Hybrid Intelligence for Driver Assistance

Chung Hee Hwang†, Noel Massey, Bradford W. Miller†, Kari Torkkola

Motorola Labs
7700 S. River Parkway
Tempe, AZ 85284, U. S. A.
{chunghee.hwang, noel.massey, bradford.w.miller, kari.torkkola}@motorola.com

Abstract

We report on our on-going effort to build an adaptive driver support system, *Driver Advocate*^{TM}, merging various AI techniques, in particular, agents, ontology, production systems and machine learning technologies. The goal of DA is to help drivers have a safer, more enjoyable, and more productive driving experience, by managing their attention and workload. This paper describes the overall architecture of the DA system, focusing on how we integrate agent and machine learning technologies to make it support the driver intelligently and unobtrusively. The architecture has been partially implemented in a prototype system built upon a high-fidelity driving simulator, letting us run human experiments. The human driving data collected from the simulator are used as input to machine learning tools to make DA learn to adapt to the unique driving behavior of each driver. Once the DA demonstrates the desired capabilities, it will be tested in a real car in an actual driving environment.

Introduction

With the growing number of telematic devices found inside the cockpit – cell phones, navigators, internet access tools and PDAs, to name a few – the problem of driver distraction is becoming more and more serious. This has resulted in the acceleration of research in the area of intelligent transportation. In this vein, we are building a driver support system that is capable of monitoring the driver and the driving situation and intelligently assisting her to have safer, more enjoyable and more productive driving experiences.

Before discussing our approach to building the driver support system, let us briefly review the current research in the intelligent transportation domain. The goals of research effort in this area may be divided into autonomous vehicles, intelligent vehicles, and smart highways. Autonomous vehicles are automobiles that are equipped with sensors, computers and a control system to drive by themselves. The sensors allow the autonomous vehicles to perceive the environment – roads, pedestrians, other vehicles, potential hazards, etc. – as well as the driver, passengers and various components of the vehicle itself. The computers process the information from the sensors and determine actions to take, which are then executed by the control system. Although certain autonomous vehicles have shown impressive performance (e.g., Franke, et al., 1999; Thorpe, et al., 2002), we believe it will be some time before autonomous vehicles became practical. In the meantime, however, intelligent vehicles with the human in control appear to be more promising; these observe the traffic situation and support the driver by providing warnings and advice as needed (as discussed in, e.g., Dagli and Reichardt, 2002; Malec and Österling, 1996). Intelligent vehicles are also equipped with computers and sensors so they can perceive the environment, model and predict the driver's behavior, and take appropriate actions, e.g., issuing a warning when the driver dangerously deviates from her lane. The goal of smart highways or automated highway systems (Abreu et al., 2000; Ünsal, 1997; Varaiya, 1993) on the other hand is to make highways safer and more efficient through communication between vehicles and the control center. Intelligent vehicles could provide similar benefits through cooperation among themselves.

The goal of our project is to build an intelligent driver assistance system, the Driver Advocate^{TM} (henceforth, DA), to deploy in a car, as a step toward building an intelligent vehicle. Building an intelligent vehicle requires much of the same technology required to build autonomous vehicles, including the technologies in sensor, vision and image processing, context and situation awareness, task modeling, and inference and decision making capability. That is, an intelligent vehicle may be considered an 'autonomous vehicle that lacks control of the vehicle.' In addition, constructing the DA requires work in user modeling, intention recognition and an understanding of human factors.

In designing the DA, we have adopted an agent architecture, integrating it with an ontology-based knowledge representation system, production system-based decision making system, and technologies in adaptive learning. At this time, we skip the sensor and vision issues as we use a driving simulator that provides high-level environmental information for free, unlike the situation in an actual vehicle. We have partially implemented the design in a prototype system using a high-fidelity driving

†Current email addresses: {chwang, bradfordmiller}@mac.com

simulator (Figure 1). We have begun human experiments, and are collecting driving data as input to the machine learning tools to learn to adapt to the unique driving behavior of each driver.

The rest of the paper is organized as follows. We first provide a conceptual view of the DA system and its overall architecture. We then discuss the agent architecture and its interaction with machine learning tools, followed by further discussion on how machine learning techniques may be used to build an adaptive driver assistance system. A brief description of remaining issues and our plan for future work concludes the paper.

Figure 1. The Driving Simulator

The Driver Assistance System

As shown in Figure 2, the Driver Advocate™ needs to perform three core classes of tasks to successfully assist the driver: monitoring the driver ('Human Driver'), establishing the driver model ('User Model') and interacting with the driver ('Driver Interaction'). More specifically,

- Monitoring the Human Driver: This involves making assumptions about what information the driver is paying attention to (by monitoring her actions and behavior, such as eye gaze), in order to understand what she is reacting to, as well as inferring her intentions.
- "User" Modeling: The model of the driver is established and continuously updated. The DA will make predictions of what the driver should be doing, in terms of her attention, actions, mental state, etc., based on this model and the observations of the world (from the sensors in the car), in the context of the current situation.
- Driver Interaction: When the DA needs to communicate with the driver, the information should be presented in a timely manner so she could immediately act upon it. I.e., the DA has hard real-time constraints related to control of the vehicle especially when the vehicle is in motion. Also, the DA should present the information in a manner that is both clear and non-distracting to the driver. For example, it should not pull the driver's attention from a high priority task to a lower priority one.

Figure 2 further explains how these core tasks are related. The key to our design is the comparison between the prediction the DA makes of the driver action and the actual driver action (shown in the figure by the diamond indicating comparison). Because the DA's user model is not going to be completely accurate, its prediction may be probabilistic as well as involving timeframes. For instance, the DA may infer that, since the driver has looked at the bicycle that is 10 meters ahead, there is a 60% probability she will reduce speed and a 40% probability she will change lanes within the next second (the action must be taken within 3 seconds). If neither action has been taken after a second, the DA must assume that either the driver is in error and needs to be prompted to take a corrective action or that its user model is in error. Currently, the DA assumes that all errors identified are driver errors. We isolate model errors and revise the user model during a separate, off-line training phase. Online adaptation is our ultimate goal, but as (Miller, *et al.*, 2002) point out, maintaining system safety will require additional effort in the face of such adaptation, which we are not prepared to address at this time.

Given that a prediction has not obtained, we send the failed prediction to the Driver Interaction core, where it is used along with the information about the general situation and the current environmental information to select a communication operation to perform. This may be implemented using a variety of modalities, e.g., visual, audible, or even haptic. The core representation of such interactions is a speech act, which can also be used to update the user model in a reinterpretation stage.

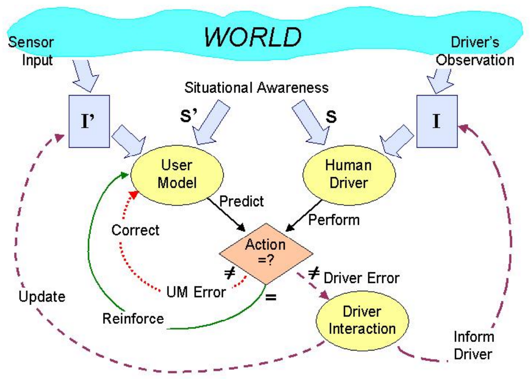

Figure 2. Conceptual view of the Driver Advocate™ System

The basis of our system architecture is intelligent agents, for reasons to be discussed later.[1] At the center is an upper and domain ontology shared by the agents which can be used for service advertisements and in agent negotiation to

[1] In fact, the agent architecture has been used in many systems in the intelligent transportation domain, e.g., see (Abreu et al., 2000; Shapiro, 2001; Sukthankar, *et al.*, 1998).

select the best service provider that meets service requirements (e.g., of response time, quality of result, etc.). Our agent system can support agents written in any language because only the agent communication language (ACL) and the ontology for message content are prescribed. Our ACL includes enough pragmatics to assure that the response to a particular speech act message will have the desired effect on the agents state and future performance. Some of our agents can encapsulate formal logical reasoning systems, some machine learning algorithms, and some are just purely reactive (particularly those critical to real-time response).

For the Driver Interaction core, we have adopted the Soar architecture (Lehman, *et al.,* 1996). It has been used in flexible cooperative team-based agent applications such as (Tambe, 1997) and in building driving agents (Aasman, 1995; Aasman and Michon, 1992). We plan on using the architecture as at least part of the User Modeling core as well.[2] Soar is capable of proposing and evaluating operations on the environment using a "psychologically plausible" architecture – which may be useful in creating better user models. At the present moment, however, our user model consists primarily of a model derived from supervised learning. We use our driving simulator to collect driving data from human drivers and distill them into a general driving model, which is used (when connected to the proper agents, i.e., driving agents) to effect driving the vehicle. In this fashion we are able to check and correct our model, by examining (as an observer) if the resultant driving is reasonable and natural, and can take palliative actions when it is not.

The Agent Architecture

We selected the agent architecture as it provides the best means to bridge together the various technologies and components required to build an intelligent driver support system. We first explain our choice of the agent architecture and then discuss some of the DA agents of particular interest.

There are several factors that drove the decision to choose the agent architecture. First, the DA system must be robust to unanticipated input and to dynamic reconfiguration. Due to the unconstrained task of driving, it is clearly not possible to predict all possible situations that drivers will find themselves in. Also, the system must be robust to sensor and component reconfiguration during its lifetime (for instance, sensors may break or become ineffective during driving). Unlike today's cars that usually remain configured the same as when they were manufactured (except for maybe a new stereo) modern cars permit much more flexibility in their configuration. The agent architecture supports this need of dynamic configuration as it allows agents to enter and leave the community transparently and enlist or delist other agents based on their capability and the quality of the service they provide.

[2] Soar has also integrated a learning mechanism, i.e., chunking, with rule-based symbolic reasoning, although we are not making use of it in the DA system at this time.

Second, conflict resolution: An intelligent driver support system should be able to deal with conflicting sensor information (e.g., GPS and momentum-based sensors may differ) and conflicting system recommendations (e.g., navigation system says *turn left* but collision avoidance system says *swerve right*). BDI (Belief, Desire, Intention)-type agents (Bratman, *et al*., 1988) can be used for conflict resolution by reasoning over why the agents are making the current suggestions using human cognitive models, resulting in a system that resolves conflict similar to how drivers resolve conflict. Additionally, there are conflicts with run-time resource constraints that may affect the quality of subsystem performance, which requires run-time decision making capabilities. For example, a vehicle radar system may be able to trade resolution for beam width. Agents are suitable for determining this tradeoff by using goal-based decisions to balance quality and frequency of sensor updates with known resources.

Lastly, easy integration with multiple platforms: During the development of the system, there will be multiple teams creating components using different software tools and languages. We also plan to move from a driving simulator to an actual instrumented vehicle where hardware may replace our simulation software. Since all agent communication takes place via text messages, any hardware or software components that can communicate through a network port can be used in the system. We are currently using TCP/IP with Java, C++, and Lisp code running on multiple platforms, but it can be easily switched to automotive bus protocol such as used by the CAN standard (ISO-11898, 1993).

The DA Agents

Table 1 shows some of the DA agents that are unique to our domain (Percept Agent, Soar Agent and Agenda Agent) together with User Interface Agent and Machine Learning Agent.

The User Model in Figure 2 is implemented with a Soar Agent, which maintains prioritized rules, and a Machine Learning agent, which generates probabilistic information based on the driver model (see the following section). The decision diamond in Figure 2 can be handled by the Soar Agent, which will have access to the results of the Machine Learning agent and may defer any decisions to the most probable result as determined by the Machine Learning agent. When the Soar Agent decides that the driver is in error, it sends a message to an Agenda Agent who will ultimately determine the best way (modality, timing, intensity, etc.) to inform the driver of this error.

Agent	Role
User Interface Agent	Converts user commands into speech acts for other agents. Eventually these may act in place of a user (for testing purposes) and be able to self-configure to best elicit a response from a user.
Machine Learning Agent	Receives data or rules, and generates prediction or classification using a variety of ML techniques.
Percept Agent	Obtains driving parameters, e.g., speed, heading, surrounding traffic, etc., from the simulator (but eventually from sensors).
Soar Agent	Wraps the Soar architecture. Also, translates percept or parameter value changes into data structures used by Soar and translates output actions from Soar into speech acts.
Agenda Agent	Alerts the driver to various problems using visual, audio, or haptic channel.

Table 1. DA Agents

Machine Learning in Advising a Driver

To avoid having to program the responses of a system to every imaginable situation, machine learning (ML) techniques are often used in intelligent advising systems, including the ones in automotive domain (see the survey by Campbell and Torkkola, 2002). We also use ML techniques to enable the DA to learn appropriate responses from the driving data. In this section, we describe how we apply ML technology to the DA, specifically, in learning user models, detecting driving/ driver states, and learning to interact with the driver.

User Modeling

Let us consider a simple scenario with a lane departure warning system, which adapts to the current driver's habit of driving within the lane. The system might be initially very eager to warn the driver of drifting to the shoulder, if she tends to drive close to the right lane edge. By adapting the user model based on a few minutes of her driving, the system demonstrates improved usability with reduced false alarm rate.

Learning and adaptation of a user model require collected or streaming data. The model could be, for example, a Hidden Markov Model (HMM), or just a set of linear regression coefficients, depending on the complexity of the driver characteristic chosen to be modeled and adapted. This direction will eventually be extended towards much more faithful modeling of the human driver, including many aspects of driving, as well as her intentions.

Recognizing the Driving States

The second direction is also closely tied to user modeling, and could be characterized as learning models/detectors for various driving situations, including the state of the driver. This could include discovery of meaningful driving states in an unsupervised fashion by means of clustering or HMMs, or supervised learning of driving states based on driving data that has been annotated with relevant labels. The model/detector can be one of many possibilities, such as a HMM, if history of the states is important, or any classifier for static data, such as a decision tree or a neural network. Annotation of the states should follow a predefined ontology for driving situations. Recognition and automatic labeling of these states can then loosely be thought as grounding the symbols in the ontology.

We have experimented with unsupervised state segmentation using a HMM. Figure 3 depicts a four-state HMM segmenting a driving path into states. Input variables to the HMM were physical parameters from the car, such as location, speed, acceleration, and the positions of all the controls in the car.

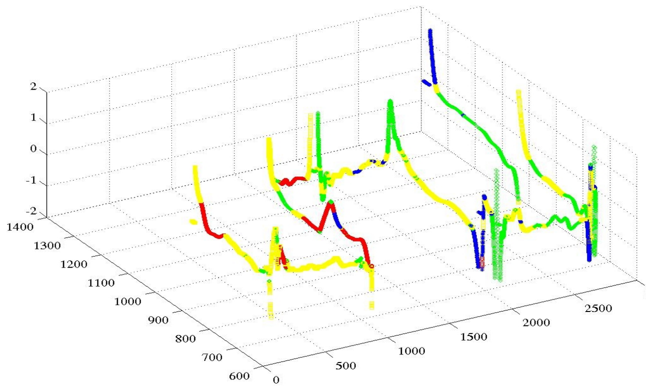

Figure 3. Unsupervised segmentation of driving states using HMMs. x, y axes are (geographical) driving coordinates, and z is the accelerator position (just to illustrate a third variable on the same graph). Segmentation is indicated by the color.

Learning to Interact with the Driver

One of the requirements of the DA system is to learn from the environment and adapt to it. Since the adaptation will mainly occur through user interaction - giving advice to the driver, and driver reacting to it - it is not possible to work with static collected databases only. Interaction may be either real interaction with human drivers who are driving the simulator or simulated interaction. (The state space in driving is too large to learn by exploration with human interaction only.) Our plan is to use simulated interaction initially, and a driving agent is being used for this purpose. We are envisioning the driving agent to be bootstrapped from, and constrained by models describing human driving, especially limitations of humans. This driving agent may also learn the bulk of its driving capabilities through interacting with the simulated world. Humanized driving agents make it possible to test and develop suitable learning methods in simulated worlds.

Concluding Remarks

We described the architecture of the Driver Advocate™ (DA) system, focusing on the integration of agent and machine learning technologies. The architecture has been partially implemented in a prototype system built upon a high-fidelity driving simulator. We have been running human driving experiments using the prototype DA so it may learn to adapt to individual variances in their driving behavior. Once the DA demonstrates the desired capabilities, we will test the system in a real car in an actual driving environment.

Much work needs to be done however, before the DA could be deployed in a real car assisting drivers in an actual driving environment. We conclude this paper by discussing some of the interesting challenges we are facing, or expect to face during the transitioning phase. First is the scalability issue, i.e., whether the results we obtain from simulated driving data will scale to the actual environment with real sensors. Unlike the 'high-level situational information' input we get from the simulator, the output from real sensors would be uncertain and of low-fidelity (cf. Sukthankar, 1997). Also, we may encounter grounding and epistemological problems, which would make object identification and situation recognition difficult requiring "deep" analysis. This could be a problem since the DA will need to work in real-time.

Another challenge is related to the vast amount and diverse nature of the data and knowledge the DA needs to deal with; from statistical or numeric data (driving data collected from the simulator, control data to send to the actuators in the vehicle, etc.) to knowledge in symbolic form (needed to describe task models and for planning and reasoning about situations). Although our architecture allows each agent to work using its own representation formalism, some kind of hybrid reasoning capability needs to be shared across multiple agents. Again, the efficiency issue could be a key factor due to the real-time requirement of the DA.

Finally, our machine learning tools will have to handle the problem of temporal credit. The goal of the advising system is reached only after a long sequence of actions, but what actions in the sequence will have to be credited/blamed for reaching/not reaching a specific goal? The training data is not labeled except one instance at the very end of the sequence when the goal is reached. What action is then to take at each possible state of the system in order to maximize the expected future reward (or to minimize penalty)? This happens to be the exact problem setting for reinforcement learning (Sutton and Barto, 1999), and we are currently investigating various options to apply reinforcement learning to our problems. The main idea is to use a trained driving agent to drive in a simulated world with simulated incidents and high simulated cognitive loads, and let the DA system give advice according to the current policy and adjust its policy according to the reward it receives.

References

Aasman, J. 1995. *Modeling Driver Behaviour in Soar.* Ph.D. Thesis, Rijksuniversiteit Groningen, Leiden, The Netherland.

Aasman, J. & Michon, J.A. 1992. "Multitaking in Driving," in *Soar: A Cognitive Architecture in Perspective*. Klewer Academic Pub.

Abreu, B., Botelho, L, Cavallaro, A., Douxchamps, D., Ebrahimi, T., Figueiredo, P., Macq, B., Mory, B., Nunes, L, Orri, J., Trigueiros, M.J. & Violante, A. 2000. "Video-based multiagent traffic surveillance system," *Proc. Intelligent Vehicles Conf.*, Dearborn, MI.

Bratman, M.E., Israel, D.J. & Pollack, M.E. 1988. "Plans and resource-bounded practical reasoning," *Computational Intelligence, v4.*

Campbell, W. M. & Torkkola, K. 2002. "Machine learning for advising a driver: A Survey." *Proc. Internat. Conf. on Machine Learning & Applications*, Las Vegas, NV.

Dagli, I. & Reichardt, D. 2002. "Motivation-based approach to behavior prediction." *Proc.IEEE Intelligent Vehicle Symp. (IV-2002)*, Versailles, France.

Franke, U., Gavrila, D., Görzig, S., Lindner, F., Paetzold, F. & Wöhler, C. 1999. "Autonomous driving approaches downtown," *IEEE Intelligent Systems, v13.*

ISO-11898. 1993. *Road Vehicles – Interchange of Digital Information – Controller Area Network (CAN) for High Speed Communication (TC 22/SC3 ICS 43.040.15).*

Lehman, J.F., Laird, J.E. & Rosenbloom, P.S. 1996. "A Gentle introduction to Soar: An Architecture for human cognition," in *Invitation to Cog.Sci.*, v*4*. MIT Press.

Malec, J. & Österling, P. 1996. *Driver Support System for Traffic Manoeuvres.* Dept. Comp. & Info. Sci., Linköping U., Sweden.

Miller, B.W., Massey, N. & Gardner, R.M. 2002. "Safe adaptation in an automotive vehicle: The Driver Advocate™," *Proc. Workshop on Safe Learning Agents, AAAI Spring Symp*.

Shapiro, D.G. 2001. *Value-Driven Agents*. Ph.D. Thesis, Stanford U., Stanford, CA.

Sukthankar, R. 1997. *Situation Awareness for Tactical Driving.* Ph.D. Thesis, CMU, Pittsburgh, PA.

Sukthankar, R., Baluja, S. & Hancock, J. 1998. "Multiple adaptive agents for tactical driving," *Internat. J. of A.I., v9.*

Sutton, R.S. & Barto, A.G. 1999. *Reinforcement Learning*. MIT Press.

Tambe, M. 1997. "Agent architectures for flexible, practical teamwork," in *Proc. AAAI-97*.

Thorpe, C., Aufrere, R., Carlson, J.D., Duggins, D., Fong, T.W., Gowdy, J., Kozar, J., MacLachlan, R., McCabe, C., Mertz, C., Suppe, A., Wang, C. & Yata, T. 2002. "Safe robot driving," in *Proc. ICMA-2002.*

Ünsal, C., Sukthankar, R. & Thorpe, C. 1997. "Functional sensor modeling for Automated Highway Systems simulations," presented at *SPIE Internat. Symp. on Intelligent Systems & Advanced Manufacturing,* Pittsburgh, PA.

Varaiya, P. 1993. "Smart cars on smart roads: Problem of Control," *IEEE Trans. on Automatic Control, v38.*