

ADES: AUTOMATIC DRIVER EVALUATION SYSTEM

PhD Thesis Defense
by
Kemal Kaplan

June 2, 2011



INTRODUCTION

INTRODUCTION

Thrilling Facts

- More than one million accidents a year in EU.
- More than 40 000 deaths and nearly two million injuries.
- The direct and indirect estimate cost: 160 billion Euros.
- 97 percent of the accidents in our country is caused by driver mistakes.

Motivation

Current applications are usually focused on driver assistance and early warning systems or autonomous driving.

In the near future, intelligent vehicles will enforce the traffic rules.

Possible applications of ADES:

- Assist drivers in driving more safely,
- Provide information about violations committed by the drivers to the traffic centrals,
- Automate of driver license examinations,
- Keep inventory of the traffic signs on highways,
- Provide supervision for the development of the autonomous urban driving systems,
- Augmented reality applications based on traffic signs or lanes.

Problem Statement

- **Data acquisition**

Get the data as quickly, cheaply and accurate as possible.

- **Data processing**

Extract information from data.

- **Reasoning**

Design of an inference engine which can deal with the complexity and uncertainty of the acquired information.

- **Miscellaneous engineering problems.**

- Integration with other traffic control systems or the vehicle itself.
- Cost/Performance trade-offs.
- Maximum sensor utilization.

Outline

- 1 INTRODUCTION
- 2 RELATED WORK
- 3 PROPOSED APPROACH
- 4 METHODOLOGY
 - Vision Module
 - RF Module, GPS/GIS and CAN Bus
 - GIS/GPS and CAN-Bus
 - Inference Engine
 - Modeling Driver Aggressiveness
- 5 APPLICATIONS
 - The ADES Detector
 - ADES Simulation Environment
- 6 TESTS AND EVALUATION
- 7 CONCLUSION



RELATED WORK

RELATED WORK

Current ADAS Technologies

- Adaptive cruise control
- Forward collision warning
- Lane Departure Warning and Blind Spot Detection
- Speed Limit Monitoring
- Driver Drowsiness Detection

Detection and Tracking of Lane Markings

- Detection

Most common method is Hough Transform.

Neural networks, B-splines, dynamic programming approaches are also used.

- Tracking

Kalman filtering and particle filtering techniques applied.

Detection and Tracking of Traffic Signs

- Detection and tracking
Neural network, SVM, Kalman Filter, GA, Template matching, Radial symmetry transform, Wavelet features with Ada-Boost training
- Tracking
Neural networks, LDA with maximum likelihood, SVM, Feature extraction, Wavelet expansions

Classification of Traffic Signs

- Neural networks
- SVM based techniques
- Custom Methods
 - Ring Partitioned Method
 - Two camera methods
 - Matching Pursuit

Planning Techniques Used in Autonomous Vehicles

- Fuzzy if-then rules.
- Fuzzy Petri Nets.
- Knowledge-based and case-based applications.
- Voting procedures for distributed planning modules.
- Bayesian Networks.

PROPOSED APPROACH

PROPOSED APPROACH

The Big Picture

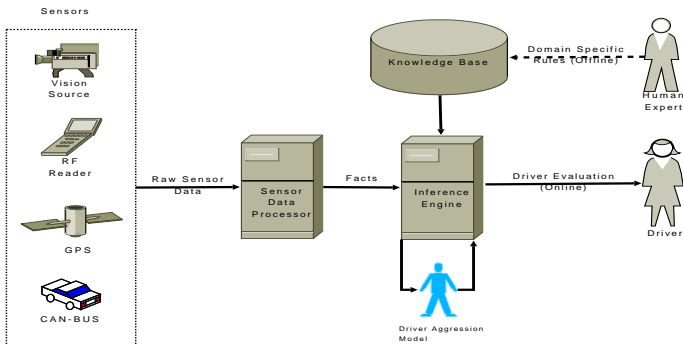


Figure: Basic system architecture of ADES project.

Data Acquisition and Processing

- Vision module
 - Detection and tracking of lane markings.
 - Detection, tracking, and recognition of traffic signs.
- RFID system
- GPS/GIS system
- CAN-Bus integration

Inference Engine

- A common interface for expert system implementations
- Prolog Based
- Belief Networks Based

Table: 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

METHODOLOGY

METHODOLOGY

Overview

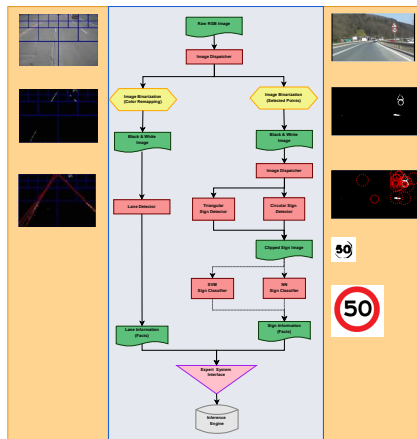


Figure: Vision processing.

Image Binarization - Color Remapping

Table: Color remapping.

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

Adaptive image binarization

$$f(r, g, b) = \left\{ \begin{array}{l} 1 \rightarrow r > \alpha \times g, r > \beta \times b \\ 0 \rightarrow o/w \end{array} \right\} \quad (1)$$
$$\alpha > 1$$
$$\beta > 1$$

Adaptive image binarization (cont'd)



Figure: Good, medium and poor conditions for traffic sign detection.

Adaptive image binarization (cont'd)

$$HIST = \text{Histogram}(HSL_{rgb}) \quad (2)$$

$$\alpha = \beta = 1 + \frac{L_{mean}}{2}$$

A better approach is;

$$p_i = \left\{ \begin{array}{l} 1 \rightarrow p = \text{white} \\ 0 \rightarrow p = \text{black} \end{array} \right\} \quad (3)$$

$$p_{white} = \sum p_i, \forall p \in \text{image}$$

$$\alpha = \left\{ \begin{array}{l} \alpha \times \alpha' \rightarrow p_{white} > \max_{white} \\ \frac{\alpha}{\alpha'} \rightarrow p_{white} < \min_{white} \\ \alpha \rightarrow o/w \end{array} \right\}, \alpha' > 1$$

$$\beta = \left\{ \begin{array}{l} \beta \times \beta' \rightarrow p_{white} > \max_{white} \\ \frac{\beta}{\beta'} \rightarrow p_{white} < \min_{white} \\ \beta \rightarrow o/w \end{array} \right\}, \beta' > 1$$

Adaptive image binarization (cont'd)

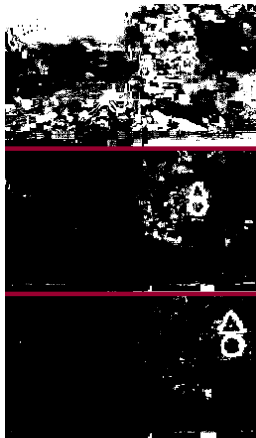


Figure: Effect of selected point count on image binarization.

Road Lane Detection

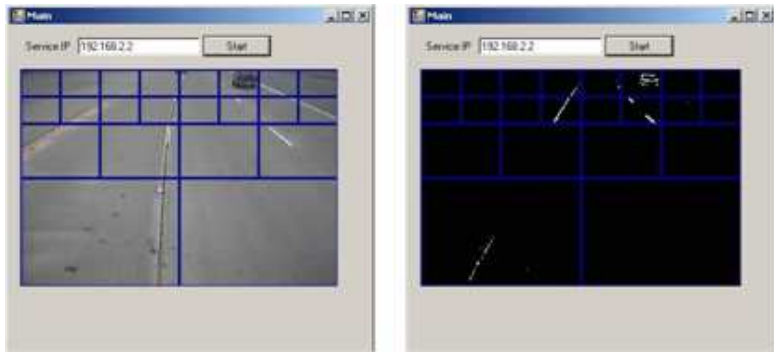


Figure: (a) Partitioned image, (b) Binary image.

Road Lane Detection (cont'd)

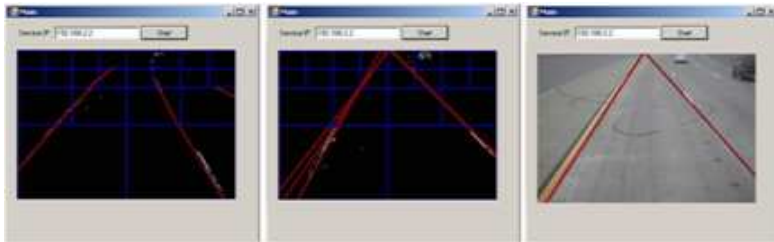


Figure: (a) Candidate lines, (b) Transformed line, (c) Detected lines.

Road Lane Detection (cont'd)

Translation function for Hough Lines;

$$\begin{aligned} r' &= r + (x - x') \cos(\theta) + (y - y') \sin(\theta) \\ \theta' &= \theta \end{aligned} \quad (4)$$

Road Lane Tracking

Table: (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

Road Lane Tracking (cont'd)

Table: (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

Traffic Sign Detection

Affine transformation

is any transformation that preserves collinearity and ratios of distances. (MathWorld)

$$\begin{vmatrix} u' \\ v' \\ w \end{vmatrix} = \begin{vmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ 1 \end{vmatrix} \quad (5)$$

$$u = u'/w$$

$$v = v'/w$$

Traffic Sign Detection (cont'd)

For example;

$$\begin{vmatrix} u' \\ v' \\ 1 \end{vmatrix} = \begin{vmatrix} 2 & 1 & 100 \\ 1 & 2 & 50 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ 1 \end{vmatrix} \quad (6)$$

Traffic Sign Detection (cont'd)

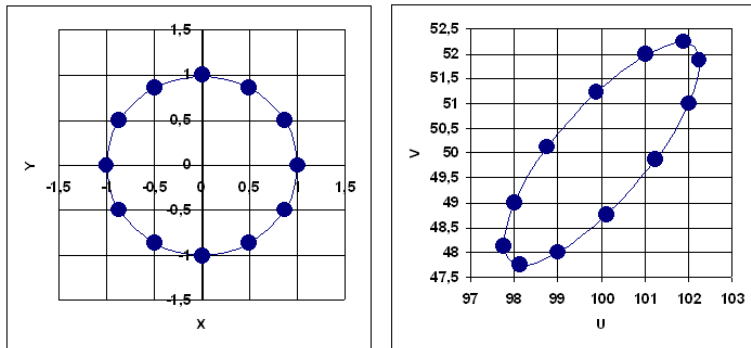


Figure: Template characteristic points in (x,y) domain, and (u,v) domain after geometric transformation.

Traffic Sign Detection (cont'd)

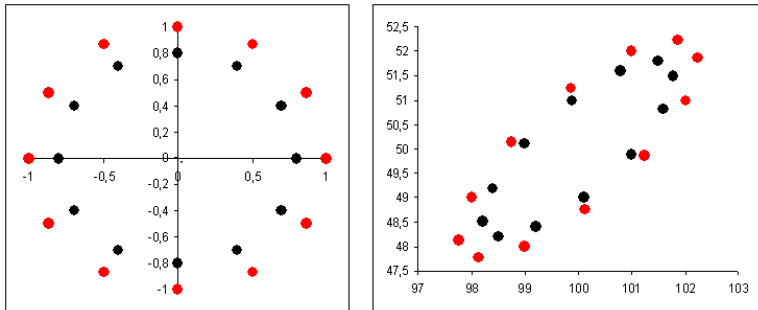


Figure: Red and non-red template points.

Traffic Sign Detection (cont'd)

In our implementation;

$$\begin{vmatrix} u \\ v \\ 1 \end{vmatrix} = \begin{vmatrix} a & 0 & c \\ 0 & a & f \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} x \\ y \\ 1 \end{vmatrix} \quad (7)$$

Traffic Sign Detection (cont'd)

Crossover function:

$$\begin{aligned}a_{newchromosome} &= \alpha \cdot a_{chromosome1} + \beta \cdot a_{chromosome2} \\c_{newchromosome} &= \alpha \cdot c_{chromosome1} + \beta \cdot c_{chromosome2} \\f_{newchromosome} &= \alpha \cdot f_{chromosome1} + \beta \cdot f_{chromosome2} \\1 &= \alpha + \beta\end{aligned} \quad (8)$$

Traffic Sign Detection (cont'd)

GA individuals;

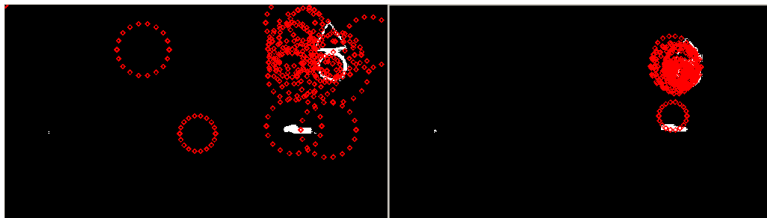


Figure: Initial and converged chromosomes.

A Modified Radial Symmetry check is performed after GA detection.

Traffic Sign Detection (cont'd)



Figure: Detected traffic sign.

Traffic Sign Classification



Figure: Traffic signs recognized by the proposed system.

Traffic Sign Classification (cont'd)

Grid based feature selection;

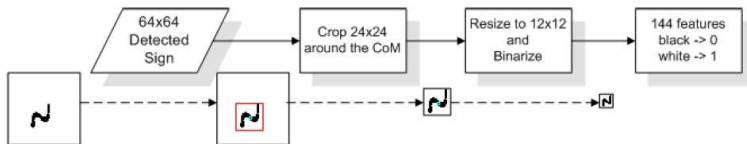


Figure: Input array formation for Grid based NN and SVM implementations.

Traffic Sign Classification (cont'd)

Surf based feature selection;

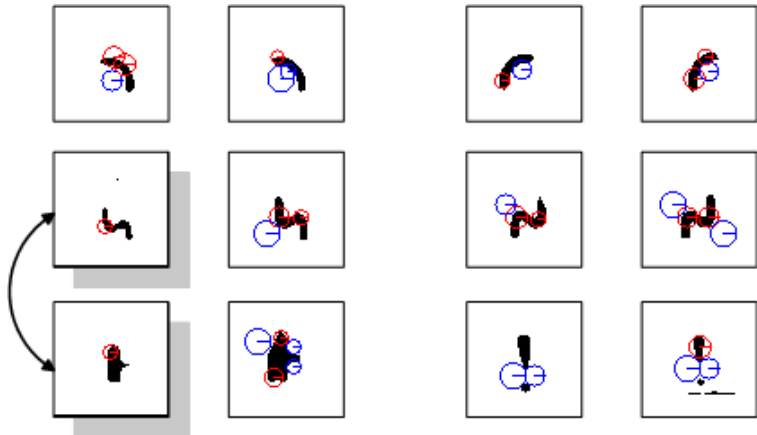


Figure: U-SURF results for different sign types (octaves=3, intervals=5).

Traffic Sign Classification (cont'd)

In the NN implementation;

- A separate network for each sign.
- Each network has one hidden layer with four nodes (144x4x1)
- NN implementation with Levenberg-Marquardt learning technique.

$$(J^t J + \lambda I) \phi = J^t E \quad (9)$$

Traffic Sign Classification (cont'd)

In the SVM implementation, the RBF kernel function is used.

$$k(u, v) = e^{-\gamma \times |u-v|^2} \quad (10)$$

Properties of Most Common RFID Systems

Table: 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

Challenges of RFID Technology

- Power consumption
- Reading range
- Antenna
- Metal interference
- Passive tag limitations

Tag ID Assignment

Table: 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 turn 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		

Simple GIS for ADES

```
<?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: Simple GIS XML for ADES.

CAN Bus

Control Area Network (CAN) is a serial communications protocol.

- Developed by Robert Bosch GmbH.
- Supports distributed real-time control with a high level of security.
- There are many ADAS applications which uses CAN bus integration.
- Provides speed of the vehicle and sensor readings from throttle, break, and steering control units.

Properties of Rule Based and Probabilistic ES

Table: 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

Comparison of Rule based and Probabilistic ES

Table: 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

ADES Interface for Expert Systems

```
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: ADES Interface for Expert Systems

Implementation with Prolog

Prolog

- is a programming language for symbolic computation,
- suitable for solving problems which contains objects and relations,
- benefits from rule-based programming, built-in pattern matching and backtracking execution,
- is not similar to a conventional programming language.

Probabilistic Prolog

Classical approach for dealing with uncertainty;

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)
```

Extra handling for probability values is required.

Probabilistic Prolog (cont'd)

Our approach;

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)
```

Probability values are coupled with facts.

Probabilistic Prolog - Path Finding Example

Consider the following graph with probability values assigned to its edges.

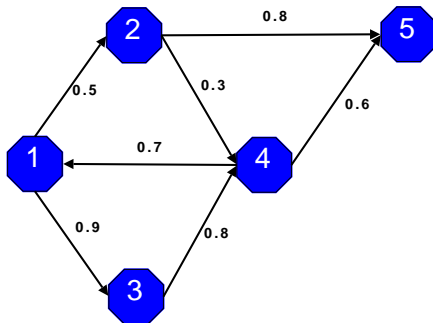


Figure: Directed graph with probability values.

Probabilistic Prolog - Path Finding Example (cont'd)

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)".
```



Probabilistic Prolog - Path Finding Example (cont'd)

Query

```
?- path(1,5,L).
```

```
Prob for this answer:0,432
```

```
L = [1,3,4,5] (0,000 sec) more? (y/n);
```

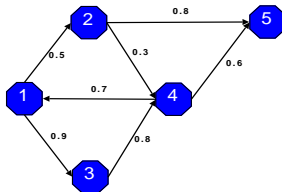
```
Prob for this answer:0,09
```

```
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
```



Time-Decaying Facts

Why?

The probability of a fact may decrease by time and may not be available after a time period.

How?

Any monotone decreasing function can be used for diminishing the probability of a fact. Sigmoid functions are used in our implementation.

Time-Decaying Facts - Sigmoid Functions

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

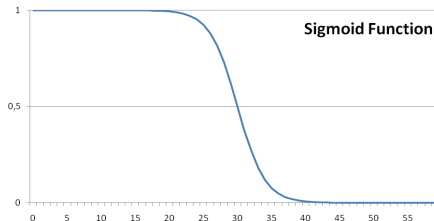
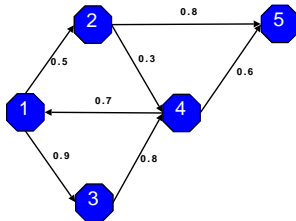


Figure: Sigmoid function used for time-decaying facts.

Time-Decaying Facts - Example



Rules and Facts

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

Time-Decaying Facts - Example (cont'd)

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








Implementation with Belief Networks

In the ADES project,

- The Microsoft Bayesian Network Editor (MSBNx) is used for implementing the BN expert system.
- A belief network is created for each regulation by using MSBNx's editor.
- These networks are queried at the assertion or retraction of the facts.

Regulation Implementations

Table: Targeted traffic violations.

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

Regulation Implementations (cont'd)

Table: 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.

Violation of Directional Regulations

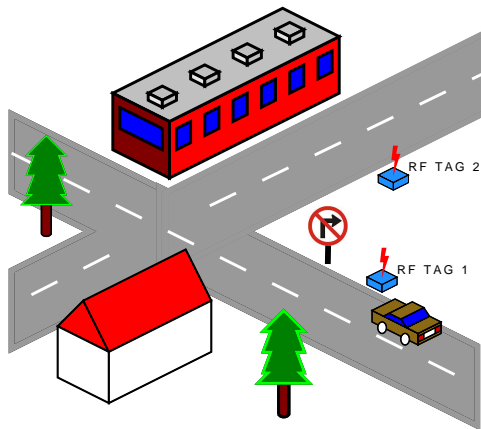


Figure: Sample scenario for detecting directional regulation violations.

Violation of Directional Regulations (cont'd)

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"
```



Violation of Directional Regulations (cont'd)

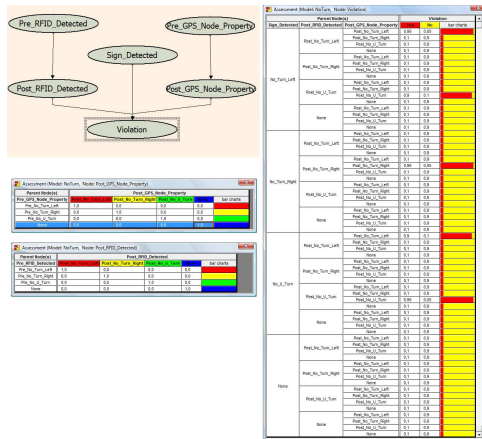


Figure: The BN for directional regulations.

Violation of Speed Limitation



Figure: Speed limit aware GPS application.

Violation of Speed Limitation (cont'd)

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'
```



Violation of Speed Limitation (cont'd)

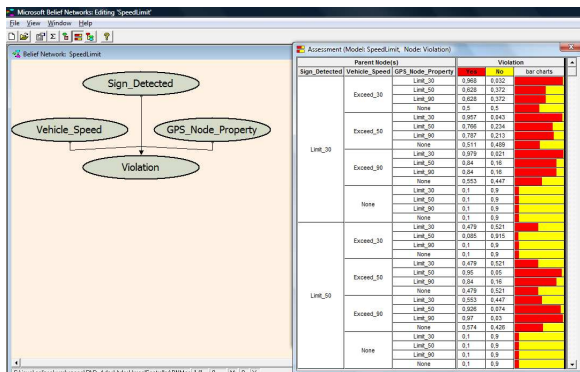


Figure: The belief network and it's property assessments for speed limitations.

Illegal Overtaking



Figure: Overtake forbidden sign.

Illegal Overtaking (cont'd)

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).
```

Illegal Overtaking (cont'd)

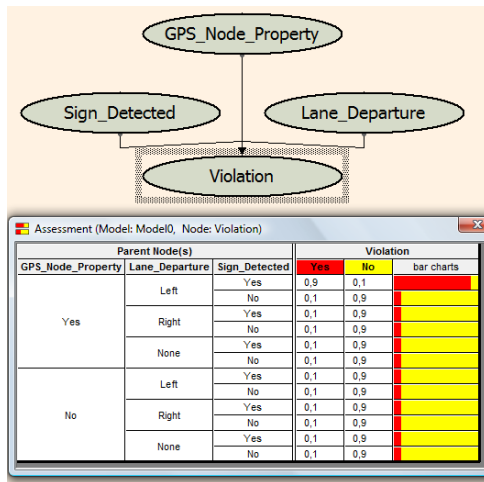


Figure: The belief network and it's property assessments for illegal overtaking.

Violation of Red Traffic Lights

Active RF tags are required!

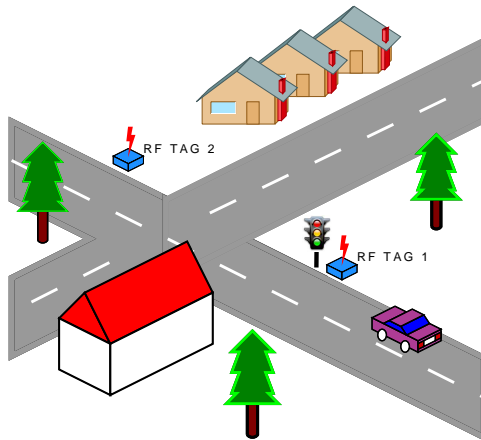


Figure: Sample scenario for red light violations.

Violation of Red Traffic Lights (cont'd)

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).
```

Violation of Red Traffic Lights (cont'd)

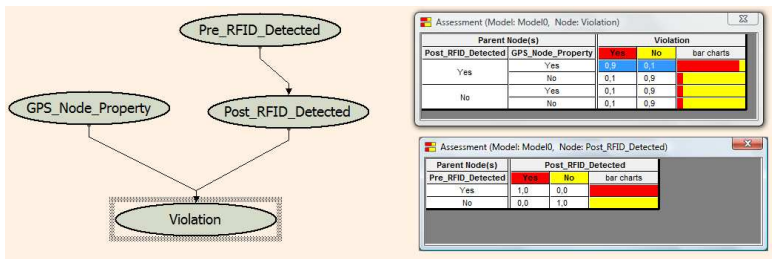


Figure: The belief network and it's property assessments for traffic light regulations.

Modeling Driver Aggressiveness

$$Violation = \begin{cases} \text{Yes} \rightarrow p > t \times (1 - \alpha \times c/c_{max}), 0 < \alpha < 1 \\ \text{No} \rightarrow o/w \end{cases} \quad (11)$$

APPLICATIONS

The ADES Detector

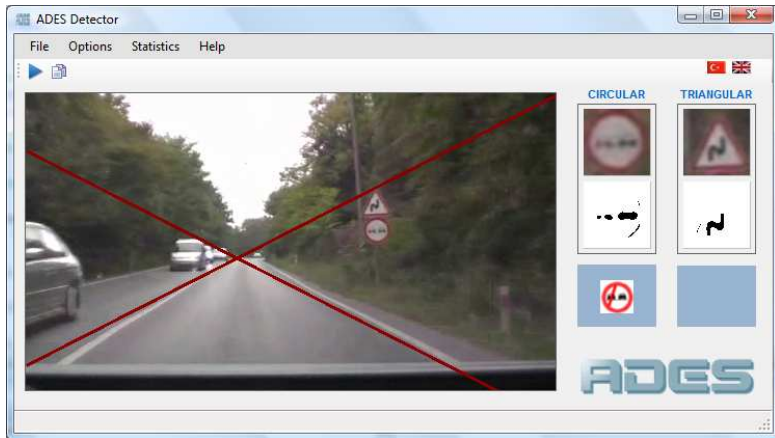


Figure: The ADES Detector.

ADES Simulation Environment

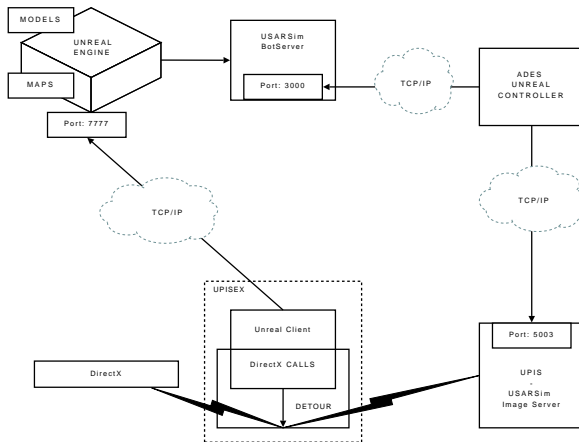


Figure: Interaction between Unreal Engine, USARSim and Unreal ADES controller.

Unreal Engine

- A commercial game engine from Epic Games.
- First version UE1 released on 1998. ADES on UE2.5.
- Powerful visual rendering and physical modeling.
- Provides object oriented programming called the UnrealScript.

USARSim

USARSim is an Unreal Engine expansion for urban search and rescue (USAR) robots and environments.

- Open source research tool.
- Used in many domains like DARPA and NIST applications.
- RoboCup Rescue Virtual Robots Championship based on USARSim.
- Provides numerous models for the use of researchers.

USARSim - Sedan Class



Figure: The Sedan in the ADES Unreal world.

USARSim - Sedan Class (cont'd)

Table: 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

USARSim - Sedan Class (cont'd)

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', ...
```



UnrealEd - The Unreal Level Editor

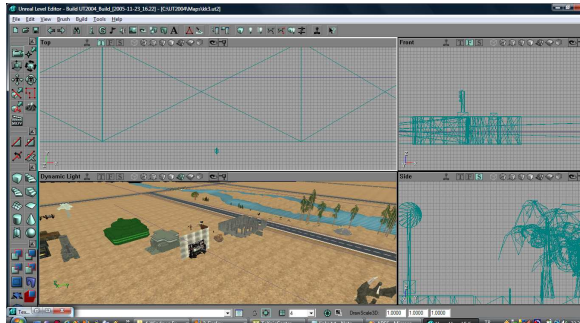


Figure: The ADES Unreal world in UnrealEd.

UnrealEd - 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).

UnrealEd - Static Meshes

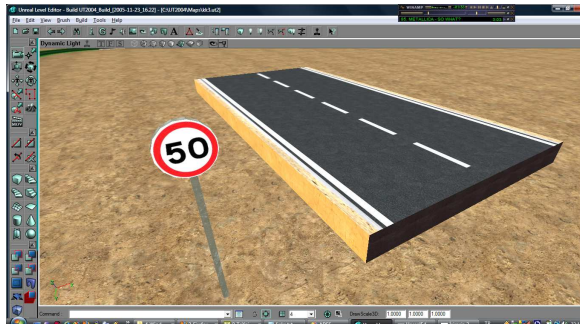


Figure: Road segment and speed limit sign static meshes.

UnrealEd - Terrain/Zone Info

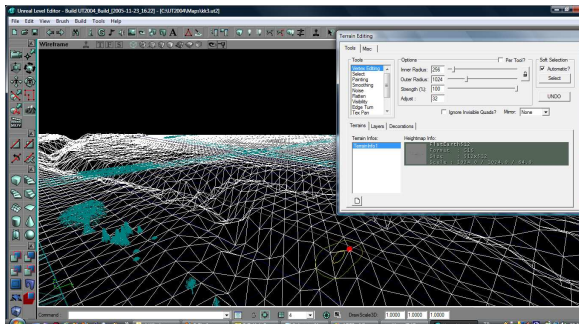


Figure: The wireframe of the terrain of the ADES Unreal world.

ZoneInfo class is used for misc. effects like zone lighting, ambient sound effects, fog effects.

UnrealEd - Sky Box

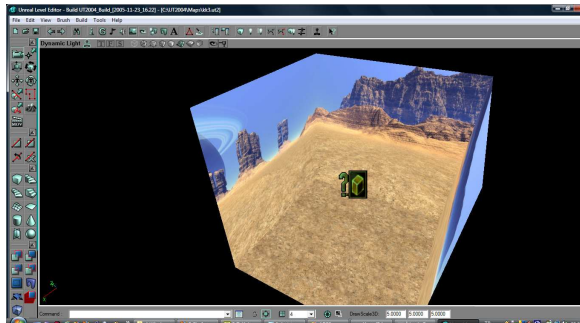


Figure: The SkyBox of the ADES Unreal world.

UPISEX

UPISEX is a wrapper project for USARSim's image server UPIIS for making it visible for .NET projects.

- Creates the Unreal Client with Detours hook.
- UPIIS captures DirectX specific calls to reach client's memory blocks.
- UPIIS opens a TCP/IP port for serving the client's video stream.

ADES Unreal Controller



Figure: The Unreal ADES controller application interface.

TESTS AND EVALUATION

TESTS AND EVALUATION

Evaluation of Lane Detection



Figure: Differences between classical Hough transformation and proposed approach.

Evaluation of Sign Detection

Table: 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

Evaluation of Sign Detection (cont'd)

Table: 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

Evaluation of Sign Classification

Table: Classification success rate of circular signs.

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

Table: Classification success rate of triangular signs.

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

Evaluation of Sign Recognition System

Table: 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

Evaluation of Sign Recognition System (cont'd)

Table: Comparison with selected methods from the last decade.

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

Output of Prolog Based ES

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'
```

Figure: Speed violation output of Prolog based expert system.

Output of Prolog Based ES (cont'd)

$$p_v = (p_d \times \alpha_d)(p_s \times \alpha_s)(p_g \times \alpha_g) \quad (12)$$

Note: The probabilities of the facts are decreasing by the time!

Output of BN Based ES

BN based expert system

```
SpeedLimit:Vehicle_Speed()->(0,1,0,0)
SpeedLimit belief: 0.5436
SpeedLimit:GPS_Node_Property()->(0.0333,0.0333,0.0333,0.9)
SpeedLimit:Sign_Detected()->(1,0,0,0)
SpeedLimit:Vehicle_Speed()->(0,1,0,0)
SpeedLimit belief: 0.5436
SpeedLimit:GPS_Node_Property()->(0.9708,0.0097,0.0097,0.0097)
SpeedLimit:Vehicle_Speed()->(0,1,0,0)
SpeedLimit belief: 0.9491
```

Figure: Speed violation output of BN based expert system.

Output of BN Based ES (cont'd)

$$p_v = \sum_{d \in \text{Sign_Detected}, s \in \text{Vehicle_Speed}, g \in \text{GPS}_{Node_Property}} p_d \times p_s \times p_g \times vp(d, s, g) \quad (13)$$

For the previous output;

$$\begin{aligned}
 p_v &= 0.9708 \times 1 \times 1 \times 0.9570 & (14) \\
 &+ 0.00973 \times 1 \times 1 \times 0.7660 \\
 &+ 0.00973 \times 1 \times 1 \times 0.7870 \\
 &+ 0.00973 \times 1 \times 1 \times 0.5110 \\
 p_v &= 0,94913832
 \end{aligned}$$

CONCLUSION

CONCLUSION

Key Contributions

- 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,
- Application of BN to driver evaluation system,
- The applications for real and simulated environments.

Future Works

Not limited to;

- End-to-end implementation of the application for the real environment,
- Self modifying, or learning, inference engine,
- Improved sensor processor modules for given modules,
- Integration of advance vision processing modules,
- Wireless integration of the system to the traffic control center,
- More vehicle to vehicle and vehicle to infrastructure communication,
- Integration with existing intelligent highway systems.

Acknowledgements

I would like to thank to my thesis supervisor Professor Levent Akın, my committee, Assoc. Prof. Tankut Acarman, Prof. Gökmen Ergün, Prof. Cem Say, and Prof. Oğuz Tosun, the past and present members of the Robotics Group of Boğaziçi University Artificial Intelligence Laboratory, and finally my family for their never ending support.

QUESTIONS?

Kemal Kaplan
kaplanke@boun.edu.tr