

# Expert System Design for an Autonomous Driver Evaluation System

Kemal Kaplan and H. Levent Akin, *Senior Member, IEEE*

**Abstract**—In this paper, two expert system implementations for the Automatic Driver Evaluation System (ADES) Project are introduced. These expert systems are used for deciding whether a driver violates the traffic rules or not, according to the facts derived from the data acquired by the sensors. Sample scenarios are executed in a highly realistic simulation environment. The advantages and shortcomings of both approaches are discussed.

## I. INTRODUCTION

Traffic accidents are one of the main causes of death and economic loss in most of the developed countries. According to the Road Safety Action Program of European Commission, more than one million accidents a year cause more than 40 000 deaths and nearly two million injuries on the roads. On the other side, the direct and indirect cost has been estimated at 160 billion Euros [1]. Nearly all of these accidents are caused by driver mistakes.

The ADES project is a framework for developing real time on-vehicle applications for detecting the traffic violations committed by the drivers. The project has three main modules. The first module acquires raw data using different sensors like camera, RF reader or GPS. The second module processes the data acquired by these sensors. The final module, which is the inference engine, tries to detect the violations by using this information. The inference engine should contain a reasoning mechanism, which is usually an expert system or a mixture of expert systems, for deciding and justifying the violations. Any expert system proposed for ADES project should be able to use the information gathered from the environment. This information should be processed considering the traffic rules. The proposed expert system should also handle the uncertainty due to the inaccuracy of the sensors or the misleading data processing results. The overall structure of the ADES project is shown in Figure 1.

Although autonomous driver evaluation is a new research area, expert systems are widely used for similar purposes such as autonomous driving. Sukthankar *et al* [2] presented a distributed solution, which consists of a collection of reasoning objects that vote upon a set of possible actions for dealing with the complexity of tactical reasoning. Rosa *et al* [3] implemented a fuzzy expert system to organize traffic regulations in a town area. Al-Shihabi *et al* [4] proposed a framework for modeling driver behavior with perception, emotions, decision-making, and the decision-implementation units formed of fuzzy variables and if-then

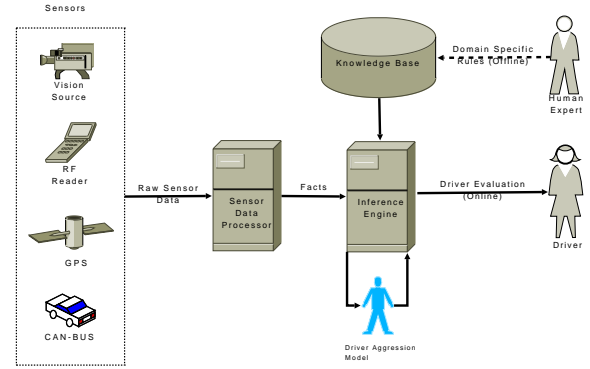


Fig. 1. The overall structure of the ADES project.

rules. Gao and Zhou [5] 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* [6] 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* [7] presented a review of software platforms for autonomous vehicles and their artificial intelligence development capabilities. Ferguson *et al* [8] presented a reasoning framework, which consists of the mission, behavioral, and motion planning components performing reasoning for an autonomous vehicle navigating through urban environments. Vacek *et al* [9] used case-based reasoning in order to predict the progress of the current situation and to select the appropriate behavior. DuToit *et al* [10] 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.

In this study we consider two different expert systems namely a probabilistic *Prolog* based and Bayesian Network based, to manage the challenging task of real-time driver evaluation. The proposed systems can accept the predefined rules from the human experts, use the information provided by the outside world for reasoning, and handle the uncertainty by considering the probabilities of the accuracy of this information.

The organization of the rest of the paper is as follows. *Prolog* and Bayesian Network based expert systems are discussed in Section II. In Section III the proposed approach is explained. Experimental evaluation of the proposed expert systems in a highly realistic simulation environment is given in Section IV. Discussion of the results and the conclusions are given in Section V.

Manuscript received April 25, 2011.

Kemal Kaplan, and H. Levent Akin are with the Department of Computer Engineering, Boğaziçi University, 34342, Bebek, İstanbul, Turkey. (e-mail : kaplanke@boun.edu.tr; akin@boun.edu.tr)

## II. KNOWLEDGE REPRESENTATION APPROACHES

Expert systems may use different knowledge representation approaches. Here we consider two of them, namely *Prolog* based and Bayesian Networks based.

### A. Prolog Programming Language

*Prolog* is a logic programming language. It is suitable for solving problems which contains objects and relations between these objects. The relations defined over the variables are called the *rules*. On the other hand, the *facts* are the means of stating that relations that exist between the objects. Therefore, the relations of the facts only contain atoms []. 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 [11].

There are a few applications of *Prolog* in autonomous vehicle researches. Brunet *et al* used a *Prolog* based programming language for developing a multi-agent architecture for an autonomous vehicle driver model [12]. Piaggio *et al* used a *Prolog* based application to implement the symbolic component of their real time autonomous motion planning approach, which is responsible for high level planning [13].

### B. Belief Networks

A belief network (BN) is an acyclic graph used for modeling the uncertainty in a domain [14]. The nodes of the graph are random values and the arcs are conditional dependencies between these variables. BNs are reasoning tools based on probability. Although it is possible to design multiply-connected (or cyclic) BNs, probabilistic inference using this type of networks is NP-hard [15].









BNs have been studied in autonomous driving or driver assistant systems for more than a decade. The idea behind the time dependent dynamic Bayesian networks is quite similar with Markovian Models and Kalman Filters. In 1995, Forbes *et al* [16] 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 considered in 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* [17] described a distributed information fusion system based on hierarchical BNs using D'Agent mobile agent system. In the proposed system the structure of the BN can be rearranged according to the gathered information. Kumagai *et al* [18] also used dynamic BN to predict the driver's stopping 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* [19] also used a similar technique for predicting the lane departure behaviors of the drivers. In 2008, Petrovskaya *et al* [20] used a dynamic BN 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.

## III. PROPOSED APPROACH

The targeted traffic rules considered in this study, the sample traffic signs related with these regulations, and the possible inputs to the system for assertion, retraction, and evaluation of these rules are given in Table I. Different combinations of these inputs can be used for different violations.

TABLE I  
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
Emergency Lane		

Here for the sake of simplicity we present only two kind of traffic violations which are directional restrictions and speed limits. In addition, the inputs are the detected traffic signs, RFID readings and speed of the vehicle. Other inputs like GPS information, road lanes, or steering angle are not considered in the following sections. However, the design of the proposed expert systems allows the human expert to insert new rules without changing the implementation of the expert system.

### A. ADES Interface for Expert Systems

In order to allow implementing generic expert systems we defined an interface for integrating various expert system implementations into the ADES project as shown in Fig. 2.

The *init* function initializes the expert system with various parameters. 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

```

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();
}

```

Fig. 2. ADES Interface for Expert Systems

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.

### B. Probabilistic Prolog Based Expert System

In the ADES project and many other real life problems, the relations between the objects are not always very clear. Therefore, the expert system should also consider the uncertainties associated with the supplied data. Therefore we used a modified version of *Prolog* language which allows assignment of probabilities to the facts. There are also other *Prolog* implementations available where the probability processing extensions can be consulted as libraries without changing the engine itself [21]. We also introduce another improvement which is called *time decaying facts*. We assign time limits for the facts with a second parameter whenever necessary. This is an important feature since the duration of some local regulations (like directional restrictions) can be determined according to the speed of the vehicle.

The *Prolog* rule and sample facts for speed limit violation detection are as given in Fig. 3.

```

%rule declaration
violation(V,A,P) :- 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).

%sample facts asserted by sensor systems
velocity_exceeds(50)
"0.9::sign_detected(speed_limit,50)"

```

Fig. 3. Rules and Facts for Speed Limit

The speed limit violation is decided according to the speed of the vehicle and the detection of the traffic sign for a particular speed limit. The violation predicate returns the final probability of the traffic violation if it can unify the *V* parameter. The first sample fact in Fig. 3 states that the vehicle exceeds 50 km/h with a probability of 1. The second sample fact is asserted for a speed limit sign for 50 km/h is detected with a probability of 0.9.

The *Prolog* rule and sample facts for directional restrictions are as given in Fig. 4.

```

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

%sample facts asserted by sensor systems
"0.9::sign_detected(no_turn_left)"
"0.9::sign_detected(no_turn_right)"
"0.9::sign_detected(no_turn)"
"0.9::30::rfid_detected(pre,no_turn_left)"
"0.9::30::rfid_detected(pre,no_turn_right)"
"0.9::30::rfid_detected(pre,no_turn)"
"0.9::30::rfid_detected(post,no_turn_left)"
"0.9::30::rfid_detected(post,no_turn_right)"
"0.9::30::rfid_detected(post,no_turn)"

```

Fig. 4. Rules and Facts for Directional Restrictions

This violation occurs when the vehicle detects a no turning sign for left, right or U turn. In addition to the sign, the RFID tag which is assigned to that sign should be sensed. In addition to the speed limits signs, an additional confirmation RFID tag should also be sensed to complete the unification of the violation rule. The *V* parameter gives the name of the violation and the *P* parameter gives the probability of the violation. The sample rules are similar to the previous ones where 0.9 presents the accuracy of the information. The sample facts about the RFID detections benefit from both improvements of our *Prolog* implementation. The probability of this reading is 0.9 and this fact will be removed from the system after 30 seconds.

### C. Belief Network Based Expert System

Two belief networks designed for speed limitations and directional restrictions as shown in Fig. 5. These graphs are introduced to the system by an human expert. The sensor values are mapped to the conditional variables, or nodes, in these graphs. The states of these variables are arranged according to possible inputs from the sensors.

The *violation* node in both networks have the same state values which are *yes* and *no*. However, the *sign\_detected* node has different states in each network. In the speed limitation network, the states of the *sign\_detected* node are 30, 50, and 90 km/h signs. The *velocity* node also has similar state values for representing the vehicle speed. On the other hand, in the directional limitations network, the *sign\_detected* node has *no\_turn\_left*, *no\_turn\_right*, *no\_u\_turn*, and *none* states. The states for the RFID related nodes are the same for this network.

The probability distributions of the conditionally independent nodes are assigned by the *assertFact* function described in the expert system interface. For example, if a control RFID for no left turning restriction is sensed with a probability of 0.9, the probability value of the *no\_turn\_left* state becomes 0.9 for the *post\_RFID\_detected* node. The remaining 0.1 is shared equally by the other three states (*no\_turn\_right*, *no\_u\_turn*, and *none*).

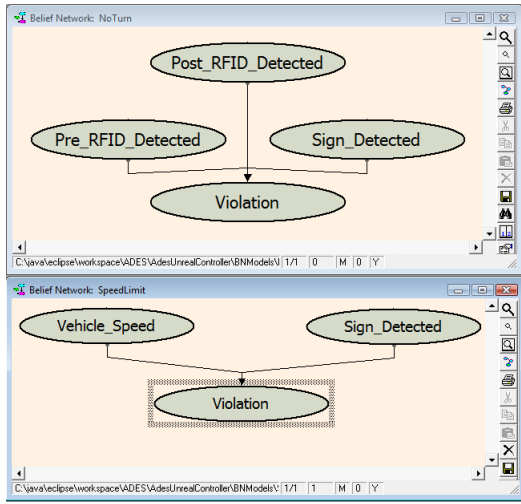


Fig. 5. Belief Networks for Directional Restrictions and Speed Limitations.

Different from other nodes, the probabilities of the violation node is entered to the system by the human expert during the network design as shown in Figure 6.

Assessment (Model: SpeedLimit, Node: Violation)				
Parent Node(s)		Violation		
Sign_Detected	Vehicle_Speed	Yes	No	bar charts
Limit_30	Exceed_30	0,95	0,05	
	Exceed_50	0,97	0,03	
	Exceed_90	0,99	0,01	
	None	0,1	0,9	
Limit_50	Exceed_30	0,1	0,9	
	Exceed_50	0,95	0,05	
	Exceed_90	0,97	0,03	
	None	0,1	0,9	
Limit_90	Exceed_30	0,1	0,9	
	Exceed_50	0,2	0,8	
	Exceed_90	0,95	0,05	
	None	0,1	0,9	
None	Exceed_30	0,1	0,9	
	Exceed_50	0,1	0,9	
	Exceed_90	0,2	0,8	
	None	0,1	0,9	

Fig. 6. Belief Networks for and Speed Limitations.

The probability distribution of the violation node of the belief network for the directional limitations is another table with 64 rows. The structure of the values are similar to the speed violation network, therefore this table is not presented here.

#### IV. EXPERIMENTAL RESULTS

Although the ADES project targets the real physical environment, the development process can be accelerated by the use of a simulation environment. Our choice for the simulation engine is the *Unreal Engine* which provides both the physics engine and the renderer [22]. The capabilities of this engine makes it a favorite tool for both robotics researchers and game developers. In addition to the core simulation engine, we also used *USARSim* which is an *Unreal Engine* expansion for urban search and rescue (USAR) robots and environments. *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 *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 in the *USARSim* manual [23]. We used the *Sedan* class provided as an *USARSim* model as our vehicle which is shown in Figure 7.



Fig. 7. The Sedan in the ADES Unreal world.

The simulation environment requires a map where the vehicle can cruise on. This map is prepared using the level editor of the *Unreal Engine*. The *ADES Unreal Controller* application acts as an *Unreal* client which connects to the *Unreal Engine*. The engine runs a *USARDeathMatch* game on the prepared map. The vehicle can be controlled by using the *ADES Unreal Controller* application as shown in Fig. 8. Moreover, the inputs and the outputs of the expert systems can also be followed by using this application.

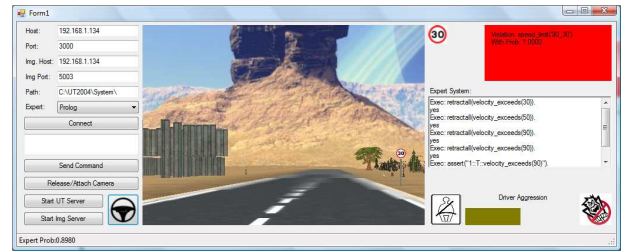


Fig. 8. The ADES Unreal Controller.

The sample run contains a violation condition for the 30 km/h sign and a violation for a no turning left condition.

For the *Prolog* inference engine, the underlying system is developed based on the *Prolog* engine *CSharp Prolog* [24]. The inference outputs of the *Prolog* expert system for these sample runs are as given in Fig. 9 and Fig. 10.

In both cases the expert system decides that the driver makes the related violation according to the asserted facts. The lifetime of the facts are replaced by the default value denoted as  $T$ . This means that these RFID readings will be available as facts for the next 45 seconds.

For the belief network implementation, the *Microsoft Bayesian Network Editor* (MSBNx), which is a *Microsoft*

```

Exec: retractall(sign_detected(speed_limit,30)).
yes
Exec: assert("0.9963::sign_detected(speed_limit,30)").
yes
Exec: retractall(velocity_exceeds(30)).
yes
Exec: retractall(velocity_exceeds(50)).
yes
Exec: retractall(velocity_exceeds(90)).
yes
Exec: retractall(velocity_exceeds(50)).
yes
Exec: assert("1::velocity_exceeds(50)").
yes
Exec: violation(V,A,P).
Prob for this answer:0.9963

P = 0.9963
V = speed_limit
A = '50_30'

```

Fig. 9. *Prolog* Inference Output for Speed Limitation

```

Exec: retractall(sign_detected(no_turn_left)).
yes
Exec: assert("0.9999::sign_detected(no_turn_left)").
yes
Exec: retractall(rfid_detected(pre,no_turn_left)).
yes
Exec: assert("1::T::rfid_detected(pre,no_turn_left)").
yes
Exec: retractall(rfid_detected(post,no_turn_left)).
yes
Exec: assert("1::T::rfid_detected(post,no_turn_left)").
yes
Exec: violation(V,A,P).
Prob for this answer:0.9999

P = 0.999830009331709
V = no_turn_left
A = "N/A"

```

Fig. 10. *Prolog* Inference Output for Directional Limitation

Windows library for creating and evaluating Bayesian Networks [25], is used. 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 inference results of the belief networks for the same samples runs are as given in Fig. 11 and Fig. 12.

```

SpeedLimit:Sign_Detected()->(0.9963,0.0012,0.0012,0.0012)
SpeedLimit:Vehicle_Speed()->(0,0,1,0)
SpeedLimit violation belief 0.9890

```

Fig. 11. Belief Network Inference Output for Speed Limitation

The values in the parentheses are the probabilities assigned for each state for the node with the displayed name. For example, in the directional restriction inference case in

```

NoTurn:Sign_Detected()->(0.9999,3e-005,3e-005,3e-005)
NoTurn:Pre_RFID_Detected()->(1,0,0,0)
NoTurn:Post_RFID_Detected()->(1,0,0,0)
NoTurn violation belief 0.8999

```

Fig. 12. Belief Network Output for Directional Limitation

Fig. 12, the probability of the *no\_left\_turn* state for the *Sign\_Detected* node is 0.9963. Similarly the pre and post RFID detection nodes have a perfect sensing value for their *no\_left\_turn* states. However, different from the previous method, the probability of the violation is not simply the multiplication of the probabilities of the selected state probabilities. The probability distribution of the violation node should also be considered. For example, in the speed limit inference case, the *Sign\_Detected* node has the highest probability value for 30 km/h sign which is 0.9963. In addition, the *Vehicle\_Speed* node states that the velocity of the vehicle exceeds 90 km/h with the probability of 1. The probability of the violation is calculated from the following equation,

$$p_v = \sum_{d \in \text{Sign\_Detected}, s \in \text{Vehicle\_Speed}} p_d \times p_s \times vp(d, s) \quad (1)$$

where  $d$  and  $s$  are the state probabilities for *Sign\_Detected* and *Vehicle\_Speed* nodes.  $vp$  denotes the probability values for violation node which can be found from Figure 6. For this particular case  $p_v$  is calculated as:

$$\begin{aligned}
p_v &= 0.9963 \times 1 \times 0.99 \\
&\quad + 0.0012 \times 1 \times 0.97 \\
&\quad + 0.0012 \times 1 \times 0.95 \\
&\quad + 0.0012 \times 1 \times 0.2 \\
p_v &= 0.9890
\end{aligned}$$

Note that the arguments which are zero due to the  $vs$  multiplicand are not shown in the equation above.

## V. DISCUSSION AND CONCLUSIONS

In this study we propose two expert system designs for evaluating the driver violations in the ADES project. The implemented expert systems are based on *Prolog* and Belief Networks. The properties of these methods are compared on Table II and III.

TABLE II  
PROPERTIES OF EXPERT SYSTEM MODELS.

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



TABLE III  
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, Detection of unnecessary rules	Performance issues, Certainty implementation

Both models have advantages and disadvantages. 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, relatively high number of parameters even in a small model makes the model more complicated both for understanding and management. On the other hand, the rule based systems are easier to maintain. However, this advantage usually causes performance issues for complex problems where extensive rule chaining is required.

Another issue is the retraction of the facts from the system. The retraction process is as important as the assertion of the facts since invalidated facts can cause misleading decisions. Although the *Prolog* based expert system provides a lifetime limit for the facts, additional controls should be considered for asserted facts.

The threshold used by the expert system is defined by the human expert. However, our implementation slightly changes this threshold according to the previous actions of the driver. In other words, the expert system favors a driver who drives safely by increasing the threshold value for the violation probability. Similarly, the threshold is decreased to a predefined level for faulty drivers.

A simulated environment is generated in this study to present the problem and introduce two possible expert system implementations for the solution. Other expert systems with more complex rules robust to real life data can be developed as a future work.

## REFERENCES

- [1] "European commission: Transport policy white paper." [http://europa.eu/legislation\\_summaries/internal\\_market/single\\_market\\_for\\_goods/motor\\_vehicles/technical\\_implications\\_road\\_safety/l24257en.htm](http://europa.eu/legislation_summaries/internal_market/single_market_for_goods/motor_vehicles/technical_implications_road_safety/l24257en.htm), Sept 2001.
- [2] R. Sukthankar, D. Pomerleau, and C. Thorpe, "A distributed tactical reasoning framework for intelligent vehicles," in *SPIE: Intelligent Systems and Advanced Manufacturing*, October 1997.
- [3] R. G. Rosa, T. d. Pedro, and A. Rosetti, "Fuzzy traffic police for autonomous vehicles," in *EUROCAST '97: Proceedings of the A Selection of Papers from the 6th International Workshop on Computer Aided Systems Theory*, (London, UK), pp. 285–291, Springer-Verlag, 1997.
- [4] T. Al-Shihabi and R. R. Mourant, "A framework for modeling human-like driving behaviors for autonomous vehicles in driving simulators," in *AGENTS '01: Proceedings of the fifth international conference on Autonomous agents*, (New York, NY, USA), pp. 286–291, ACM, 2001.
- [5] M. Gao and M. Zhou, "Control strategy selection for autonomous vehicles in a dynamic environment," *Systems, Man and Cybernetics*, 2005 *IEEE International Conference on*, vol. 2, pp. 1651–1656 Vol. 2, Oct. 2005.
- [6] A. D. Lattner, J. D. Gehrke, I. J. Timm, and O. Herzog, "A knowledge-based approach to behavior decision in intelligent vehicles," in *IEEE Intelligent Vehicles Symposium, Las Vegas*, pp. 466–471, 2005.
- [7] L. Long, S. Hanford, O. Janrathitkarn, G. Sinsley, and J. Miller, "A review of intelligent systems software for autonomous vehicles," in *IEEE Symposium Series in Computational Intelligence*, (Honolulu, Hawaii), April 1-5 2007.
- [8] D. Ferguson, C. Baker, M. Likhachev, and J. Dolan, "A reasoning framework for autonomous urban driving," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV 2008)*, (Eindhoven, Netherlands), pp. 775–780, June 2008.
- [9] S. Vacek, T. Gindele, J. Zollner, and R. Dillmann, "Using case-based reasoning for autonomous vehicle guidance," *Intelligent Robots and Systems*, 2007. *IROS 2007. IEEE/RSJ International Conference on*, pp. 4271–4276, 29 2007-Nov. 2 2007.
- [10] N. DuToit, T. Wongpiromsarn, J. Burdick, and R. Murray, "Situational reasoning for road driving in an urban environment," in *International Workshop on Intelligent Vehicle Control Systems (IVCS)*, 2008.
- [11] D. Merritt, *Building expert systems in Prolog*. Springer-Verlag, 1989.
- [12] C.-A. Brunet, R. Gonzalez-Rubio, and M. Tetreault, "A multi-agent architecture for a driver model for autonomous road vehicles," in *Electrical and Computer Engineering*, 1995. *Canadian Conference on*, vol. 2, pp. 772–775 vol.2, Sept. 1995.
- [13] M. Piaggio and A. Sgorbissa, "Real-time motion planning in autonomous vehicles: A hybrid approach," in *AI\*IA 99: Advances in Artificial Intelligence* (E. Lamma and P. Mello, eds.), vol. 1792 of *Lecture Notes in Computer Science*, pp. 368–378, Springer Berlin / Heidelberg, 2000.
- [14] J. Pearl, "Fusion, propagation, and structuring in belief networks\* 1," *Artificial intelligence*, vol. 29, no. 3, pp. 241–288, 1986.
- [15] G. Cooper, "The computational complexity of probabilistic inference using Bayesian belief networks," *Artificial intelligence*, vol. 42, no. 2-3, pp. 393–405, 1990.
- [16] J. Forbes, T. Huang, K. Kanazawa, and S. Russell, "The batmobile: Towards a bayesian automated taxi," in *International Joint Conference on Artificial Intelligence*, vol. 14, pp. 1878–1885, Citeseer, 1995.
- [17] K. Ross, R. Chaney, G. Cybenko, D. Burroughs, and A. Willsky, "Mobile agents in adaptive hierarchical bayesian networks for global awareness," in *Systems, Man, and Cybernetics*, 1998. *1998 IEEE International Conference on*, vol. 3, pp. 2207–2212, IEEE, 2002.
- [18] T. Kumagai, Y. Sakaguchi, M. Okuwa, and M. Akamatsu, "Prediction of driving behavior through probabilistic inference," in *Proceedings of the Eighth International Conference on Engineering Applications of Neural Networks*, pp. 8–10, Citeseer, 2003.
- [19] I. Dagli, 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.
- [20] A. Petrovskaya and S. Thrun, "Model based vehicle tracking for autonomous driving in urban environments," *Proceedings of Robotics: Science and Systems IV, Zurich, Switzerland*, 2008.
- [21] L. De Raedt, A. Kimmig, and H. Toivonen, "ProbLog: A probabilistic Prolog and its application in link discovery," in *Proceedings of the 20th international joint conference on artificial intelligence*, pp. 2462–2467, 2007.
- [22] Epic Games, "Unreal engine 3," <http://www.unrealtechnology.com/html/technology/ue30.shtml>, 2006.
- [23] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "USARSim: a robot simulator for research and education," in *Robotics and Automation*, 2007 *IEEE International Conference on*, pp. 1400–1405, IEEE, 2007.
- [24] J. Pool, "C#prolog – a prolog interpreter written in c#," 2009. <http://cs-prolog.sourceforge.net/>.
- [25] C. Kadie, D. Hovel, and E. Horvitz, "MSBNx: A component-centric toolkit for modeling and inference with bayesian networks," *Microsoft Research, Richmond, WA, Technical Report MSR-TR-2001-67*, vol. 28, 2001.