- X\==Y, edge(X,Z), notvisited(Z,A), pathX(Z,Y,[Z|A],R).

notvisited(X,[Y|Z]):- X \= Y, notvisited(X,Z).
notvisited(_,[]).

reverse([X|Y],Z,W) :- reverse(Y,[X|Z],W).
reverse([],X,X).

"0.9::edge(1,3)".
"0.5::edge(1,2)".
"0.3::edge(2,4)".
"0.8::edge(2,5)".
"0.8::edge(3,4)".
"0.7::edge(4,1)".
"0.6::edge(4,5)".
```

Query

```
?- path(1,5,L).
Prob for this answer:0,432
L = [1,3,4,5] (0,000 sec) more? (y/n);
Prob for this answer:0,09
L = [1,2,4,5] (0,000 sec) more? (y/n);
Prob for this answer:0,4
L = [1,2,5] (0,000 sec) more? (y/n);
no
```

Figure 4.15. Prolog program for graph example.

4.4.2.2. Time-Decaying Facts. Although the probabilistic extension to the Prolog engine provides mechanisms for dealing with uncertainty, the required expert system should possess additional properties. One of these improvements is the time-decaying property of the probabilistic facts. This enhancement is required when the asserted fact has some time limitations. For example, if the sensors indicate that there is a directional restriction on the way, then we may want to assign an expiration time like 30 seconds for that fact.

The decay function can be any monotonically decreasing one which starts from one and eventually decreases to zero. In the proposed system sigmoid functions are used as the decay function. The characteristic behavior of the sigmoid function, given in Equation 4.11, is illustrated in Figure 4.16 where α is 0.5 and β is 30. According to this function the probability

of the given fact will halve within 30 seconds, and the fact will retracted from the database before 45th second. It is clear that the decaying characteristics can be easily controlled by modifying the α and β parameters.

$$f(x) = 1 - \frac{1}{1 + e^{\alpha \times (\beta - x)}} \tag{4.11}$$

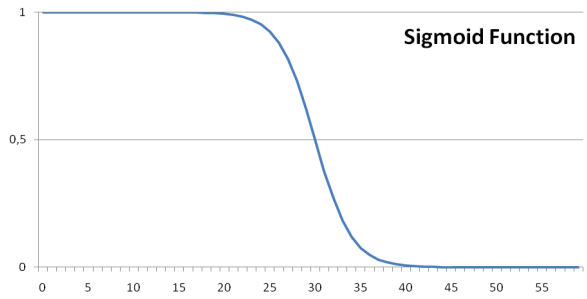


Figure 4.16. Sigmoid function used for time-decaying facts.

The time-decaying property of the facts are given in their definitions. In the previous graph example, assume that the probability of the edge between second and fourth nodes is decreasing over time. According to our sigmoid function, the probability of this edge becomes 0.15 at 30th second after the fact is asserted. And finally the fact will no longer be available after 45 seconds as shown in Figure 4.17.

The T literal between two probability operators '::' indicates that the probability of the fact will decay according to the built-in decay function, which is the sigmoid function in the proposed implementation. An integer value can also be placed instead of the T value to enlarge or shorten the duration of the value as shown in the Figure 4.18.

4.4.3. Implementation with Belief Networks

A belief network (BN) is an acyclic graph used for modeling the uncertainty in a domain [80]. The nodes of the graph are random values and the arcs are conditional dependencies between these variables. BN's are reasoning tools that depend on the laws of probability. Although it is possible to design multiply-connected (or cyclic) BN's, probabilistic inference

Rules and Facts

```
%only replace the fact about edge(2,4)
%remaining facts and rules are the same...
"0.3::T::edge(2,4)".
```

Query

```
%initial response to the query
?- path(2,4,L).
Prob for this answer:0,299999908226781
L = [2,4] (0,000 sec) more? (y/n);
no

%waiting for 30 seconds
?- path(2,4,L).
Prob for this answer:0,155176213884928
L = [2,4] (0,000 sec) more? (y/n)
no

%after 45 seconds
?- path(2,4,L).
no
```

Figure 4.17. Time decaying facts.

Rules and Facts

```
%replace T value with -20 to speed up the decay process
"0.3::-20::edge(2,4)".
```

Query

```
%initial response to the query
?- path(2,4,L).
Prob for this answer:0,299999908226781
L = [2,4] (0,000 sec) more? (y/n);
no

%waiting for 10 seconds
?- path(2,4,L).
Prob for this answer:0,155176213884928
L = [2,4] (0,000 sec) more? (y/n)
no

%after 25 seconds
?- path(2,4,L).
no
```

Figure 4.18. Graph example with time decaying facts.

using this type of networks is NP-hard [9].

The BN based expert system developed for the ADES project uses discrete random variables represented by a singly-connected graph. The value of a discrete random variable should be a member of the discrete values from a finite set of states. In addition, the sum of the probabilities of those states for each variable should be equal to one.

4.4.3.1. Microsoft Bayesian Network Editor. The Microsoft Bayesian Network Editor (MSBNx) is a Microsoft Windows library for creating and evaluating Bayesian Networks [81]. It can be used as a standalone modeling and inference application, or the run-time components which provide Bayesian reasoning services can be integrated to custom applications. The graphical interface for developing BN graphs can be used in both cases. The resulting BN files can be consulted by the custom application during run-time. The main purpose of MSBNx environment is diagnostic and troubleshooting inference using complicated evidences.

In the ADES project, MSBNx is used for implementing the BN expert system. The networks are created by using the MSBNx editor. These networks are loaded to the application when the application initializes.

When BN is selected as the active expert system, in case of an arrival of information, the probabilities of the states in the initial nodes are arranged according to the certainty of the information. After this arrangement, all of the loaded networks are queried for a possible violation. The networks respond to this inquiry by returning their derived beliefs for the *YES* state of the violation variable. The belief of this state highly depends on the subjective probabilities entered by experts.

4.4.4. Regulation Implementations

In the ADES Project, the goal is to detect the traffic violations according to the sensor values. The proposed inference engine includes rules for detecting the violations given in Table 4.9 to accomplish this mission. The assertion and retraction of each fact triggered from the sensors are given in Table 4.10. It is the responsibility of the expert system to keep the facts up to date according to these rules. The speed of the vehicle is not included in the table because it is retracted and asserted again by the system before each validation query.

Table 4.9. Targeted traffic violations.

No Turn Left		Sign Detection,
No Turn Right		RFID Detection,
No U Turn		GPS/GIS Information,
No Entrance		Vehicle Speed,
Speed Limit		Steering Angle,
No Overtaking		Lane Detection,
Traffic Lights		Time Elapsed

Table 4.10. Assertion and retraction of the facts.

Regulation	**Assertion of Facts**	**Retraction of Facts**
Directional	Traffic sign detection	After a period of time.
	GPS node rule from GIS	Exiting the node.
	RFID tag detection (Pre/Post)	After a period of time,
		Detecting another RFID tag for the same rule.
Speed	Traffic Sign Detection	Restriction ends sign detection,
		Another speed limit sign.
	RFID tag detection	After detecting another RFID tag for the same rule.
	GPS node rule from GIS	Exiting the node.
Traffic Lights	GPS node rule from GIS	Exiting the node.
	RFID tag detection (Pre/Post)	Detecting another RFID tag for the same rule.
Illegal Overtake	Traffic sign detection	Restriction ends sign detection.
	GPS node rule from GIS	Exiting the node.
	Solid Lane Departure	Immediately after query.

In the following sections the explanations of these rules are given. In the expert system design, it is assumed that the inference engine acquires data from three sensor systems which are the vision system, the RFID system and the GPS system. Additional systems can be introduced to the application as long as they can provide facts to the inference engine with their certainty factors.

4.4.4.1. Violation of Directional Regulations. The directional regulations considered in our application are 'no turn left', 'no turn right', 'no U turn', and 'no entrance' conditions. For detecting such violations, inputs from the vision, the RFID and the GPS systems are utilized in the proposed rule. A sample scenario is illustrated in Figure 4.19.

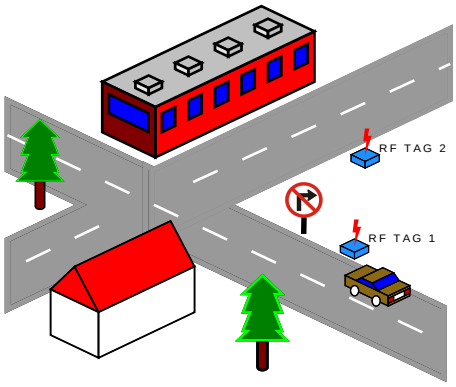


Figure 4.19. Sample scenario for detecting directional regulation violations.

In the prolog implementation as shown in Figure 4.20, the rule definition for violation detection and examples for possible facts asserted by the sensor systems are given below.

Please note that the probabilities are decreasing over time after the facts are asserted. Therefore, the probability of the violation may slightly differ from the multiplication of the probabilities of the facts.

In the BN design, the network and the human expert provided probability values are given in Figure 4.21. The probability values may be assigned considering the possible false recognition of the signs. For example, the 'no U turn' sign can be detected as 'no left turn' sign by the vision module. If the human expert considers these possible conflicts while making the probability assessments, then the system has more chance of recovering its error by using the information provided by the other sensors.

Rules

```
%rule declaration
violation(V,A,P) :- sign_detected(V),
                    rfid_detected(pre,V),
                    rfid_detected(post,V),
                    gps_node_property(post,V),
                    gps_node_property(pre,V),
                    A is "N/A",
                    prob(P),
                    P>0.8.
```

Query

```
Exec: assert("0.9999::T::sign_detected(no_turn_left)").
Exec: assert("1::T::rfid_detected(pre,no_turn_left)").
Exec: assert("0.9110::T::gps_node_property(pre,no_turn_left)").
Exec: assert("0.8962::T::gps_node_property(post,no_turn_left)").
Exec: assert("1::T::rfid_detected(post,no_turn_left)").

Exec: violation(V,A,P).
Prob for this answer:0.816324396689487

P = 0.816324396689487
V = no_turn_left
A = "N/A"
```

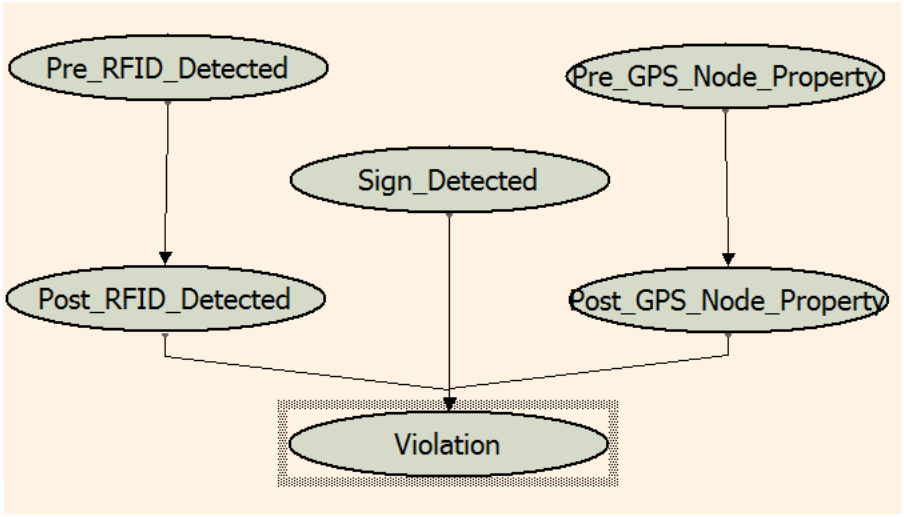
Figure 4.20. Prolog program for directional regulation violations.

4.4.4.2. Violation of Speed Limitation. Speed limitations are usually indicated by red circular signs across Europe. These circular signs contain a number which is the maximum allowed speed limit for the motorway it is assigned for. GPS input can be used as well as visual and RF inputs for detecting the speed limit violations [82]. In addition, the proposed method also uses information gathered from the vehicle itself. The speed of the vehicle is assumed to be supplied by the vehicle.

The prolog rules for speed limit violation detection are given in Figure 4.23.

Similarly, the BN implementation benefits from the vision, RF and GPS inputs. The probability assessments are again given by the human expert. The network diagram and assessments are shown in Figure 4.24.

a)



b)

Assessment (Model: NoTurn, Node: Violation)

Parent Node(s)			Violation		
Sign_Detected	Post_RFID_Detected	Post_GPS_Node_Property	Yes	No	bar charts
No_Turn_Left	Post_No_Turn_Left	Post_No_Turn_Left	0,95	0,05	
		Post_No_Turn_Right	0,1	0,9	
		Post_No_U_Turn	0,1	0,9	
		None	0,1	0,9	
	Post_No_Turn_Right	Post_No_Turn_Left	0,1	0,9	
		Post_No_Turn_Right	0,1	0,9	
		Post_No_U_Turn	0,1	0,9	
		None	0,1	0,9	
	Post_No_U_Turn	Post_No_Turn_Left	0,1	0,9	
		Post_No_Turn_Right	0,1	0,9	
		Post_No_U_Turn	0,9	0,1	
		None	0,1	0,9	
	None	Post_No_Turn_Left	0,1	0,9	
		Post_No_Turn_Right	0,1	0,9	
		Post_No_U_Turn	0,1	0,9	
		None	0,1	0,9	
No_Turn_Right	Post_No_Turn_Left	Post_No_Turn_Left	0,1	0,9	
		Post_No_Turn_Right	0,1	0,9	
		Post_No_U_Turn	0,1	0,9	
		None	0,1	0,9	
	Post_No_Turn_Right	Post_No_Turn_Left	0,1	0,9	
		Post_No_Turn_Right	0,95	0,05	
		Post_No_U_Turn	0,1	0,9	
		None	0,1	0,9	

Figure 4.21. Directional regulation's a) belief network and b) property assessments.

4.4.4.3. Illegal Overtaking. Illegal overtaking occurs during the course of passing a slower car under unsafe conditions. The overtake forbidden sign is given in Figure 4.25

The prolog rules for illegal overtaking detection are given in Figure 4.26.

The BN implementation of illegal overtaking is similar to speed limitation. However the probability assessment is relatively easy since the nodes has less attributes. The network and the probability assessment are given in Figure 4.27.

Figure 4.22. Speed limit aware GPS application.

Rules

```
%rule declaration
violation(V,A,P) :- gps_node_property(V,L),
                    velocity_exceeds(S),
                    sign_detected(V,L),
                    S>=L,
                    atom_chars(S,L1), append(L1,[95],L2),
                    atom_chars(L,L3), append(L2,L3,L4), atom_chars(A,L4),
                    prob(P),
                    P>0.9.
```

Query

```
Exec: assert("1.0000::T::sign_detected(speed_limit,30)").
Exec: assert("0.9759::T::gps_node_property(speed_limit,30)").
Exec: assert("1::T::velocity_exceeds(50)").

Exec: violation(V,A,P).
Prob for this answer:0.975898794128071

P = 0.975898794128071
V = speed_limit
A = '50_30'
```

Figure 4.23. Prolog program for speed limit violations.

4.4.4.4. Violation of Red Traffic Lights. The assertion of the red light rule is similar to directional regulations as shown in Figure 4.28. However, red light rules are temporary. Which means that, the rule is strongly related with time. Since it is usually hard to recognize the traffic signs with the vision system, an RFID system with active tags can be used for this purpose. However, the RF system should contain active RF tags since the tags should be responding for a specific period of time. The system certainly can benefit from a visual feedback for the traffic lights. The traffic light detection can be used for enforcing the rules on crossroads like blocking the road in heavy traffic conditions.

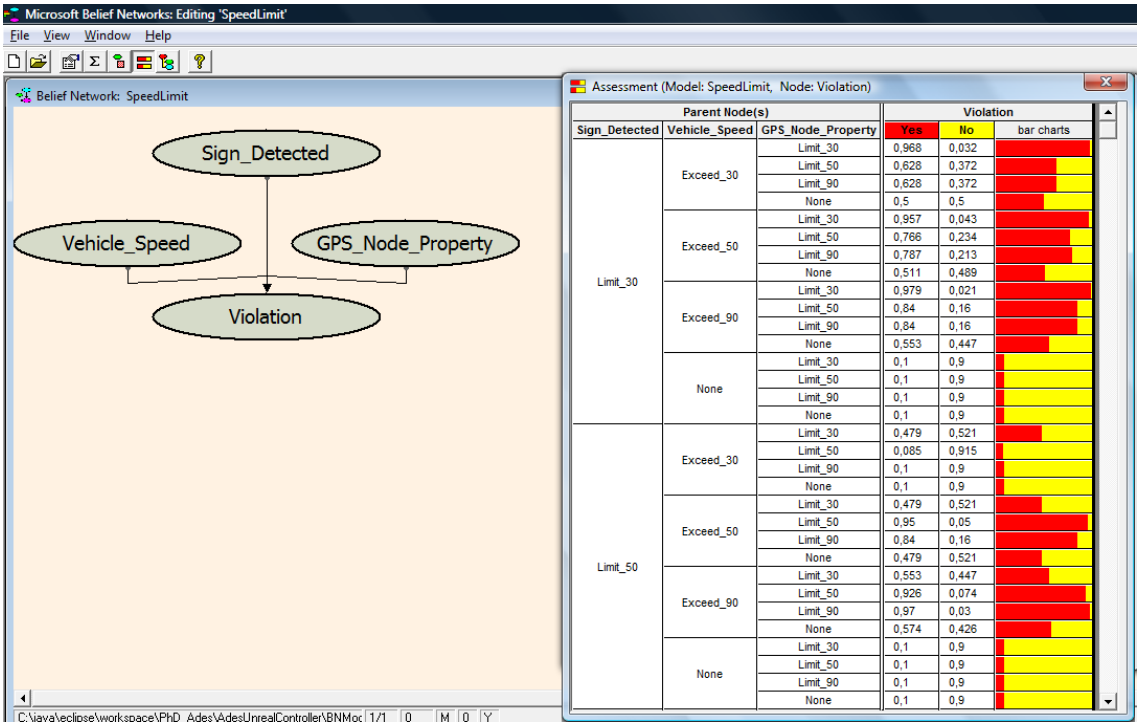


Figure 4.24. The belief network and it's property assessments for speed limitations.

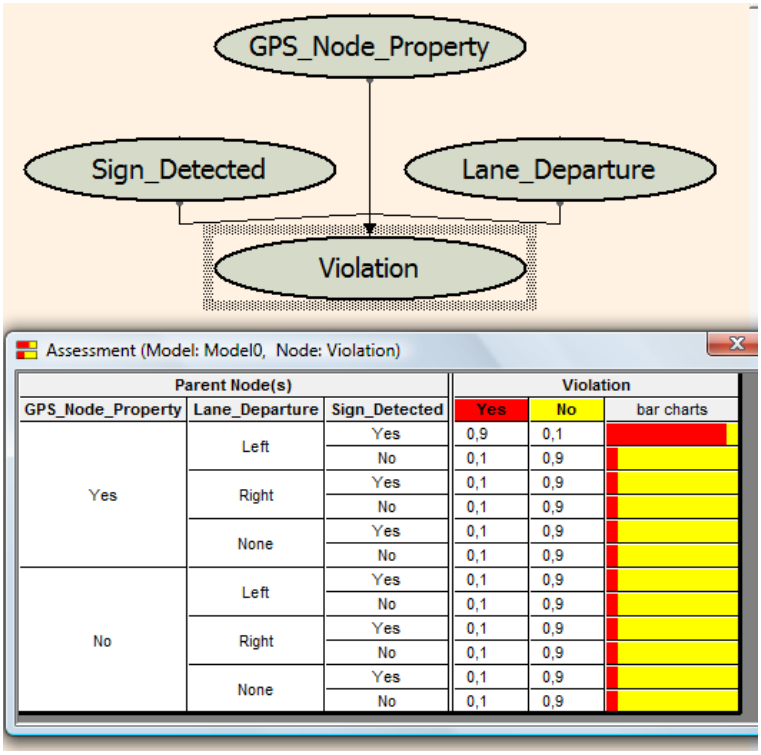
Figure 4.25. Overtake forbidden sign.

Rules

```
%rule declaration
violation(V,A,P) :- sign_detected(no_overtake),
                    rfid_detected(no_overtake),
                    gps_node_property(no_overtake),
                    lane_departure(left),
                    A is "N/A",
                    V is "no_overtake",
                    prob(P).
```

Figure 4.26. Prolog program for illegal overtaking.

The prolog rules for red ligth violation detection are given in Figure 4.29.



Parent Node(s)			Violation		
GPS_Node_Property	Lane_Departure	Sign_Detected	Yes	No	bar charts
Yes	Left	Yes	0,9	0,1	
		No	0,1	0,9	
	Right	Yes	0,1	0,9	
		No	0,1	0,9	
	None	Yes	0,1	0,9	
		No	0,1	0,9	
No	Left	Yes	0,1	0,9	
		No	0,1	0,9	
	Right	Yes	0,1	0,9	
		No	0,1	0,9	
	None	Yes	0,1	0,9	
		No	0,1	0,9	

Figure 4.27. The belief network and it's property assessments for illegal overtaking.

The BN implementation of traffic light regulations is similar to directional regulations. The vehicle should sense both pre and post RF tag to violate the traffic rule. The network and the probability assessment are given in Figure 4.30.

4.5. Modeling Driver Aggressiveness

There are various social studies about the causes of driver aggressiveness [83] or results of this aggression [84]. According to the Florida Department of Highway Safety and Motor Vehicles' Aggressive Driver Study [85], there is no well defined definition of the term *aggressive driving*. A driver who performs actions such as weaving in and out of traffic lanes to get ahead, or tailgating can be defined as an aggressive driver. Exceeding speed limits, changing lanes frequently, and cutting other drivers way ahead are common behaviors of aggressive drivers. The aggressive drivers exhibit more hard deceleration, acceleration, and swerve maneuvers during baseline driving than did the safe drivers, which may increase drivers' relative crash risk above that of normal driving [86].

In the ADES project, driver aggressiveness is measured using the number of traffic

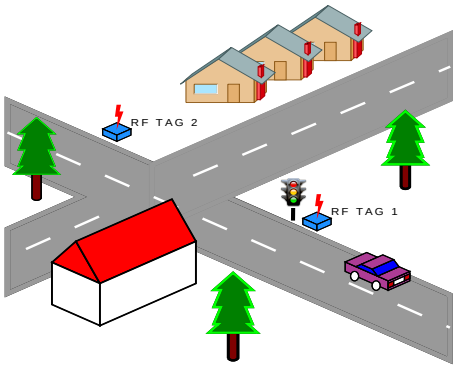


Figure 4.28. Sample scenario for red light violations.

Rules

```
%rule declaration
violation(V,A,P) :- rfid_detected(pre,light_red),
                    rfid_detected(post,light_red),
                    gps_node_property(traffic_lights),
                    A is "N/A",
                    V is "red_light",
                    prob(P).
```

Figure 4.29. Prolog program for red light violation.

violations performed by the driver. Each violation increases the coefficient c by one. If the driver does not violate any rules for a specific period of time, this coefficient is decreased by one. This coefficient is used as the multiplicand of the threshold value used against the violation probabilities of the expert systems as shown in Equation 4.12.

$$Violation \quad = \quad \begin{cases} Yes \rightarrow p > t \times (1 - \alpha \times c/c_{max}), 0 < \alpha < 1 \\ No \rightarrow o/w \end{cases} \tag{4.12}$$

The c_{max} value is the maximum possible aggressiveness coefficient for any driver. p is the output of the expert system and t is the predefined threshold value for accepting the probability of the expert system output as violation. Finally, α is the strength of the effect of the driver aggression model on the threshold. Its value is set to 0.1 in the current implementation which means the maximum effect of the driver aggression model can be 0.1 on the threshold value.

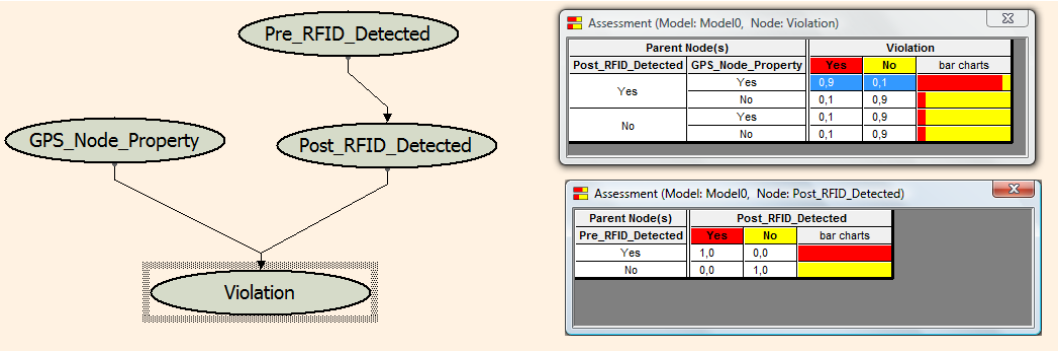


Figure 4.30. The belief network and it's property assessments for traffic light regulations.

5. APPLICATIONS

Two main applications have been developed for the ADES project. The first one, the ADES detector, is used for detecting road lanes and traffic signs. The second application is a simulation environment which is used as a test bed for both sensor modules and the inference engine.

5.1. The ADES Detector

The ADES detector application is designed for processing video feedbacks for detecting road lanes and recognizing traffic signs. This video feedback can be a real-time streaming video feed from a camera, a recorded video stream, a stream fed from network or any memory mapped image source. The main purpose of this tool is to test the performances of the vision processors. The results of different processors can be visualized in real-time as shown in Figure 5.1. The red lines on the figure are the detected road lanes where the rightmost images are the recognized circular traffic sign and the detected triangular sign. Another very important

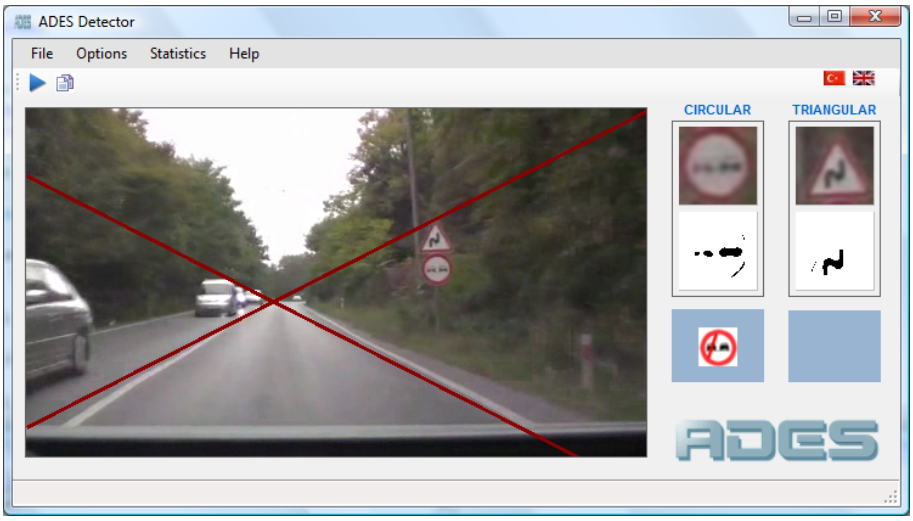


Figure 5.1. The ADES Detector.

service of the ADES detector application is the training of the neural networks and support vector machines used for sign identification. The user can initiate the training sequence after appropriately placing the training files to the corresponding file folders. The NN training phase creates a network for each sign. The SVM training phase creates a single model for all

signs.

The application also provides an option for using the Linear Discriminant Analysis (LDA) features instead of all input vector for traffic signs [87, 88]. The LDA process traverses all training files and figures out the most important parametric number of pixels on the traffic sign images. Then the training can be done by using these features, and similarly the test process can also use these features.

5.2. ADES Simulation Environment

Although the ADES project targets the real physical environment, the development process can be accelerated by the use of a realistic simulation environment. Our choice for the simulation engine is the *Unreal Engine* which provides the physics engine and the renderer. The success of this engine makes it a favorite tool for both robotic researchers and game developers.

The interaction between the Unreal Engine, USARSim and the Unreal ADES controller is shown in Figure 5.2. The system has three components, each of which can be run on separate computers. The first part is the Unreal Engine with the USARSim expansion. This is the core of the simulation system where rendering and physics calculations are performed. The second part is the extended version of the USARSim Image Server (UPIS). The role of this part is serving vision information to the custom applications rather than the Unreal Engine's default client application. The last part, which is the ADES Unreal Controller, is a client for the USARSim and the UPIS servers. The details of each module are given in the following sections.

5.2.1. Unreal Engine

The Unreal Engine is a commercial game engine developed by Epic Games. Its first version, UE1, was announced in 1998. In 2002, the next version, UE2, was released. The ADES simulation environment is based on UE2.5, which is an improved version of UE2 and released a few years later. However, the latest version of the engine is UE3 and equipped

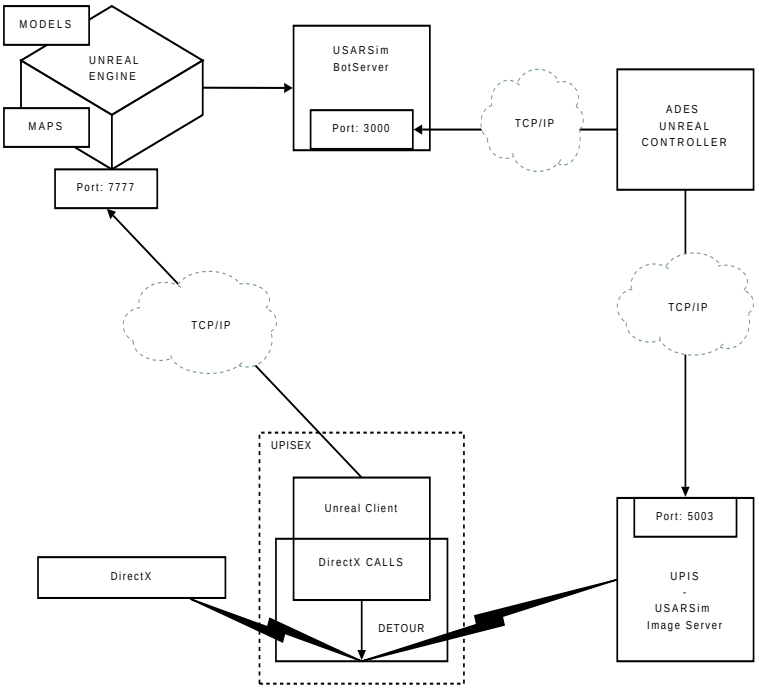


Figure 5.2. Interaction between Unreal Engine, USARSim and Unreal ADES controller.

with many advanced techniques. However the hardware requirements for this engine are not compatible with our project. Unfortunately, the UE2.5 engine is not freeware and can be obtained by purchasing the Unreal Tournament 2004 game. It also includes the physics engine, which is known as the Karma engine [89].

Unreal Engine also provides an object oriented programming called the UnrealScript which is a high level programming environment for game developers. The Java style coding with built in basic tasks like object's event handling, automatic garbage collection, and many base classes for different kinds of actors enables fast and neat game development.

5.2.2. USARSim

USARSim is an Unreal Engine expansion for urban search and rescue (USAR) robots and environments. It is an open source research tool for the study of various kinds of robotic applications. USARSim platform can be used in many domains like Defense Advanced Research Projects Agency (DARPA) Urban Challenge or National Institute of Standards (NIST) Reference Test Facility for Autonomous Mobile Robots for Urban Search and Rescue. However, RoboCup Rescue Virtual Robots Championship is maybe the most important domain for

us where the USARSim platform is used. The know-how gained from this project can also be transferred to the teams of our researchers in our laboratory who are planning to participate in this competition.

The USARSim environment provides numerous models for the use of researchers. We are using the Sedan class as our vehicle in the simulated environment. The object hierarchy and brief descriptions of these classes are given in Table 5.1.

Table 5.1. The class hierarchy of Sedan class.

	Unreal Classes (Commercial)	
	Class	**Description**
1	Actor	The base class of every object that have a position in the Unreal world.
2	Pawn	Parent class for all controlled entities.
3	Vehicle	Parent class for all vehicles controlled by the player.
4	KVehicle	Vehicles Karma physics.
	USARSim Classes (Open Source)	
	Class	**Description**
5	KRobot	USARSim's physical robot base
6	GroundVehicle	Vehicles with wheels and steering
7	AckermanSteeredRobot	Ackerman Steered Vehicles
8	Sedan	The Sedan class used by ADES Unreal Controller

In addition to various vehicle classes, USARSim also provides numerous sensors to equip these vehicles. In the current implementation of ADES simulation, the camera and the RF reader sensors are mounted on the Sedan as shown in Figure 5.3. In order to mount a sensor to the vehicle, it is sufficient to append the descriptive line to the definition of the vehicle in the Usarsim.ini file located at the System directory of the Unreal Engine. The current definition of the Sedan class is given in Figure 5.4.

In this definition, there are several parameters about the physical properties of the vehicle. In addition, the tires of the vehicle also represented since the Sedan class is extended from AckermanSteeredRobot class. MisPkgs are used for defining objects which can accept Mission Packages. For example, the pan and tilt of a camera can be controlled by using Mission Packages. In the ADES project, since all of the sensors on the vehicle are fixed, Mission Packages are not used. The sensors on the Sedan are RF reader, Camera and GroundTruth. The GroundTruth sensor provides information about the position and orientation of the vehicle.

Figure 5.3. The Sedan in the ADES Unreal world.

From the Usarsim.ini file in UT2004\System directory.

```
[USARBot.Sedan]
bDebug=False
Weight=5
Payload=2
ChassisMass=10
MaxTorque=200.0
MotorTorque=200.0
bMountByUU=False
JointParts=(PartName="RightFWheel",PartClass=class'USARModels.SedanTireRight' ...
JointParts=(PartName="LeftFWheel",PartClass=class'USARModels.SedanTireLeft',. ..
JointParts=(PartName="RightRWheel",PartClass=class'USARModels.SedanTireRight', ...
JointParts=(PartName="LeftRWheel",PartClass=class'USARModels.SedanTireLeft', ...
MisPkgs=(PkgName="CameraPanTilt",PkgClass=Class'USARMisPkg.CameraPanTilt')...
Cameras=(ItemClass=class'USARBot.RobotCamera',...
Sensors=(ItemClass=class'USARBot.GroundTruth', ...
Sensors=(ItemClass=class'USARBot.RFIDSensor', ...
Sensors=(ItemClass=class'USARBot.GPSSensor', ...
```

Figure 5.4. The definition of Sedan class.

Since this sensor is prone to cumulative errors, it is not used in the current implementation. However, the GPS sensor, which is also available in USARSim, will be used in near future.

USARSim also provides the USARDeathMatch game type which enables the BotServer. The BotServer opens a communication channel between the Unreal Engine and the outside

world. Custom objects derived from the *pawn* class can be initiated from remote controlling applications by connecting to this server. USARSim also provides its own communication protocol. The details of this protocol can be found on the USARSim manual [90].

The USARSim games require maps to run on. Therefore, we developed an initial version of the ADES Unreal world by using the Unreal level editor.

5.2.3. UnrealEd: Level Editor

The ADES simulated environment is developed using the UnrealEd, which is the level editor of Unreal Engine. UnrealEd user interface is based on quad viewports which are side, front, top views and an additional perspective view as shown in Figure 5.5.

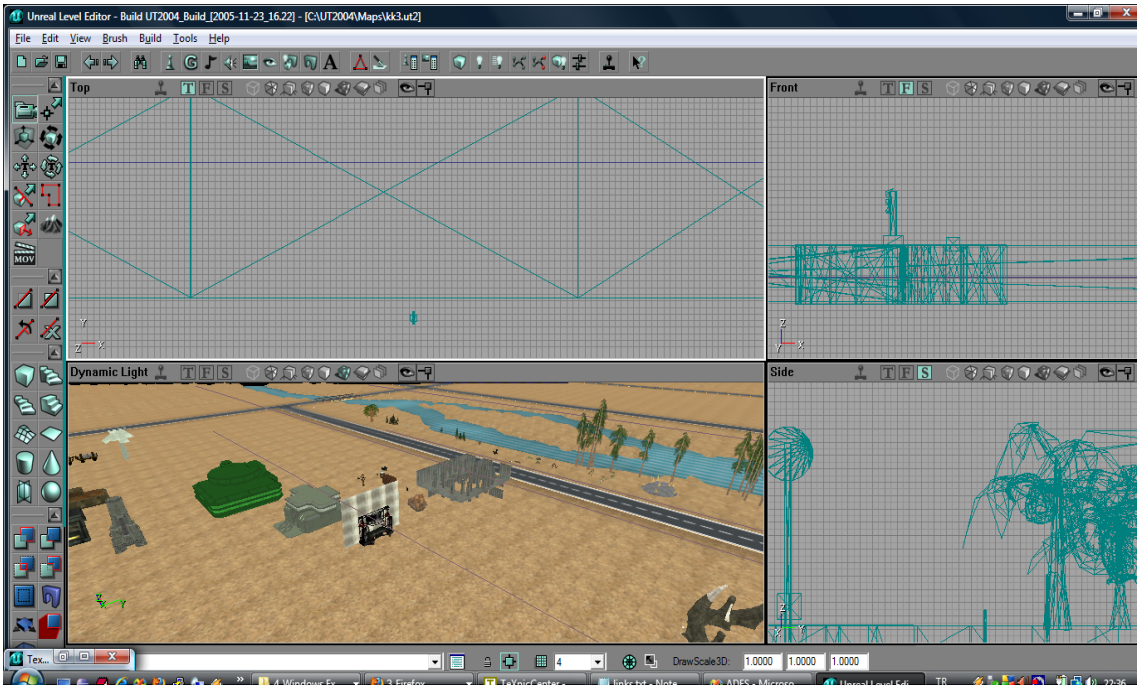


Figure 5.5. The ADES Unreal world in UnrealEd.

The UnrealEd has subtractive and additive brushes for adding or removing objects with various shapes. Although the latest version of the UnrealEd supports both empty and filled initial spaces, we are using the former version which only supports filled initial spaces. Therefore, the initial step for creating an environment is carving a quite big rectangular object from the space.

The unreal editor has the following set of artifacts;

- Actors: Any object that can be placed into the environment. These objects can be controlled pawns like Sedan, or, other items like pickups, decorations, zone information etc.
- Textures: Textures are images which can be applied to the new created shapes by using brushes.
- Static Meshes: The reusable objects which contain previously created shapes covered with textures.
- Meshes and Animations: Set of meshes which forms repeating animations for shapes.
- Other Artifacts: Sounds, Music, Prefabs (saved brushes).

In the ADES project different textures and static meshes are used. Some examples are given in Figure 5.6. Different actors are also placed for different purposes.

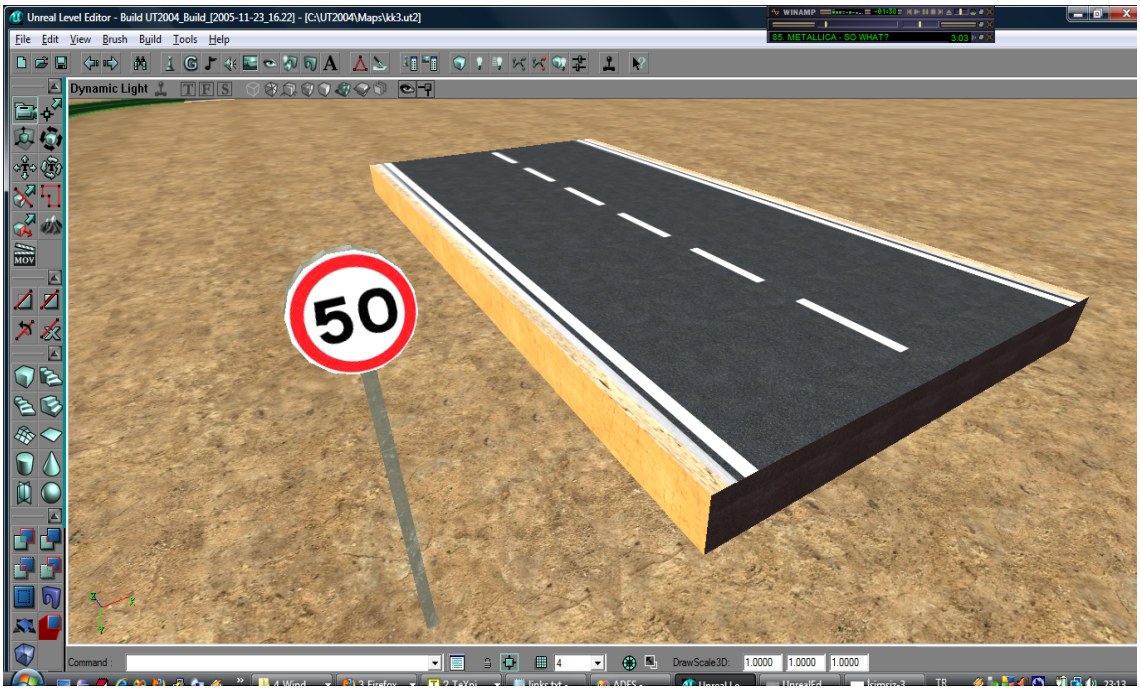


Figure 5.6. Road segment and speed limit sign static meshes.

5.2.3.1. Terrain Info. The TerrainInfo object places a ground structure on the Unreal level. The parameters for the distributions of the random distortions are parametric. In addition, there are various tools for increasing and decreasing the levels of the terrain at specific points. The wireframe representation of the terrain of the ADES Unreal world is given in Figure 5.7.

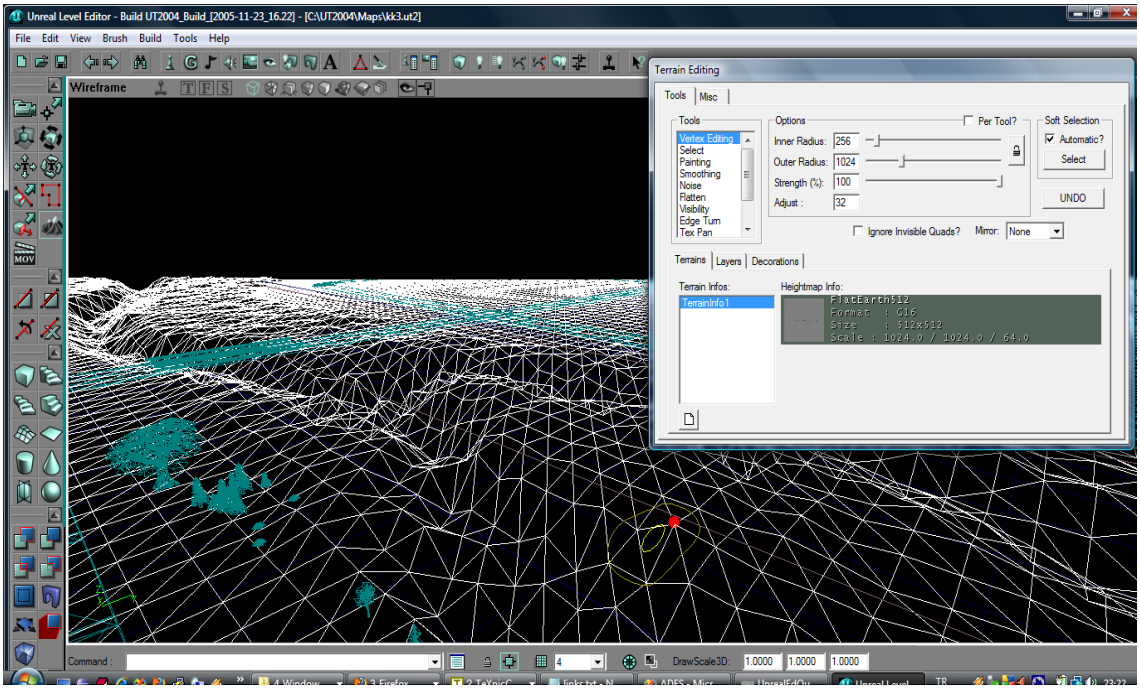


Figure 5.7. The wireframe of the terrain of the ADES Unreal world.

5.2.3.2. Zone Info. Miscellaneous properties of the level can be configured by using the ZoneInfo object. The zone lighting, ambient sound effects, fog effects are some of these properties. In addition to the ZoneInfo, different objects derived from Light class can be used for ambient lighting, like SunLight or SpotLight classes.

5.2.3.3. Sky Box. As we mentioned before, the first step of the level design is carving the world as a big rectangle . Initially, the walls of this rectangle which bound our world are empty. The SkyZoneInfo object provides artificial scenes for these walls. This object is placed within a relatively small closed empty space like a cube as shown in Figure 5.8. Then the walls of this smaller empty space are decorated with appropriate wallpapers. Finally, the walls of the bigger rectangle is set as *Fake Backdrop* which is a setting in *Surface Flags*.

5.2.4. UPISEX

UPISEX is a wrapper project for the USARSim's image server UPIS for making it visible for .NET projects. The main task of this project is creating the Unreal Client with Detours hook. Detours [91] is a library for intercepting Win32 functions by re-writing the in-memory code for new target functions. The detoured function call is returned to the intended

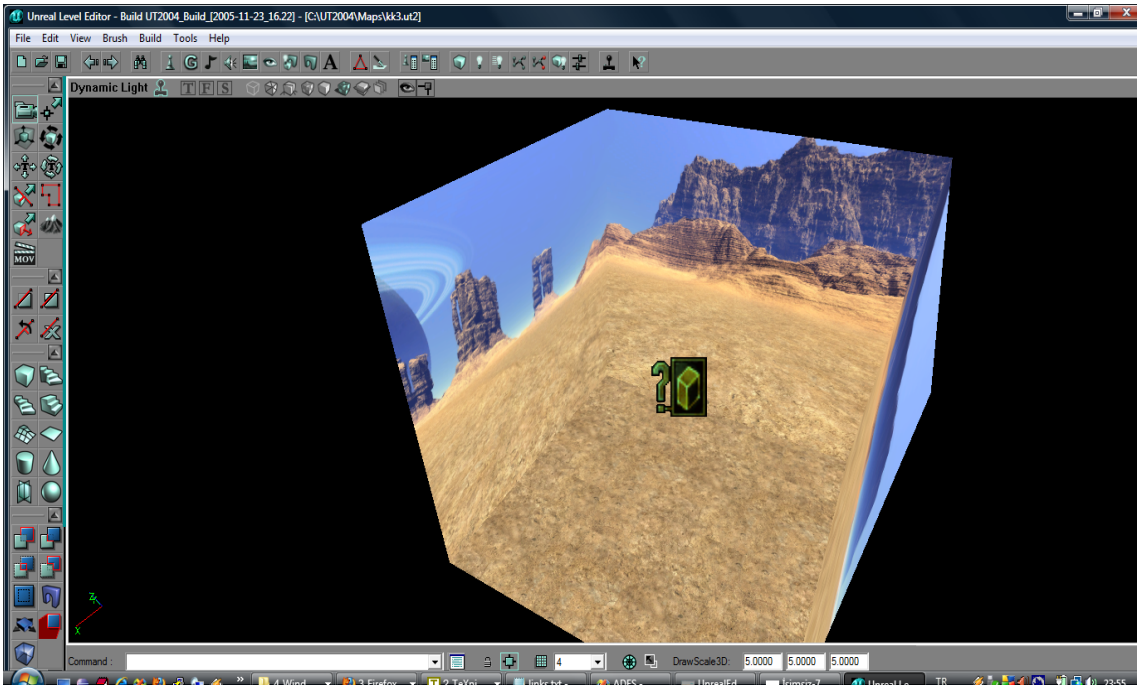


Figure 5.8. The SkyBox of the ADES Unreal world.

target after the intercepting function is called. The Detours are used for capturing some of the DirectX [92] calls. These calls are branched to the UPIS server which performs additional instructions in order to make the video memory visible to itself. The UPIS server also starts to listen to a specific TCP/IP port to serve those images to the ADES unreal controller.

5.2.5. ADES Unreal Controller

The Unreal Engine provides the simulation environment. USARSim provides various vehicles and sensors that can be reached by outside applications. The UPISEX provides client images to outside applications. The ADES Unreal Controller uses these tools in order to place a vehicle, Sedan, into the ADES Unreal world. Then the human operator controls the Sedan from the application. The application processes the image data and other sensor data and provides facts to its selected expert system. And finally, the expert system is queried after each fact assertion for a traffic violation. If the human operator violates a traffic rule, the application is expected to warn the operator by providing the type of violation.

The application interface is given in Figure 5.9. The host and the port is the address of the Unreal Engine. The image host and port is the address of the UPISEX image server. The

Figure 5.9. The Unreal ADES controller application interface.

path is the directory which contains Unreal Tournament 2004 installation. The Expert System selection can be done by using the expert drop down list. The send command button sends the string entered into the white box to the Unreal Engine. The Release/Attach Camera command attaches the client viewpoint to the vehicle's camera or return to the global camera. There are two helper buttons for initiating the Unreal Engine and the USISEX with correct parameters. These applications can also be started from the command line of the target computer.

The panel in the middle of the application displays the image from the UPISEX server. The images can be captured by the vehicle camera or the global viewpoint according to the effective camera in the simulation.

At the right hand side of the camera image, the outputs of different processing can be followed. The detected sign is given in the upper part of the application. The violation box turns to red if a violation occurs. In addition, the details of the violation are also given in this box. The text area below the violation box displays the facts asserted to or retracted from the expert system. Finally, the driver aggression meter is placed at the right bottom of the application.

5.2.6. Deprecated Simulators

Since the framework is developed by using Microsoft's Visual Studio.NET, the first simulation environment considered is Robotics Studio which is also a product of Microsoft

company. At the beginning of the project the urban challenge proposed by KIA, which provides a good simulation environment for autonomous urban driving, is available as a research platform. The simulated agent has stereo vision, GPS, bumper detectors and LIDAR sensors. And the actuators are modeled as differential drives for simplicity. However, the company discontinued the simulator which ends our efforts on the environment and the Robotics Studio.

Figure 5.10. KIA Urban Challenge

6. TESTS AND EVALUATION

The ADES project provides an end-to-end framework design and implementation for automatically detecting the traffic violations. To overcome this challenging task various modules are implemented. The performance of these modules are investigated in this section. The presented results are obtained from the real world applications of the proposed system.

The road lane detection and traffic sign recognition processes are tested on a recorded video sequence of the urban roads in Turkey under different lightning conditions. The video sequence contains more than 150 traffic signs. For the expert system tests, an artificial map created for the simulation environment is used.

6.1. Road Lane Detection

The first solution that will be considered is the road lane detection system, which is a sub-module of the vision system. The average processing time is 21.25 milliseconds for a laptop PC with Intel T2050 processor at 1.6 GHz whereas the average cost of the classical approach is 15.29 milliseconds. The proposed approach managed to detect and track at least one line in most of the sequence.

In order to validate the results of the proposed approach, the classical Hough Transform approach is also implemented and the results are compared. In the implementation of the classical approach, the same part of the image is processed using the Hough transform routine without dividing smaller partitions. The most intensive 10 lines are merged according to their r and θ values. Finally three or less candidate lines are selected as the lane markers. The major differences between the two approaches are shown in Figure 6.1. The images on the left hand side are the detected lines by the classical approach. The right hand side images are the outputs of the new approach for the same frames. The result of this comparison shows that the detection capabilities are increased and the rate of false positives are also decreased by the new approach.

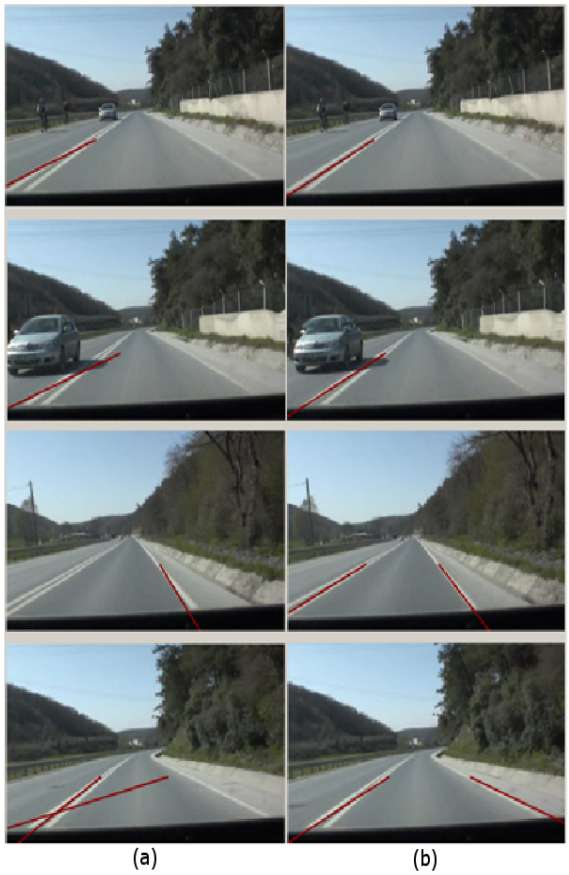


Figure 6.1. Differences between (a) classical Hough transformation and (b) the proposed approach.

6.2. Traffic Sign Detection and Recognition

The second vision module implemented within the ADES project is the traffic sign detection and recognition system.

For the detection phase, several measurements are performed. The results are given in Tables 6.1 and 6.2. The results show that the system has an efficient CPU time utilization for real life applications. Therefore, it is possible to re-run the whole process more than one hundred times in a second. This gives an opportunity to utilize the GA to track the detected sign over a sequence of frames. For this purpose, half of the best chromosomes at the end of each processed frame are passed to the next frame.

The upper rows of the Tables 6.1 and 6.2 show the parameter sets, while the lower rows (painted in gray) show the experiment results. *True detections* correspond to the correctly

Table 6.1. Detection rate of circular signs.

Fitness threshold	30	30	30	35
Population size	60	120	60	60
Number of Generations	2	2	4	2
Mutation rate	0.35	0.35	0.35	0.35
Crossover rate	0.75	0.75	0.75	0.75
Selection method	Elitist	Elitist	Elitist	Elitist
Avg. Process Time(msec)	9	14	14	9
True detections (percent)	95	96	96	65
Misses (percent)	5	4	4	35
False detections (percent)	5	7	6	1

detected signs, whereas the *misses* are the signs that could not be detected at all. Note that, the sum of the true detections and misses is always 100 percent. *False detections*, on the other hand, are the cases where the system indicates a sign existence even though there exists no sign in that location. *False detection rate* is calculated by comparing the number of false detections with the total number of real traffic signs in the video. The first column of values in the tables, is the preferred set of parameters for both the circular and the triangular cases.

For the circular signs, a fitness threshold of 30, GA population size of 60 and 2 generations yields the ideal results in terms of accuracy and CPU time. It is possible to enhance the accuracy up to 96 percent by increasing either the population size or the number of generations. But this will almost double the processing time for a very small accuracy enhancement, hence is not worth the trade-off. The fitness threshold, on the other hand, has considerable effect on the accuracy, as seen on the rightmost column.

The triangular sign process yields the ideal results with a fitness threshold of 16. That is due to the different geometric characteristics of the triangular signs. The *true detection* rate degrades from 95 percent to 87 percent. It is possible to increase this value by decreasing the fitness threshold from 16 to 12. But this has a side-effect of increasing the false detections from 4 percent to 9 percent.

Table 6.2. Detection rate of triangular signs.

Fitness threshold	16	16	16	12
Population size	60	150	60	60
Number of Generations	2	2	6	2
Mutation rate	0.35	0.35	0.35	0.35
Crossover rate	0.75	0.75	0.75	0.75
Selection method	Elitist	Elitist	Elitist	Elitist
Avg. Process Time(msec)	9	16	14	9
True detections (percent)	87	88	90	94
Misses (percent)	13	12	10	6
False detections (percent)	4	4	5	9

In the classification phase, Tables 6.3 and 6.4 exhibit the classification results on *"properly detected"* signs.

Table 6.3. Classification success rate of circular signs.

Classification Method	NN	SVM
Avg. Process Time(msec)	5	10
Error rate (percent)	9	20

Table 6.4. Classification success rate of triangular signs.

Classification Method	NN	SVM
Avg. Process Time(msec)	3	6
Error rate (percent)	7	9

Table 6.5 depicts the overall system performance, considering the detection and classification steps as a whole. The errors of the detection step have an adverse effect on the classification process. For both circular and triangular signs, NN with 12×12 grid features have yielded the best results in terms of CPU time and error rate. The visual results of the experiments and the related data set can be found on the project web site [93].

Table 6.5. Overall system performance.

	CIRCULAR	
Classification Method	NN	SVM
Avg. Process Time(msec)	14	19
Error rate (percent)	14	24
	TRIANGULAR	
Classification Method	NN	SVM
Avg. Process Time(msec)	12	15
Error rate (percent)	11	13

The comparison of the sign recognition system with existing systems developed in the last decade is given in Table 6.6. It can be seen that the overall accuracy of the system with NN classification is comparable if not better than the other approaches. Additionally, the processing time of each frame is smaller than any of the approaches even without considering the differences in the platforms.

6.3. Expert Systems

Unlike the sensor processing modules, the performance criteria of the inference engine does not depend on the execution time or accuracy. The proposed expert system implementations should be complete and easy to maintain in order to make correct judgments about the drivers' actions.

Outputs of the proposed expert systems for a speed violation scenario are given in Figure 6.2. In both examples the proposed system detects the speed violation after the required facts are provided by the sensors. In the Prolog based expert system implementation, the facts are asserted to the knowledge-base. The expired facts are also retracted from the knowledge-base before any query for the violation is done. On the other hand, in the BN based expert system, the probabilities of the related state are arranged according to the probability of the acquired sensor data. At any time during the execution of this expert system, the sum of the probability values of a specific node should be one.

Table 6.6. Comparison of the Proposed Approach with selected methods from the last decade.

Method	**Reference**	**Year**	**Hardware**	**Image Size (px)**	**Time (msec)**	**Success Rate (%)**
Matching Pursuit	Hsu and Huang [29]	2000	Intel P2	512x480	250	up to 88
Genetic Algorithm	Escalera *et al* [27]	2002	AMD Duron 1Ghz	384x286	450 (51x8.8)	N/A
Kalman Filter	Fang *et al* [28]	2003	Intel P4 1GHz	320x240	$>$1000	N/A
Feature Extraction	Gao *et al* [94]	2005	Intel P3	1680x1680	450	up to 95
SVM	Maldonado-Bascon *et al* [32]	2005	Intel P4 2.2Ghz	720x576	$>$ 1000	up to 93
Geometric Fragmentation	Soetedjo and Yamada [39]	2005	Intel P4 2.6 GHz	240x180	750	up to 86
Haar Wavelet Features	Bahlmann *et al* [30]	2005	Intel Xeon 2.8Ghz	384x288	100	85
Neural Networks	Nguwi and Kouzani [95]	2006	Pentium M 1.7Ghz	N/A	550	up to 95
Adaboost	Baro *et al* [96]	2007	N/A	60x60 stereo	$>$ 1000	87-90
Color Distance Transform	Ruta *et al* [42]	2007	N/A	640x480	35	up to 85
Affine Trans., GA, NN	Proposed Approach	2010	Intel T5450 1.66Ghz	512x288	13	87

Prolog based expert system

```
Exec: assert("1.0000::T::sign_detected(speed_limit,30)").
Exec: retractall(velocity_exceeds(X)).
Exec: retractall(velocity_exceeds(50)).
Exec: assert("1::T::velocity_exceeds(50)").
Exec: violation(V,A,P).
Exec: retractall(gps_node_property(speed_limit,30)).
Exec: assert("0.9577::T::gps_node_property(speed_limit,30)").
Exec: retractall(velocity_exceeds(X)).
Exec: retractall(velocity_exceeds(50)).
Exec: assert("1::T::velocity_exceeds(50)").
Exec: violation(V,A,P).
P = 0.957698767978062
V = speed_limit
A = '50_30'
```

BN based expert system

```
SpeedLimit:Vehicle_Speed()->(0,1,0,0)
SpeedLimit belief: 0.5436
SpeedLimit:GPS_Node_Property()->(0.0333333,0.0333333,0.0333333,0.9)
SpeedLimit:Sign_Detected()->(1,0,0,0)
SpeedLimit:Vehicle_Speed()->(0,1,0,0)
SpeedLimit belief: 0.5436
SpeedLimit:GPS_Node_Property()->(0.9708,0.00973333,0.00973333,0.00973333)
SpeedLimit:Vehicle_Speed()->(0,1,0,0)
SpeedLimit belief: 0.9491
```

Figure 6.2. Speed violation outputs of proposed expert systems.

In order to validate the probabilities of the detected violations, the formulation of the probability calculations for both systems should be examined. The probability calculation of the Prolog based system is given in Equation 6.1

$$p_v = (p_d \times \alpha_d)(p_s \times \alpha_s)(p_g \times \alpha_g) \tag{6.1}$$

where p_v is the probability of the violation. p_d, p_s, and p_g are the certainty of velocity reading from the car, the detected sign, and the GIS information respectively. In the current implementation, all facts in the knowledge-base are valid for a predefined time period. The α values are the decaying factors of the related fact. For the given example the small difference between the multiplication of the detection probabilities and the violation probability is due to the time elapsed. Although it is not easy to verify the exact probability of the violation, the multiplication of the detection probabilities gives an upper bound for the violation probabil-

ity. According to the performed experiments and measurements, the time complexity of the Prolog based expert system is O(n.log(n)). The predicate list for the proposed four violations are queried in less than 10 milliseconds.

However, artificially asserted 400 rules for violation predicate are queried at nearly 3500 milliseconds. Although the measured execution times are feasible for real time operation in the proposed system, improvements should be considered if the number of violation rules are increased.

The calculation of the violation probability for the BN based expert system is somewhat more complicated. The probability assessments introduced by the human expert should be considered while calculating the violation probability. For the given example the violation probability is calculated by Equation 6.2

$$p_v = \sum_{d \in Sign\_Detected, s \in Vehicle\_Speed, g \in GPS_Node_Property} p_d \times p_s \times p_g \times vp(d, s, g) \qquad (6.2)$$

where then definitions of p_d, p_s, and p_g are as explained previously, $vp(d, s, g)$ is the predefined probability assessment value at the yes state of the violation node for the detected sign, velocity and GPS node. And finally, the s, d, and g elements are the states of the corresponding node in the BN graph. For the previous example, the violation probability is calculated in the Equation 6.3

$$\begin{aligned} p_v &= 0.9708 \times 1 \times 1 \times 0.9570 \qquad (6.3) \\ &\quad +0.00973 \times 1 \times 1 \times 0.7660 \\ &\quad +0.00973 \times 1 \times 1 \times 0.7870 \\ &\quad +0.00973 \times 1 \times 1 \times 0.5110 \\ p_v &= 0,94913832 \end{aligned}$$

where the last multiplicand of each line is the probability assessment value for the violation node. Please note that the lines with zero p_d and p_s values are omitted for readability. The belief network calculation may differ for other traffic regulations due to the structure of the network. For example, since the post nodes in directional regulations depend on pre nodes, a

similar calculation should be performed for the post nodes before finding the violation probability.

As a result of the implementations performed on the simulation environment, it can be seen that the proposed expert system implementations do not require extensive efforts for constructing and maintaining. This is due to the simplicity of the existing traffic rules. However, for more complicated rules the performance of the system depends on the human expert who introduces the predefined rules to the inference engine.

7. CONCLUSION AND RECOMMENDATIONS

The social and economic impact of the traffic accidents are increasing every day. Fortunately the number of institutes and efforts to prevent traffic accidents and compensate their damages are also increasing day by day. Most of the countries develop their traffic safety programs according to the 4E's of safety which are engineering, enforcement, education, and emergency. Engineering step deals with the conditions of roads and highways, proper settlements of signs and other traffic indicators. Education step is the proper public access to the basic rules and important knowledge about traffic. Emergency part is the planning and improvement of the actions which should be taken after an accident occurs. And the enforcement step adapts the road traffic legislation and the way it is enforced. Since most of the accidents are caused by the drivers, education and enforcement becomes more important for preventing the traffic accidents.

7.1. The Main Conclusions

In the ADES project we addressed the problem of developing an application framework for automatic detection of the violations performed by the drivers. The problem is considered in two main sections. The first section is the acquisition of the raw data from different sensors and processing these data to obtain the valuable information which can be transferred to the inference engine. Although there are numerous kinds of sensors available in the market, we focused on camera, RF reader and GPS devices since these devices that are relatively easy to obtain and already used by various systems in the industry. The algorithms used in the vision system are tested upon real images where the RF and GPS systems are modeled in the simulation environment. The second section of the problem is the design of the inference engine. The rules of data flow between the sensors and the inference engine are defined by an interface which should be implemented by any candidate expert system. The valuable information acquired from the sensors are introduced to the inference engine as facts. The inference engine uses its internal rules and definitions plus these facts to make a judgment about the actions of the driver. The final decision of the inference engine is output by the system with adequate details.

Although this problem is not addressed before in the literature, to the best of our knowledge, there are many applications of ADAS systems are on the market. The aim of the ADES project **is not** to compete with these implementations. On the contrary, our aim is to present a framework where these enhancements can easily be plugged in. This modularity requires a flexible system design where abstraction should be carefully utilized in the software. The integration of various ADAS systems to the proposed solution requires implementation of a flexible and easy to maintain expert system. The proposed expert system tries to identify the challenges on the way and present feasible solutions to these problems. The presented expert system implementations give examples of both rule based and probabilistic models. Although the addressed traffic rules are not very complicated, they are the most important regulations for the road safety. In addition, the proposed implementations give a roadmap for extending the system for including other types of the traffic rules.

The methodology of the proposed system presents various novelties. The selected point based image binarization, the MHT-HMM lane detection, the GA and affine transformation based sign detection mechanism, the RFID and GPS data structures, the enhanced Prolog engine with probabilistic and time-decaying facts, the simulator environment for driver fault detection are all original artifacts of the ADES project. Finally, the performance of the proposed system with all these enhancements is evaluated to be feasible for the selected purpose. The tests performed in the developed applications shows that the system is capable of detecting traffic violations. Obviously there are some assumptions and prerequisites for performing this challenging task. This is the main reason why we think the ADES project as an developable framework instead of a packaged solution.

7.2. Recommendations for Future Work

The proposed system presents an end-to-end implementation for the simulated environment. Some applications for real environment are also presented. However, there are areas which need more research for the project. The topics for these research areas are,

- End-to-end implementation of the application for the real environment: The current state of the project provides an end-to-end solution for the simulation environment.

Although the source feed can be easily changed to a real camera input, certain amount of configurations should be performed to set up the system.

- Self modifying, or learning, inference engine: The proposed expert systems require predefined rules and assignment to perform their operations. The automation of mining these rules will be a valuable improvement to the proposed system.
- Improved sensor processor modules for given modules: Better sensors and better processing algorithms are the inevitable requirements for improving the system. A vision processor which can handle bad weather conditions, or a more precise GPS system will obviously increase the performance of the system.
- Integration of advance vision processing modules: Vision is the heart of the system while interacting with the real environment. GPS or RF technologies cannot provide information about the dynamic environment without additional deployments. Therefore the vision data should be processed more intensively to understand the environment. The traffic lights, the pedestrians, other vehicles on the road, unexpected obstacles, signs printed on the roads, and many other features can be extracted from the vision data. The advanced vision processing modules may also be capable of detecting traffic signs in adverse weather conditions.
- Wireless integration of the system to the traffic control center: Although the goal of this project is informing the driver about his or her violations, a traffic control center like a police center, can be notified about the drivers actions.
- More vehicle to vehicle and vehicle to infrastructure communication: In the near future the vehicles will be more intelligent and possess various means to communicate with their environments. The proposed system may benefit from this communications by obtaining more information about the environment. For example, if a vehicle detects a sign it can broadcast this information to the other vehicles.
- Integration with existing intelligent highway systems: In most of the advanced cities there are already deployed traffic control mechanisms. Integration with these systems may be useful for both parties.

APPENDIX A: ADES Software Design Documentation

This design document presents the details of the software implemented for the ADES project. The document follows the guidelines given in the IEEE Computer Society's Recommended Practice for Software Design Descriptions [97]. The references and acronyms sections are omitted from the document because these sections are already given in the thesis documentation. However, the document is enriched by the section about the software development methodology.

A.1. INTRODUCTION

A.1.1. Scope

This document contains the design description of the Automatic Driver Evaluation System software. The description contains the architectural features of the proposed system layout where the operations of each implemented module is explained in detail. The primary audience of this document is the software engineer. ADES software provides a library for implementing a framework for processing the data acquired from different sensors. In addition, sample applications for real life and simulated environment examples. The system is developed by the ADES project group of Boğaziçi University Artificial Intelligence Laboratory for research purposes.

A.1.2. Document Overview

This software design description document is organized into the following sections:

- Introduction: The scope and the outline of the document is presented.
- Decomposition Description: The main modules of the software and their duties are explained.
- Dependency Description: The main dependencies of the software is listed.
- Interface Description: The details of the user interface for the end user is given.
- Detailed Design: The low level details are documented.

- Software Development Methodology: The tools and techniques used for the development process are explained.

A.2. DECOMPOSITION DESCRIPTION

A.2.1. Module Decomposition

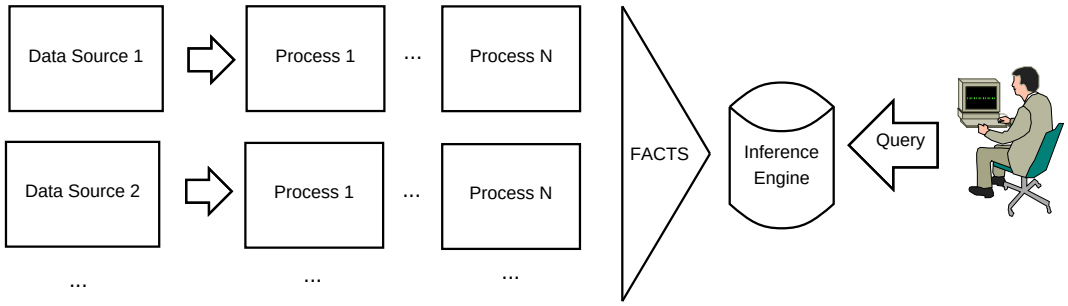


Figure A.1. The module decomposition of ADES project.

The main modules used in the ADES project can be grouped in three categories. The first category contains the source classes derived from message initiator base class which has start and stop features. The source module is responsible for providing information from the outside world to the application.

The second module is the processor layer. There are various processor classes for different purposes. They are all extended from the message consumer base class which has a method for message consuming. All processor classes also provide a function for exposing the message types they can process. This message type is presented in the construction of these classes.

The last module is the inference engine. The proposed expert system implementations for the inference engine should implement the expert system interface as explained in the thesis documentation.

A.2.2. Process Decomposition

The process decomposition of the project is based on the process implementation classes. The processor classes consumes the messages provided by the source module. The imple-

mented processors are,

- HistogramProcessor: Finds the histogram of the captured image for improving the binarization process.
- AForgeProcessor: A generic processor for utilizing the AForge image library features.
- LaneDetectorProcessor: MHT and HMM base lane detector processor.
- CircularSignDetectorProcessor: Processor for detecting circular signs.
- TriangularSignDetectorProcessor: Processor for detecting triangular signs.
- HoughLineProcessor: Hough line processor for classical lane detection method.
- AutoBrightnessProcessor: Automatic brightness correction processor.
- SVM_SURFProcessor: Sign classification processor uses SVM based on SURF features or grid based features.
- NN_Processor: Sign classification processor uses NN based on grid based features.
- NN_SURFProcessor: Sign classification processor uses NN based on SURF features.

A.2.3. Data Decomposition

The data flow of the ADES project is controlled by the message flow class. The bindings between the source objects and the processor objects are declared by adding those objects to the message flow. The messaging system provides two types of services. The first one is the basic message service which simply transfers the acquired message to the target object. The second one, which is the message dispatcher, creates a copy of the acquired message for all registered target objects.

The messages implement the message interface which defines the basic requirements for a message object. There are four types of messages in the current implementation of the system.

- Generic Message: Any basic message format. All messages can be converted to this type for serialization purposes.
- UDP Message: Carries information in distributed deployments of the processors and source classes.
- Vision Message: The basic image information from the source objects.

- Text Message: Text based information provided by the source objects.

A.3. DEPENDENCY DESCRIPTION

ADES projects depends on several projects. Most of these projects are open source. The brief descriptions of these projects, their usages and web addresses are as follows.

- AForge.NET: This is a CSharp framework designed for developers and researchers in the fields of Computer Vision and Artificial Intelligence. Various components of this project is used in the ADES project for implementing neural networks, genetic algorithms and vision processing. The source codes can be achieved from the project home page, which is, *http://www.aforgenet.com*.
- Accord.NET: This is another CSharp framework extending the excellent AForge.NET Framework external with new tools and libraries. Accord.NET provides many algorithms for many topics in mathematics, statistics, machine learning, artificial intelligence and computer vision. The Levenberg-Marquardt implementation of this library is used in the ADES project. The web page from the project is *http://accord-net.origo.ethz.ch* where the source codes can be found.
- ALGLIB: ALGLIB is a cross-platform numerical analysis and data processing library. It supports several programming languages (C++, CSharp, Pascal, VBA) and several operating systems (Windows, Linux, Solaris). The LDA implementation of the ALGLIB project is used in the ADES project. The project can be accessed at *http://www.alglib.net*.
- DirectShowNet Library: The purpose of this library is to allow access to Microsoft's DirectShow functionality from within .NET applications. We used this library to read video files and acquiring live camera images. Detailed information about the project can be found at *http://directshownet.sourceforge.net*.
- OpenSURF: OpenSURF is an implementation of SURF (Speeded Up Robust Features) which detects landmark points in an image, and describe the points by a vector. The surf features in the ADES project are detected by the CSharp port of the OpenSurf library which can be found at *http://www.chrisevansdev.com/computer-vision-opensurf.html*.
- SVM.NET: This is a .NET conversion of libsvm project, which is is an integrated software for support vector classification, regression, and distribution estimation. The .NET

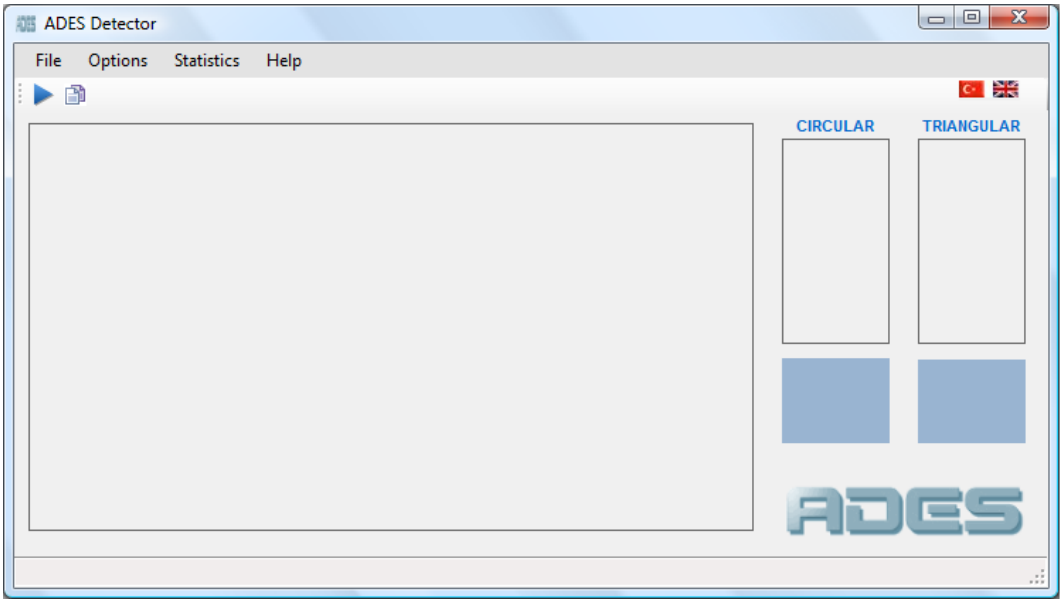


Figure A.2. ADES Detector.

port can be found at http://www.matthewajohnson.org/software/svm.html. The project is used for SVM based classification of the traffic signs.

- USARSim: USARSim is a high-fidelity simulation of robots and environments based on the Unreal Tournament game engine. The details for the usage of the USARSim library is given in the thesis documentation. The project homepage is *http://usarsim.sourceforge.net*.
- CSharpProlog: A CSharp based open source Prolog engine. We modified the engine for implementing the time-decaying facts and the probabilistic facts. The original project can be found at *http://cs-prolog.sourceforge.net/*.
- MSBNx: MSBNx is a component-based Windows application for creating, assessing, and evaluating Bayesian Networks, created at Microsoft Research *http://research.microsoft.com*.

A.4. INTERFACE DESCRIPTION

A.4.1. ADES Detector

The real world tests are performed by the ADES detector application. The menu options of the application are;

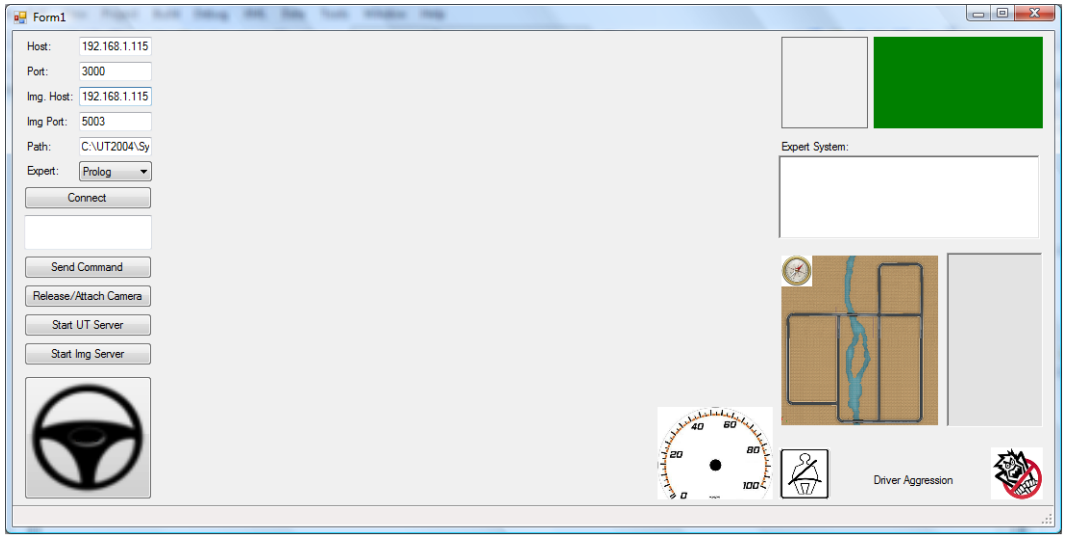


Figure A.3. ADES Unreal Controller.

- File/SVM/Train: Initiates a SVM training process. Options for the training process are defined in Constants class.
- File/NN/Train: Initiates a SVM training process. Options for the training process are defined in Constants class.
- File/LDA/Generate Feature Array: Selects the most important features according to the LDA analysis.
- File/LDA/Use LDA Features: Forces system to use the detected LDA features.
- Options/Lane Detection: Invoke lane detection process.
- Statistics: Show statistics log file.
- Help: Help menu (to be implemented).

The buttons on the window are for starting the entire process and dumping the statistic data to the log file. The detected, and recognized signs are displayed at the right side of the window. The detected road lanes are directly drawn within the processed video sequence.

A.4.2. ADES Unreal Controller

The ADES unreal controller has a simple but efficient interface for starting the simulation environment and the client application. The definition of the user interface components are,

- Host (Text Box): The TCP IP address of the simulation engine.
- Port (Text Box): The TCP port of the simulation engine.
- Img. Host (Text Box): The TCP IP address of the UPISEX application.
- Img. Port (Text Box): The TCP port address of the UPISEX application.
- Path (Text Box): The path for the executables of the simulation engine.
- Expert (Combo Box): The expert system should be selected before connecting the agent.
- Connect (Button): Spawns a new vehicle agent in the simulation environment.
- Unnamed Text Area: Custom USARSim command area.
- Send Command (Button): Sending custom USARSim command to the simulation engine.
- Release/Attach Camera (Button): Active camera selection. The global camera or the camera attached to the vehicle is selected alternatively.
- Start UT Server (Button): Starts the simulation engine.
- Start Img Server (Button): Starts the client application.
- Steering Wheel (Picture Button): The vehicle can be connected via W, A, S, and D buttons, after the vehicle is connected to the simulation environment.
- Speedometer (Picture Box): Displays the speed of the vehicle.
- Sign Box (Picture Box): The detected sign.
- Inference Result (Colored Label): The result of the violation query from the inference engine.
- Expert System (Text Area): The message log for expert system.
- Map (Picture Box): The position of the vehicle according to the GPS system.
- Unnamed Disabled Text Area: The GPS and GIS information.
- Driver Aggression (Progress Bar): Current aggression coefficient of the driver.

A.4.3. ADES Sign Recognition Test Utility

A very simple test utility for evaluating the sign recognition strategies is developed. The user simply selects a recorded picture, and initiates the sign recognition process. The system displays the recognized sign at the left of the window, it manages to detect a traffic sign.

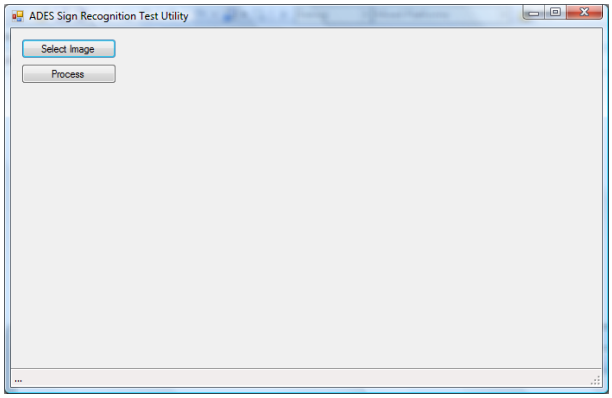


Figure A.4. ADES Sign Recognition Test Utility.

A.5. DETAILED DESIGN

The entire class hierarchy diagram of the ADES project is given in Figure A.5. Please note that some utility classes are removed from the diagram for simplicity.

A.5.1. Class Reference for BOUNLib Namespace

The BOUNLib.Processors.AForgeProcessor class is used as a generic processor for utilizing the AForge image library features. This class inherits Messaging.MsgConsumer. The public member functions of this class are as follows.

- override void consumeMessage (IMessage message, int msgID): Consumes the provided message.

The Processors.AutoBrightnessProcessor class is used as an automatic brightness correction processor. This class inherits Messaging.MsgConsumer. The public member functions of this class are as follows.

- override void consumeMessage (IMessage message, int msgID): Consumes the provided message.

The Messaging.BasicMsgService class simply transfers the acquired message to the target object. This class inherits Messaging.MsgService. The public member functions of this

Figure A.5. ADES class hierarchy diagram.

class are as follows.

- override void sendMsg (IMessage msg): Send message to the target.

The ES.BeliefNetworkES class is used as the Belief Networks based ES implementation. This class inherits ES.ExpertSystems. The public member functions of this class are as follows.

- string init (params object[] esParams): Initializes the expert system.
- string assertFact (params object[] esParams): Fact assertion.
- string retractFact (params object[] esParams): Fact retraction.
- string[] query (params object[] esParams): Violation query feature.
- void setThreshold (double threshold): The violation threshold setter for driver aggressiveness.
- double getThreshold (): Violation threshold getter.

The Sources.BitmapMemoryVisionSource class reads the vision information from memory. This class inherits Messaging.MsgInitiator. The public member functions of this class are as follows.

- override void start (): Starts the information flow.
- override void stop (): Stops the information flow.

The static ToolBox.ByteTools class is used for various byte operations. The public member functions of this class are as follows.

- static int intToByteArr (int data, byte[] outBuf, int pos): Converts integer to byte array.
- static int byteArrToByteArr (byte[] inBuf, byte[] outBuf, int pos): Copies a subarray of a byte array.
- static int intFromByteArr (byte[] inBuf, int pos): Extracts integer from byte array.
- static byte[] byteArrFromByteArr (byte[] inBuf, int pos, int size): Returns byte array from a subarray of a byte array.
- static byte[] BmpToBytes (Bitmap bmp, PixelFormat pf): Gets byte array from Bitmap.

- static Bitmap BytesToBmp (byte[] bmpBytes, int bmpWidth, int bmpHeight, PixelFormat pf): Constructs Bitmap from byte array.
- static PixelFormat bppToPixelFormat (int bpp): Bits per pixel to PixelFormat.
- static int pixelFormatToBPP (PixelFormat pf): PixelFormat to bits per pixel integer.
- static void imageCoM (Bitmap bmp, ref int com_x, ref int com_y): Center of Mass calculation for an image.

The Processors.CircularSignDetectorProcessor class is used as a processor for detecting circular signs. This class inherits Messaging.MsgConsumer. The public member functions of this class are as follows.

- override void consumeMessage (IMessage message, int msgID): Consumes the provided message.

The NET.ImageFilters.ColorLabelFilter class is used as a custom AForge filter for color labeling.

The static Constants class contains options for processors. The public member functions of this class are as follows.

- static float getAngle (int X1, int Y1, int X2, int Y2): Gets angle with atan(X2-X1/Y2-Y1).
- static double DIST (int x1, int x2, int y1, int y2): Euclidean distance.
- static double DIST (Point p1, Point p2): Euclidean distance.
- static double MAX (double[] inArr, ref int idx): Gets MAX of array.
- static double MIN (double[] inArr, ref int idx): Gets MIN of array.
- static double getColorValForLabeling (Color clr): Arranges labeling color. Static Public Attributes
- static string base_folder: The base folder definition for training files and system outputs.
- static short IMAGE_WIDTH: The width of the image.
- static short IMAGE_HEIGHT: The height of the image.

The ToolBox.CRC8Calc class is used as a Class for calculating CRC8 checksums.

The Sources.DeviceVisionSource class reads the vision information from a connected camera. This class inherits Messaging.MsgInitiator. The public member functions of this class are as follows.

- override void Dispose (): Release everything.
- override void start (): Starts the information flow.
- override void stop (): Stops the information flow.

The Messaging.DispatchMsgService class creates a copy of the acquired message for all registered target objects. This class inherits Messaging.MsgService. The public member functions of this class are as follows.

- override void sendMsg (IMessage msg): Send message to the target.

The ES.ExpertSystems Interface should be used by the expert system implementations Inherited by ES.BeliefNetworkES, and ES.PrologES. The public member functions of this class are as follows.

- string init (params object[] esParams): Initializes the expert system.
- string assertFact (params object[] esParams): Fact assertion. item string retractFact (params object[] esParams): Fact retrection.
- string[] query (params object[] esParams): Violation query feature.
- void setThreshold (double threshold): The violation threshold setter for driver aggressiveness.
- double getThreshold (): Violation threshold getter.

The static NET.Toolbox.FileTools class is used for the file operations. The public member functions of this class are as follows.

- static List of type FileInfo getTrainingFiles (ref int class_count): Gets training files for NN and SVM.
- static void binarySerialize (String filename, object o): Writes a binary file to disk.
- static object binaryDeserialize (String filename): Reads a binary file from disk.

The Sources.FileVisionSource class reads the vision information from video file. This class inherits Messaging.MsgInitiator.The public member functions of this class are as follows.

- override void Dispose (): Release everything.
- override void start (): Starts the information flow.
- override void stop (): Stops the information flow.

The Messages.GenericMessage class is used as the basic message format. All messages can be converted to this type for serialization purposes. This class inherits Messages.IMessage.

The Processors.GeoTransChromosome class is used as the genetic algorithm processing class.

The static Globals class contains global options for logging. The public member functions of this class are as follows.

- static void init (): Intializes the options with default values.

The Processors.HistogramProcessor class finds the histogram of the captured image for improing the binarization process. This class inherits Messaging.MsgConsumer. The public member functions of this class are as follows.

- override void consumeMessage (IMessage message, int msgID): Consumes the provided message.

The Processors.HoughLineProcessor class is used as the Hough line processor for classical lane detection method. This class inherits Messaging.MsgConsumer. The public member functions of this class are as follows.

- override void consumeMessage (IMessage message, int msgID): Consumes the provided message.

The Messages.IMessage Interface is the messaging interface implemented by message classes. Inherited by Messages.GenericMessage, Messages.TextMessage, Messages.UDPMessage, and Messages.VisionMessage.

The Processors.LaneDetectorProcessor class is used as the MHT and HMM base lane detector processor. This class inherits Messaging.MsgConsumer. The public member functions of this class are as follows.

- override void consumeMessage (IMessage message, int msgID): Consumes the provided message.

The NET.ImageFilters.LaneFilter class is used as a custom AForge filter for lane detection.

The NET.Toolbox.LDA class is used for the LDA Calculation from training files.

The ToolBox.Logger class is used as statistics logger.

The Messaging.MessageFlow class is used as the main message flow binding object. The public member functions of this class are as follows.

- void addConsumer (MsgConsumer item): Register consumers.
- void addInitiator (MsgInitiator item): Register sources.
- void startFlow (): Triggers the sources to start.
- void stopFlow (): Stops the flow.

The Messaging.MsgConsumer class is used as the interface definition for processor classes. Inherited by Processors.AForgeProcessor, Processors.AutoBrightnessProcessor, Processors.CircularSignDetectorProcessor, Processors.HistogramProcessor, Processors.HoughLineProcessor, Processors.LaneDetectorProcessor, Processors.NN_Processor, Processors.NN_SURFProcessor, Processors.PanelDisplayProcessor, Processors.SVM_SURFProcessor, Processors.TextMsgDisplayProcessor, and Processors.TriangularSignDetectorProcessor. The public member functions of this class are

as follows.

- Type getMsgType (): Returns the message type that the processor handles.
- abstract void consumeMessage (IMessage message, int msgID): Consumes the provided message.

The Messaging.MsgInitiator class is used as the interface definition for source classes. Inherited by Sources.BitmapMemoryVisionSource, Sources.DeviceVisionSource, and Sources. FileVisionSource. The public member functions of this class are as follows.

- abstract void start (): Starts the information flow.
- abstract void stop (): Stops the information flow.

The Messaging.MsgService class is used as the base message service class. Inherited by Messaging.BasicMsgService, Messaging.DispatchMsgService, and UDPServices.UDPMsgService. The public member functions of this class are as follows.

- abstract void sendMsg (IMessage msg): Send message to the target.
- virtual void receiveMsg (byte[] msg): Receive message from source.

The Processors.NN_Processor class is used as the sign classification processor uses NN based on grid based features. This class inherits Messaging.MsgConsumer. The public member functions of this class are as follows.

- override void consumeMessage (IMessage message, int msgID): Consumes the provided message.

The Processors.NN_SURFProcessor class is used as the sign classification processor uses NN based on SURF features. This class inherits Messaging.MsgConsumer. The public member functions of this class are as follows.

- override void consumeMessage (IMessage message, int msgID): Consumes the provided message.

The NET.Toolbox.NNTrain class is used for the neural network training process.

The ES.PrologES class is used as Prolog based ES implementation. This class inherits ES.ExpertSystems. The public member functions of this class are as follows.

- string init (params object[] esParams): Initializes the expert system.
- string assertFact (params object[] esParams): Fact assertion.
- string retractFact (params object[] esParams): Fact retrection.
- string[] query (params object[] esParams): Violation query feature.
- void setThreshold (double threshold): The violation threshold setter for driver aggressiveness.
- double getThreshold (): Violation threshold getter.

The NET.ImageFilters.SignFilter class is used as a custom AForge filter for sign detection.

The static ToolBox.Statistics class contains statistics functions. The public member functions of this class are as follows.

- static double GetVariance (int[] data): Get variance.
- static double GetStdev (int[] data): Get standard deviation.

The NET.Toolbox.SURF class is used as the OpenSURF integration.

The Processors.SVM_SURFProcessor class is used as the sign classification processor uses SVM based on SURF features or grid based features. This class inherits Messaging. MsgConsumer. The public member functions of this class are as follows.

- override void consumeMessage (IMessage message, int msgID): Consumes the provided message.

The NET.Toolbox.SVMTrain class is used for the SVM training process.

The Messages.TextMessage class is the text based information provided by the source objects. This class inherits Messages.IMessage.

The Processors.TriangularSignDetectorProcessor class is used as the processor for detecting triangular signs. This class inherits Messaging.MsgConsumer. The public member functions of this class are as follows.

- override void consumeMessage (IMessage message, int msgID): Consumes the provided message.

The UDPServices.UDPAddress class is used for UDP connection address definition.

The UDPServices.UDPClient class is used for UDP client implementation.

The Messages.UDPMessage class carries information in distributed deployments of the processors and source classes. This class inherits Messages.IMessage.

The UDPServices.UDPMsgService class is used as the UDP Messaging service. This class inherits Messaging.MsgService.Classes. The public member functions of this class are as follows.

- override void sendMsg (IMessage msg): Send message to the target.

The UDPServices.UDPServer class is used as the UDP server implementation.

The Messages.VisionMessage class is used as the basic image information from the source objects. This class inherits Messages.IMessage.

A.6. SOFTWARE DEVELOPMENT METHODOLOGY

Since the ADES project is developed by a team of engineers, basic software development tools and methods are applied. The most important component of these tools is the SVN repository which keeps the latest versions and all history of the codes. The svn repository

of the project can be accessed via *http://robot.cmpe.boun.edu.tr/svn/ADES/*. Since this is a restricted site, the authorization requests should be submitted to the project administrators. The details about the SVN can be accessed at *http://subversion.apache.org/*.

Another tool used during the development of the project is Redmine, which is a flexible project management web application. Redmine is open source and released under the terms of the GNU General Public License v2 (GPL). Detailed information about the Redmine can be found in *http://www.redmine.org*.

REFERENCES

1. Peden, M. *et al.*, *World Report on Road Traffic Injury Prevention*, World Health Organization, Geneva, 2004.

2. European Commission Directorate-General for Energy and Transport, *Halving the Number of Road Accident Victims in the European Union by 2010: A Shared Responsibility*, 2001, `http://ec.europa.eu/transport/road_safety/observatory/doc/memo_rsap_en.pdf`, June 2011.

3. T.C. Sayıştay Başkanlığı, *Performans Denetimi Raporu: Trafik Kazalarını Önleme Faaliyetleri*, 2008, `http://www.sayistay.gov.tr/rapor/rapor4.asp?id=78`, June 2011.

4. T.C. Sayıştay Başkanlığı, *Daimi veya Geçici Sürelerle Geri Alınan Sürücü Belgesi Sayısının Yıllara Göre Dağılımı*, 2010, `http://www.trafik.gov.tr/istatistikler/gerial.asp`, June 2011.

5. T.C. Sayıştay Başkanlığı, *Genel Kaza İstatistikleri*, 2010, `http://trafik.gov.tr/istatistikler/10_yil_istatistik.asp`, June 2011.

6. Rumar, K., "The Role of Perceptual and Cognitive Filters in Observed Behaviour", *Human Behaviour and Traffic Safety*, pp. 151–170, 1985.

7. Finkenzeller, K., *Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication*, Wiley, Chichester West Sussex; Hoboken NJ, 3rd edn., 2010.

8. Kaplan, K., C. Kurtul and H. L. Akın, "Fast Lane Tracking for Autonomous Urban Driving Using Hidden Markov Models and Multiresolution Hough Transform", *Industrial Robot: An International Journal*, Vol. 37, No. 3, pp. 273–278, 2010.

9. Cooper, G., "The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks", *Artificial Intelligence*, Vol. 42, No. 2-3, pp. 393–405, 1990.

10. Vahidi, A. and A. Eskandarian, "Research Advances in Intelligent Collision Avoidance and Adaptive Cruise Control", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 4, No. 3, pp. 143–153, 2003.

11. Dagan, E., O. Mano, G. P. Stein and A. Shashua, "Forward Collision Warning with a Single Camera", *IEEE Intelligent Vehicles Symposium*, pp. 37–42, 2004.

12. Risack, R., N. Mohler and W. Enkelmann, "A Video-Based Lane Keeping Assistant", *IEEE Intelligent Vehicles Symposium*, pp. 356–361, 2000.

13. Schofield, K. and N. Lynam, "Vehicle Blind Spot Detection Display System", US Patent 5786772, July 1998.

14. Torresen, J., J. Bakke and L. Sekanina, "Efficient Recognition of Speed Limit Signs", *IEEE Conference on Intelligent Transportation Systems*, pp. 652–656, 2004.

15. Ji, Q., "Real-Time Eye, Gaze, and Face Pose Tracking for Monitoring Driver Vigilance", *Real-Time Imaging*, Vol. 8, No. 5, pp. 357–377, 2002.

16. Biswas, S., R. Tatchikou and F. Dion, "Vehicle-to-Vehicle Wireless Communication Protocols for Enhancing Highway Traffic Safety", *IEEE Communications Magazine*, Vol. 44, No. 1, pp. 74–82, 2006.

17. Hough, P., "Methods and Means for Recognizing Complex Patterns", US Patent 3069654, December 1962.

18. Li, Q., N. Zheng and H. Cheng, "Springrobot: A Prototype Autonomous Vehicle and Its Algorithms for Lane Detection", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 5, No. 4, pp. 300–308, 2004.

19. Yu, B. and A. Jain, "Lane Boundary Detection Using a Multiresolution Hough Transform", *International Conference on Image Processing*, p. 748, 1997.

20. Pomerleau, D., "Neural Network Vision for Robot Driving", *The Handbook of Brain Theory and Neural Networks*, 1996.

21. Kang, D. J. and M. H. Jung, "Road Lane Segmentation Using Dynamic Programming for Active Safety Vehicles", *Pattern Recognition Letters*, Vol. 24, No. 16, pp. 3177–3185, 2003.

22. Wang, Y., E. K. Teoh and D. Shen, "Lane Detection and Tracking Using B-Snake", *Image and Vision Computing*, Vol. 22, No. 4, pp. 269–280, 2004.

23. Kreucher, C., S. Lakshmanan and K. Kluge, "A Driver Warning System Based on the LOIS Lane Detection Algorithm", *Proceedings of IEEE International Conference on Intelligent Vehicles*, pp. 17–22, 1998.

24. Apostoloff, N. and A. Zelinsky, "Robust Vision Based Lane Tracking Using Multiple Cues and Particle Filtering", *IEEE Intelligent Vehicles Symposium*, pp. 558–563, 2003.

25. Zhou, Y., R. Xu, X. Hu and Q. Ye, "A Robust Lane Detection and Tracking Method Based on Computer Vision", *Measurement Science and Technology*, Vol. 17, p. 736, 2006.

26. McCall, J. and M. Trivedi, "Video-Based Lane Estimation and Tracking for Driver Assistance: Survey, System, and Evaluation", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 7, No. 1, pp. 20–37, 2006.

27. De la Escalera, A., L. Moreno, M. Salichs and J. Armingol, "Road Traffic Sign Detection and Classification", *IEEE Transactions on Industrial Electronics*, Vol. 44, No. 6, pp. 848–859, December 1997.

28. Fang, C., S. Chen and C. Fuh, "Road-sign Detection and Tracking", *IEEE Transactions on Vehicular Technology*, Vol. 52, No. 5, pp. 1329–1341, 2003.

29. Hsu, S. H. and C. L. Huang, "Road Sign Detection and Recognition Using Matching Pursuit Method", *Image and Vision Computing*, Vol. 19, No. 3, pp. 119–129, 2001.

30. Bahlmann, C., Y. Zhu, V. Ramesh, M. Pellkofer and T. Koehler, "A System for Traffic

Sign Detection, Tracking, and Recognition Using Color, Shape, and Motion Information", *IEEE Intelligent Vehicles Symposium*, pp. 255–260, 2005.

31. Loy, G. and N. Barnes, "Fast Shape-Based Road Sign Detection for a Driver Assistance System", *International Conference on Intelligent Robots and Systems*, Vol. 1, pp. 70–75, 2004.

32. Maldonado-Bascon, S., S. Lafuente-Arroyo, P. Gil-Jimenez, H. Gomez-Moreno and F. Lopez-Ferreras, "Road-Sign Detection and Recognition Based on Support Vector Machines", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 8, No. 2, pp. 264–278, 2007.

33. Fleyeh, H., "Road and Traffic Sign Color Detection and Segmentation-A Fuzzy Approach", *IAPR Conference on Machine Vision Applications*, pp. 124–127, 2005.

34. Parada-Loira, F. and J. L. Alba-Castro, "Local Contour Patterns for Fast Traffic Sign Detection", *IEEE Intelligent Vehicles Symposium*, pp. 1–6, 2010.

35. Hoferlin, B. and K. Zimmermann, "Towards Reliable Traffic Sign Recognition", *IEEE Intelligent Vehicles Symposium*, pp. 324–329, 2009.

36. De la Escalera, A., J. M. Armingol and M. Mata, "Traffic Sign Recognition and Analysis for Intelligent Vehicles", *Image and Vision Computing*, Vol. 21, No. 3, pp. 247–258, 2003.

37. Mallat, S. and Z. Zhang, "Matching Pursuits with Time-Frequency Dictionaries", *IEEE Transactions on Signal Processing*, Vol. 41, No. 12, pp. 3397–3415, 1993.

38. Kiran, C. G., L. V. Prabhu, R. V. Abdu and K. Rajeev, "Traffic Sign Detection and Pattern Recognition Using Support Vector Machine", *International Conference on Advances in Pattern Recognition*, pp. 87–90, 2009.

39. Soetedjo, A. and K. Yamada, "Traffic Sign Classification Using Ring Partitioned Method", *IEICE Transactions on Fundamentals of Electronics Communiations and Computer Sciences E Series A*, Vol. 88, No. 9, p. 2419, 2005.

40. Miura, J., T. Kanda and Y. Shirai, “An Active Vision System for Real-Time Traffic Sign Recognition”, *IEEE Transactions on Intelligent Transportation Systems*, pp. 52–57, 2000.

41. Garcia-Garrido, M., M. Sotelo and E. Martin-Gorostiza, “Fast Traffic Sign Detection and Recognition Under Changing Lighting Conditions”, *IEEE Transactions on Intelligent Transportation Systems*, pp. 811–816, 2006.

42. Ruta, A., Y. Li and X. Liu, “Real-Time Traffic Sign Recognition from Video by Class-Specific Discriminative Features”, *Pattern Recognition*, Vol. 43, No. 1, pp. 416–430, 2010.

43. Sukthankar, R., D. Pomerleau and C. Thorpe , “A Distributed Tactical Reasoning Framework for Intelligent Vehicles”, *SPIE: Intelligent Systems and Advanced Manufacturing*, October 1997.

44. Rosa, R., T. de Pedro and A. Rosetti, “Fuzzy Traffic Police for Autonomous Vehicles”, *Computer Aided Systems Theory-EUROCAST’97*, pp. 285–291, 1997.

45. Al-Shihabi, T. and R. R. Mourant, “A Framework for Modeling Human-Like Driving Behaviors for Autonomous Vehicles in Driving Simulators”, *International Conference on Autonomous Agents*, pp. 286–291, 2001.

46. Gao, M. and M. Zhou, “Control Strategy Selection for Autonomous Vehicles in a Dynamic Environment”, *IEEE International Conference on Systems, Man and Cybernetics*, Vol. 2, pp. 1651–1656, 2005.

47. Lattner, A. D., J. D. Gehrke, I. J. Timm and O. Herzog, “A Knowledge-Based Approach to Behavior Decision in Intelligent Vehicles”, *IEEE Intelligent Vehicles Symposium*, pp. 466–471, 2005.

48. Long, L. N., S. D. Hanford, O. Janrathitikarn, G. L. Sinsley and J. A. Miller, “A Review of Intelligent Systems Software for Autonomous Vehicles”, *IEEE Symposium on Computational Intelligence in Security and Defense Applications*, 2007.

49. Ferguson, D., C. Baker, M. Likhachev and J. Dolan, "A Reasoning Framework for Autonomous Urban Driving", *IEEE Intelligent Vehicles Symposium*, pp. 775–780, 2008.

50. Vacek, S., T. Gindele, J. M. Zollner and R. Dillmann, "Situation Classification for Cognitive Automobiles Using Case-Based Reasoning", *IEEE Intelligent Vehicles Symposium*, pp. 704–709, 2007.

51. Toit, N. E. D., T. Wongpiromsarn, J. W. Burdick and R. M. Murray, "Situational Reasoning for Road Driving in an Urban Environment", *International Workshop on Intelligent Vehicle Control Systems*, 2008.

52. Forbes, J., T. Huang, K. Kanazawa and S. Russell, "The Batmobile: Towards a Bayesian Automated Taxi", *International Joint Conference on Artificial Intelligence*, Vol. 14, pp. 1878–1885, 1995.

53. Ross, K., R. Chaney, G. Cybenko, D. Burroughs and A. Willsky, "Mobile Agents in Adaptive Hierarchical Bayesian Networks for Global Awareness", *IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2207–2212, 1998.

54. Kumagai, T., Y. Sakaguchi, M. Okuwa and M. Akamatsu, "Prediction of Driving Behavior Through Probabilistic Inference", *International Conference on Engineering Applications of Neural Networks*, pp. 8–10, 2003.

55. Dagli, I., M. Brost and G. Breuel, "Action Recognition and Prediction for Driver Assistance Systems Using Dynamic Belief Networks", *Agent Technologies, Infrastructures, Tools, and Applications for E-Services*, pp. 179–194, 2010.

56. Petrovskaya, A. and S. Thrun, "Model Based Vehicle Tracking for Autonomous Driving in Urban Environments", *Robotics: Science and Systems*, 2008.

57. Shen, Y., U. Ozguner, K. Redmill and J. Liu, "A Robust Video Based Traffic Light Detection Algorithm for Intelligent Vehicles", *IEEE Intelligent Vehicles Symposium*, pp. 521–526, 2009.

58. Gavrila, D. M. and V. Philomin, "Real-Time Object Detection for Smart Vehicles", *IEEE International Conference on Computer Vision*, Vol. 1, pp. 87–93, 1999.

59. ECE, UN and European Conference of Ministers of Transport. Group on Road Transport, *Revision of the Consolidated Resolution on Road Signs and Signals (RE 2): Note*, UN, 2004.

60. Kaplan, E., *Understanding GPS : Principles and Applications*, Artech House, Boston, 2nd edn., 2006.

61. E.G.M. Trafik Hizmetleri Baskanlığı, *2011 Yılı Trafik İdari Para Ceza Rehberi*, 2011, `http://www.trafik.gov.tr/mevzuat/2011_para_cezalari.xls`, June 2011.

62. Bay, H., A. Ess, T. Tuytelaars and L. V. Gool, "Speeded-up Robust Features (SURF)", *Computer Vision and Image Understanding*, Vol. 110, No. 3, pp. 346–359, 2008.

63. Rabiner, L. and B. Juang, "An Introduction to Hidden Markov Models", *IEEE ASSP Magazine*, Vol. 3, No. 1, pp. 4–16, 1986.

64. Wilamowski, B., Y. Chen and A. Malinowski, "Efficient Algorithm for Training Neural Networks with one Hidden Layer", *International Joint Conference on Neural Networks*, pp. 1725–1728, 1999.

65. Lowe, D. G., "Object Recognition from Local Scale-Invariant Features", *IEEE International Conference on Computer Vision*, p. 1150, 1999.

66. Kurtul, C., *Road Lane / Traffic Sign Detection and Tracking*, Master's Thesis, Bogazici University, 2009.

67. Cortes, C. and V. Vapnik, "Support-Vector Networks", *Machine Learning*, Vol. 20, No. 3, pp. 273–297, 1995.

68. Sato, Y. and K. Makanae, "Development and Evaluation of In-Vehicle Signing System Utilizing RFID Tags as Digital Traffic Signs", *International Journal of ITS Research*,

Vol. 4, No. 1, 2006.

69. Harmon, J., *The Design and Implementation of Geographic Information Systems*, J. Wiley, Hoboken N.J., 2003.

70. Bosch, C., *CAN Specification Version 2.0*, Tech. rep., Robert Bosch GmbH, 1991.

71. Navet, N., "Controller Area Network [Automotive Applications]", *IEEE Potentials*, Vol. 17, No. 4, pp. 12–14, 1998.

72. Wang, A. P., J. C. Chen and P. L. Hsu, "Intelligent CAN-based Automotive Collision Avoidance Warning System", *IEEE International Conference on Networking, Sensing and Control*, Vol. 1, pp. 146–151, 2004.

73. Ulmer, B., "VITA-an Autonomous Road Vehicle (ARV) for Collision Avoidance in Traffic", *IEEE Intelligent Vehicles Symposium*, pp. 36–41, 1992.

74. Merritt, D., *Building Expert Systems in Prolog*, Amzi! inc., 2001.

75. Castillo, E., *Expert Systems : Uncertainty and Learning*, Computational Mechanics; Elsevier Applied Science, Southampton; Boston; London; New York, 1991.

76. Bratko, I., *Prolog Programming for Artificial Intelligence*, Addison-Wesley, Wokingham England; Reading Mass., 1986.

77. Sterling, L., *The Art of Prolog : Advanced Programming Techniques*, MIT Press, 2nd edn., 1994.

78. Clocksin, W., *Programming in Prolog*, Springer-Verlag, Berlin; New York, 4th edn., 1994.

79. Raedt, L. D., A. Kimmig and H. Toivonen, "ProbLog: A Probabilistic Prolog and Its Application in Link Discovery", *International Joint Conference on Artificial Intelligence*, pp. 2462–2467, 2007.

80. Pearl, J., "Fusion, Propagation, and Structuring in Belief Networks", *Artificial intelligence*, Vol. 29, No. 3, pp. 241–288, 1986.

81. Kadie, C. M., D. Hovel and E. Horvitz, "MSBNx: A Component-Centric Toolkit for Modeling and Inference with Bayesian Networks", *Microsoft Research*, Vol. 28, 2001.

82. O'Grady, J., *The Garmin Nuvi Pocketguide : All the Secrets of Nuvi, Pocket Sized*, Peachpit, Berkeley CA, 2009.

83. Lajunen, T., D. Parker and H. Summala, "Does Traffic Congestion Increase Driver Aggression?", *Transportation Research Part F: Traffic Psychology and Behaviour*, Vol. 2, No. 4, pp. 225–236, 1999.

84. Shinar, D., "Aggressive Driving: The Contribution of the Drivers and the Situation", *Transportation Research Part F: Traffic Psychology and Behaviour*, Vol. 1, No. 2, pp. 137–160, 1998.

85. Florida Department of Highway Safety and Motor Vehicles, *AGGRESSIVE Driver study*, 1999, `http://www.flhsmv.gov/reports/aggr.pdf`, June 2011.

86. Klauer, S., T. Dingus, V. Neale, J. Sudweeks and D. Ramsey, "Comparing Real-World Behaviors of Drivers with High vs. Low Rates of Crashes and Near-Crashes", *Washington, DC: National Highway Traffic Safety Administration*, 2008.

87. Sergey, B., *ALGLIB Project*, 2008, `http://www.alglib.net/`, June 2011.

88. Mika, S., G. Ratsch, J. Weston, B. Scholkopf and K. Mullers, "Fisher Discriminant Analysis with Kernels", *IEEE Conference on Neural Networks for Signal Processing*, pp. 41–48, Madison, WI, USA, 1999.

89. Epic Games, *Karma (2002) Mathengine Karma User Guide*, 2002, `http://udn.epicgames.com/Two/rsrc/Two/KarmaReference/KarmaUserGuide.pdf`, June 2011.

90. Carpin, S., M. Lewis, J. Wang, S. Balakirsky and C. Scrapper, "USARSim: A Robot

Simulator for Research and Education", *IEEE International Conference on Robotics and Automation*, pp. 1400–1405, 2007.

91. Hunt, G. and D. Brubacher, "Detours: Binary Interception of Win32 Functions", *USENIX Windows NT Symposium*, Vol. 3, p. 14, 1999.

92. Thorn, A., *DirectX 9 graphics: The Definitive Guide to Direct3D*, Wordware Pub., 2005.

93. Kaplan, K. and C. Kurtul, *Automatic Driver Evaluation System*, 2009, `http://www.adesproject.com/`, June 2011.

94. Gao, X., L. Podladchikova, D. Shaposhnikov, K. Hong and N. Shevtsova, "Recognition of Traffic Signs Based on Their Colour and Shape Features Extracted Using Human Vision Models", *Journal of Visual Communication and Image Representation*, Vol. 17, No. 4, pp. 675–685, 2006.

95. Nguwi, Y. and A. Kouzani, "Automatic Road Sign Recognition Using Neural Networks", *IEEE International Joint Conference on Neural Network*, pp. 3955–3962, 2006.

96. Baro, X., S. Escalera, J. Vitria, O. Pujol and P. Radeva, "Traffic Sign Recognition Using Evolutionary Adaboost Detection and Forest-ECOC Classification", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 10, No. 1, pp. 113–126, 2009.

97. IEEE Computer Society SA Standards Board, *IEEE Recommended Practice for Software Design Descriptions*, Institute of Electrical and Electronics Engineers, New York NY, 1998.