Toward a Unified Framework for Tracking Cognitive Processes

Dario D. Salvucci (salvucci@cs.drexel.edu)
Department of Computer Science, Drexel University
3141 Chestnut St., Philadelphia, PA 19104

Timothy Siedlecki (ts62@drexel.edu)
Department of Computer Science, Drexel University
3141 Chestnut St., Philadelphia, PA 19104

Abstract

In this paper we present an initial specification of a general, robust, and efficient computational framework for tracking cognitive processes — that is, inferring a persons' thoughts from their actions. Our framework, which we call the *mind-tracking architecture*, centers on two core processes: generating predicted cognitive and action sequences using computational cognitive models, and tracking observed actions through robust matching with predicted actions. In essence, the mind-tracking architecture "thinks along" with the person in predicting a set of possible thoughts and actions, and then matches these to the person's observed actions to infer their most likely thoughts. In the paper we provide a background of related work (e.g., for intelligent tutoring systems), outline the basic components of the architecture, and demonstrate its usefulness for a sample real-world application — real-time inference of driver intentions.

Introduction

Whenever people interact, they continually infer others' thoughts and intentions from their actions — sometimes in mundane ways (e.g., watching a friend wave "hello"), sometimes in complex and subtle ways (e.g., watching a poker player for signs of bluffing). This process of intent inference is not only a core component of human communication but also of human-computer interaction: when human users and computer systems interact, the systems must continually observe user actions and respond to the inferred thoughts and intentions behind these actions. For example, the current generation of intelligent tutoring systems center around the notion that the system can infer student intentions and knowledge in order to adapt and personalize the learning experience (e.g., Anderson et al., 1995; Anderson & Gluck, 2001; Frederiksen & White, 1990). As another example, adaptive help systems in intelligent user interfaces monitor user behavior and provide focused contextual help when prompted or even spontaneously (e.g., Horvitz et al., 1998). Such systems essentially mimic the role of human tutors, assistants, etc. in continually inferring another's intentions to best adapt and respond to another's communication.

Researchers have addressed the problem of intent inference using a variety of techniques that integrate cognitive validity with pattern matching algorithms. Work on intelligent tutoring systems has utilized "model tracing" or "tracking" algorithms (see Anderson et al., 1995, for a review) that follow a student step-by-step in a problem solution, in addition to "knowledge tracing" algorithms that keep a current estimate of student knowledge and skill. As successful as these systems have been in real-world deployment, the underlying model-tracing algorithms have two major assumptions that limit their generality: (1) all actions are discrete events without noise, and (2) each action corresponds one-to-one to a skill-related rule without ambiguity and limited backtracking. Recent interesting work on tutoring systems (e.g., Conati, Gertner, & VanLehn, 2002) and intelligent help systems (e.g., Horvitz et al., 1998) has used probabilistic Bayesian networks to make inference more robust and flexible in the face of noise and ambiguity; however, this work focuses more on the "knowledge tracing" aspect and still relies on the assumption that observations are discrete and unambiguous. In contrast, a great deal of multimodal data collected in empirical studies and real-world interactive systems are continuous and noisy — for instance, mouse movements, hand positions and gestures, eye movements, even task-specific data such as steering and acceleration/braking for driving (to be discussed later). A general framework for tracking cognitive processes must be able to handle such noisy, continuous data in order to use a full complement of multimodal data for inference.

In this paper we propose a computational framework called the *mind-tracking architecture* for modeling and tracking human cognitive processes — that is, mapping observed actions to the unobservable thoughts that produced them. The mind-tracking architecture centers on two core processes: *model simulation* and *action tracking*. First, at each time step, the architecture generates a set of predicted thought and action sequences by running computational cognitive models in simulation. Second, the architecture finds the best match between the predicted action sequences and the actual observed sequence, thereby inferring the most likely cognitive process (i.e., thought sequence) that generated the observed actions. Thus, the mind-tracking architecture "thinks" along with people as they behave, mirroring their thoughts and actions and providing a continual estimation of the person's current cognitive state, including intentions, goals, knowledge, etc. In doing so, the architecture provides a unified framework for robustly

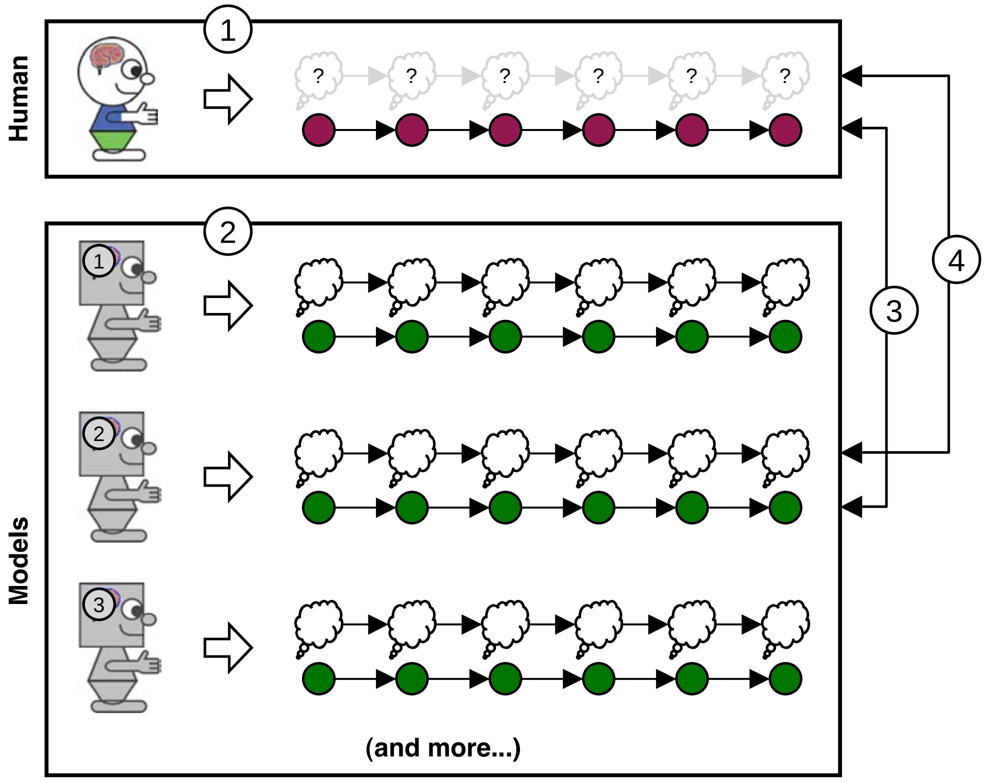

Figure 1: Overview of the mind-tracking architecture.

relating observable, low-level multimodal actions to high-level cognitive processes.

This paper outlines the mind-tracking architecture and demonstrates the core ideas in the context of a real-world application domain: driving. Inference of driver intentions is essential for the development of future intelligent vehicle systems that warn and assist drivers in critical situations, very much analogous to inference of student intentions and knowledge in an intelligent tutoring system. However, driver intent inference must handle an inundating stream of multimodal data — including steering, acceleration, braking, eye movements, even facial expressions — which are notoriously rife with noise and individual variability, making current model-tracing techniques very inadequate for the driving domain. Using a rigorous model of human driver behavior (Salvucci, Boer, & Liu, 2001) developed in the ACT-R architecture (Anderson & Lebiere, 1998), we demonstrate how the mind-tracking architecture allows for robust, accurate inference of driver behavior, specifically the problem of detecting driver lane changes.

The Mind-Tracking Architecture

In its essence, the mind-tracking architecture continually maps a person's observed actions to their inferred thoughts and intentions. We first outline the representation of the computational cognitive models that allows us to represent cognitive state as well as generate sequences of predicted thoughts and actions. We then describe the process by which the architecture uses these models to follow along with a person's observed actions and thus infer their predicted thoughts.

Model Representation

Because the mind-tracking architecture is a computational framework in which to infer thoughts, we require some computational representation for thoughts or more precisely "cognitive state" — including a person's goals, intentions, knowledge, perhaps even mood, etc. While we might imagine a number of existing models and frameworks serving this role, we have chosen the ACT-R cognitive architecture (Anderson & Lebiere, 1998), a production-system architecture based on condition-action rules and separate declarative (factual) and procedural (skill) knowledge stores. The ACT-R architecture has been used to model behavior in tasks ranging from simpler laboratory tasks (e.g., memory and recall: Anderson & Matessa, 1997) to more complex dynamic real-world tasks (e.g., air-traffic control: Lee & Anderson, 2001).

Due to space limitations, we must refer interested readers to Anderson and Lebiere (1998) for more details on the ACT-R architecture. However, we note three important features of ACT-R (and similar cognitive architectures such as Soar: Laird, Newell, & Rosenbloom, 1987, and EPIC: Meyer & Kieras, 1997) that make it an excellent choice for purposes of mind tracking. First, the architecture allows for computational models that run in simulation and generate temporal sequences of "thoughts," namely rule firings. At any point in time, the current state of the model serves as the current "cognitive state" in that it embodies all current knowledge and skills, including symbolic knowledge (e.g., whether a given fact is known at all) as well as subsymbolic knowledge (e.g., whether a given fact can be readily recalled). Second, ACT-R can produce interaction with real-world or simulated stimuli though realistic perceptual and motor modules (Byrne & Anderson, 2001), often running in identical environments as human subjects. This feature allows for direct comparison between human behavior and model predictions for real-world metrics. Third, the architecture facilitates modeling at multiple grain sizes and levels of details — for instance, both at the level of lower-level control (e.g., steering and braking) and higher-level cognition (e.g., decision making and planning). These and other features allow for direct integration of ACT-R models into the mind-tracking framework.

Mind tracking runs not one but many cognitive models simultaneously. Roughly speaking, the set of cognitive models approximates the variability of strategies and behaviors seen in a population of human performers such that the models represent as many different behaviors and strategies as possible. The issue of variability and individual differences is an extremely involved one that raises many questions concerning where the variability arises (e.g., "hardware" versus "software" differences: see Meyer et al., 2001) and how much to represent explicitly in the models. A rigorous treatment of variability and individual differences is beyond the scope of this paper; however, as we will see in the application to driving, the architecture offers several broad choices for handling variability: explicitly representing different strategies in different models, implicitly representing different behaviors by varying model parameters, and allowing the matching process in action tracking to account for unexplained noise.

Mind Tracking

Given the desired set of cognitive models, the mind-tracking process operates in four stages: (1) data collection: recording the person's observed actions; (2) model simulation: generating predicted thoughts and actions for all models; (3) action tracking: finding the best matching predicted actions for the observed actions; (4) thought inference: identifying the inferred thoughts before cycling the process in the next time step. We now describe each step of this process, as illustrated in Figure 1.

(1) Data Collection: recording the person's observed action sequence for a given period of time. The actions recorded would include any multimodal data relevant to the task, stored as a vector of numerical or categorical information. The period of time is also task-specific and related to the temporal grain size of the multimodal data, though a value around one half to several seconds should suffice for most purposes. Of course, as illustrated in the figure, the thought sequence that generated these actions is unknown and will be inferred through the mind-tracking process.

(2) Model Simulation: generating the models' predicted thought and action sequences. Starting at the same initial state as the human data sequence, each model is run for a period of time to generate its unique model simulation trace. Each trace includes both a thought sequence and an action sequence, where the thought sequence comprises states of the cognitive model and the action sequence comprises vectors of observable data. The model action sequences should be completely analogous to the human action sequence in terms of the amount and type of data, the sampling rate, and the total sampled time.

(3) Action Tracking: identifying the model whose predicted action sequence best matches the person's observed action sequence. Action tracking could be performed using any of a number of techniques. For this initial treatment of mind tracking, we assume that all features in the multimodal data are continuous values and compute the sum of squared error between the observed and predicted action sequences; thus, the best-matching model is the model with the least error between its predicted actions and the person's actions. (For future work, we are currently investigating more complex methods of action tracking — for instance, using a sequence-distance metric or complex probabilistic models.)

(4) Thought Inference: identifying the inferred thought sequence. Given the best-matching model from action tracking, this model's action sequence is associated with its corresponding predicted thought sequence, resulting in the desired inferred thoughts given the observed actions. We can then analyze the cognitive state embodied in this sequence to determine current goals, intentions, etc. For instance, by using ACT-R as our model representation, we can determine the person's inferred intentions at the end of the tracking time period by looking at the final cognitive model's "goal buffer" (a buffer that contains a link to the current declarative goal chunk). As another example, we could check whether a particular unit of knowledge is currently in memory, such as declarative chunk for factual knowledge or a production rule for skill knowledge.

Taking the final state (or any intermediate state) as the new current state, the system cycles the mind-tracking process, repeating all four steps at this next time increment.

Discussion

As mentioned, the core idea of the mind-tracking architecture is similar to a large body of work in intelligent tutoring, namely work on model tracing (Anderson et al., 1995) and tracking (Frederiksen & White, 1990). For example, the model-tracing cognitive tutors developed at Carnegie Mellon (Anderson et al., 1995) use a single cognitive model with multiple decision points and follow a student step-by-step through the problem-solving process. However, the cognitive tutors assume a direct mapping between observed actions and cognitive rules — for instance, a mapping between the student subtracting a number from both sides of an equation to a rule that embodies this skill in the equation-solving domain. In addition, the tutors do not handle ambiguity when an observed action could potentially map to multiple rules; in such a situation, the tutor is forced to prompt the user to determine which rule is intended. The mind-tracking architecture generalizes the model tracing process to allow for arbitrary mapping between observable actions and rule firings even in the presence of data noise.

Another body of work closely related to mind tracking is that of sequential protocol analysis, with origins as far back as Newell and Simon's (1972) seminal work on human problem solving. Others have explored interpreting specific types of protocol data, such as verbal protocols (Waterman & Newell, 1971) and eye movements (Salvucci & Anderson, 2001), with such techniques as sequence-distance matching and hidden Markov models (see Salvucci & Anderson, 2001, for a review). Mind tracking integrates the core ideas of these areas of literature and integrates them into a unified framework for high-density streams of multimodal data.

Sample Application: Driver Intentions

To put the mind-tracking architecture in context and to demonstrate its usefulness in a real-world domain, we performed a study of how to apply the architecture in the domain of driving — in particular, recognizing drivers' intent to change lanes. This task may seem small at first in comparison to the generality of the architecture; however, lane-change recognition is in fact quite difficult and serves as an excellent first test for two important reasons. First, lane-change recognition has proven very difficult even for complex probabilistic models intended for just such a purpose. For instance, Pentland and Liu (1999) developed hidden Markov models (HMMs) to recognize lane changes with reasonable accuracy, but this work analyzed discrete time steps and was not easily generalizable to continuous, real-time recognition. Second, the driving domain involves a large amount of noisy multimodal data (e.g., steering, pedal depression, etc.) that cannot be handled by earlier model tracing algorithms. Thus, inference of driver lane changes serves as an excellent challenge for the mind-tracking architecture on a difficult problem and an excellent demonstration of its advantages over earlier methods.

Driver Model

Mind tracking requires that we have a computational model capable of performing the given task and predicting the same types of action data collected from human subjects. For this purpose, we used a version of the ACT-R integrated driver model (Salvucci, Boer, & Liu, 2001), a computational model of highway driving; specifically, we used a C++ implementation of the original LISP model with the same core architecture and task-specific knowledge base. The driver model navigates in a simulated multi-lane highway environment and incorporates three primary modules: *control* for lower-level perception, steering, and acceleration; *monitoring* for perception of the environment and maintenance of situation awareness; and *decision making* for choice actions that depend on the current perceptual and environmental variables. Because of its implementation in the ACT-R cognitive architecture, the driver model runs in simulation using "eyes" for perception and "hands" and "feet" for steering and pedaling. Simulations produce identical protocols to human drivers navigating the same environment in a driving simulator (see Salvucci, Boer, & Liu, 2001). The model has been validated with respect to lane changing, curve negotiation, and even the effects of driver distraction (e.g., Salvucci, 2001). This well-validated model thus serves well as the basis for our mind-tracking study of driving.

Setup and Procedure

Using the driver model as a basis, we began the setup of the mind-tracking architecture by specifying the model sets with which to detect lane changes:

- **Driver-Only**: This set contained only the default driver model with no modifications. The mind-tracking architecture has no choice but to select the same model in action tracking, thus this set represents a baseline for inference in that it tells us how accurate the driver model reflects human performance.

- **Stayer-Changer**: This set explicitly represented the two possible intentions by including two models: a "stayer" model that always tries to remain in its current lane, and a "changer" model that always tries to change lanes. After a lane change, both models are re-synchronized to drive in the correct lane.

- **Variable-16**: This set implicitly represented the set of possible behaviors by creating a set of 4^2=16 models representing variations of the model's four task-specific parameters: the desired time-headway (THW) for following a lead car, the desired THW before starting a pass maneuver, the minimum cut-in distance from other cars when changing lanes, and a parameter describing how aggressively the model steers into the other lane. In this set, each parameter was varied by either 1.5 or 0.75 times its default value, thus producing 16 driver models with unique parameter settings.

- **All-19**: This set represented a combination of the Driver-Only, Stayer-Changer, and Variable-16 sets, for a total of 19 unique driver models.

Using each of these model sets, we ran the mind-tracking process on driving data collected in a previous study (Salvucci, Boer, & Liu, 2001). In this previous study, 11 human drivers navigated a highway environment with moderate traffic for approximately 30 minutes and could change lanes whenever they desired. In our study, we used only 6 of the 11 subjects for which turn-signal data was available (due to a glitch in the original data collection). One important feature of this data set was the inclusion of verbal reports, specifically markings for when drivers voiced their intentions to begin (and end) a lane change; this feature allowed us to compare directly between when the driver started a lane change and when the mind tracking first recognized the change of intention.

For mind tracking, we used three primary features from the original driving data: steering angle (in radians), accelerator depression [0-1], and brake depression [0-1]. In addition, for some of the tests, we also included both turn signals with a value of 1 for on and 0 for off. To compute the error needed for action tracking, we weighted steering angle by a multiplicative factor of 5 (to compensate for its smaller relative value) and then added the squared errors for each feature to arrive at the total error.

Results

We analyzed three aspects of how the mind-tracking architecture recognized driver lane changes, all with separate analyses without turn signals and with turn signals to examine the signals' contribution to tracking. First, Table 1 shows the *delay time* as the time (in seconds) needed for the model to detect a lane change. Delay time was calculated as the elapsed time between (1) the driver stating an intention to change lanes (as noted in the verbal protocol markings) and (2) the first time the best-matching model indicated a lane change, or the first time the vehicle crossed into the other lane (the ground truth). The Driver-Only and Variable-16 sets both required over 1 second to detect a lane change; the average time for an entire lane change was 5 seconds, and thus even these methods recognized the lane change approximately one-fifth of the way through the maneuver. The Stayer-Changer and All-19 sets both performed much better without turn signals, with delay times around .70 s. All sets performed better with turn signals than without (except the Driver-Only set, which always chooses its one model), and the Stayer-Changer and All-19 sets again performed particularly well with delay times around .55 s with turn signals.

While delay time is a reasonable indication of the speed of lane-change recognition, it may be deceptive in that the vehicle may have not shifted far during the delay, making it hard to fault the recognition system for lack of detection. To clarify this point, Table 2 shows the *lateral distance* traveled before recognition — that is, the distance traveled across the lane (i.e., orthogonally to the direction of travel) during the delay time — with separate results included for lane changes to the left and those to the right. Once again the Stayer-Changer and All-19 sets outperform the Driver-Only and Variable-16 sets across the board, for right-to-left (R→L) and left-to-right (L→R) changes and with and without turn signals. Turn signals again aided recognition, although not as clearly: distances were higher for right lane changes for the two best sets. Interestingly, lateral distances for R→L changes (to pass a lead car) were generally much larger than those for L→R changes (to return to the travel lane). The explanation arises in where drivers started their lane changes: on average, they started R→L changes exactly in the center of the right lane (position 0.50 for lane position [0,1]) but started L→R changes on the right side of the left lane (position 0.36); the larger resulting discrepancy between steering signals needed to change lanes versus stay in the lane make the L→R changes easier to detect quickly.

Table 3 shows the *time ratios* for correct predictions, namely the ratio of time during either lane keeping (LK) or lane changing (LC) for which mind tracking generated the correct intention. Again, the Stayer-Changer and All-19 sets outperform the others both without and with turn signals. Overall, the process does not perform perfect recognition, but the numbers do show some promise: a real-world system would not rely on or even expect perfect predictions, but rather would use the predictions as a suggestion for how to proceed or possibly integrate the predictions into a downstream probabilistic system that can alleviate the variability in the process.

Table 1: Delay times for recognition, in seconds.

Model Set	Without TS	With TS
Driver-Only	1.17	1.17
Stayer-Changer	.69	.54
Variable-16	1.07	.93
All-19	.70	.57

Table 2: Lateral distance before recognition for right-to-left (R→L) and left-to-right (L→R) lane changes, in meters. (Note that the lane width in the original study was 4 m.)

Model Set	Without TS		With TS	
	R→L	L→R	R→L	L→R
Driver-Only	1.12	.72	1.12	.72
Stayer-Changer	.52	.12	.20	.28
Variable-16	1.08	.40	.96	.36
All-19	.64	.08	.40	.12

Table 3: Time ratios for correct prediction for lane keeping (LK) and lane changing (LC).

Model Set	Without TS		With TS	
	LK	LC	LK	LC
Driver-Only	.70	.40	.70	.40
Stayer-Changer	.92	.57	.84	.61
Variable-16	.84	.39	.87	.45
All-19	.91	.53	.96	.61

Discussion

Overall, we were very pleased with this first attempt at applying the mind-tracking architecture to a real-world practical problem. Although there is little previous work with which to compare our results, two previous attempts at lane-change recognition come to mind, both using a very different technique (hidden Markov models). First, Pentland and Liu (1999) reported that their recognizer could detect changes in the first half second of the maneuver; their recognizer performs only analysis of discrete segments, as opposed to continuous recognition (as we do), but nevertheless this result maps well onto our best recognition times. Second, Kuge et al. (2000) reported similar findings for their continuous recognition system; however, their system uses only steering-based features and has no knowledge of the surrounding environment, which clearly affects whether and when people make lane changes. The mind-tracking architecture implicitly takes environment into account in that the model predictions are based on perception of the current environment. Thus, mind tracking offers a novel method of inferring driver intent that compares well with previous results and generalizes well to other complex intentions for lower-level maneuvers (e.g., turns) and higher-level decisions (e.g., route planning).

Conclusions

The mind-tracking architecture provides a unified framework for inferring people's thoughts from their actions. As such, it represents an extremely broad area of work with many degrees of freedom, and this initial treatment only scratches the surface of a potentially fruitful area of research. As just one example, we have chosen the ACT-R architecture as the cognitive model representation; however, we might just as easily employ another cognitive architecture (e.g., Soar: Laird, Newell, & Rosenbloom, 1987, or EPIC: Meyer & Kieras, 1997) and use models developed for these architectures — for instance, Aasman's (1995) Soar driver model. Our upcoming work will explore many of these degrees of freedom and continue applying this theoretical work to practical domains such as driving, intelligent tutoring, etc. For now, this initial work shows good promise for the mind-tracking architecture in providing a novel way of representing and thinking about the inference of human thoughts and intentions.

Acknowledgments

This work was supported in part by National Science Foundation Grant #IIS-0133083 and a grant from Nissan Motor Company, both to the first author.

References

Aasman, J. (1995). *Modelling driver behaviour in Soar*. Leidschendam, The Netherlands: KPN Research.

Anderson, J. R., Corbett, A. T., Koedinger, K., & Pelletier, R. (1995). Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, *4*, 167-207.

Anderson, J. R., & Gluck, K. A. (2001). What role do cognitive architectures play in intelligent tutoring systems?. In D. Klahr & S. M. Carver (Eds.), *Cognition and Instruction: 25 Years of Progress*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Hillsdale, NJ: Erlbaum.

Anderson, J. R., & Matessa, M. P. (1997). A production system theory of serial memory. *Psychological Review*, *104*, 728-748.

Byrne, M. D., & Anderson, J. R. (2001). Serial modules in parallel: The psychological refractory period and perfect time-sharing.. *Psychological Review*, *108*, 847-869.

Conati, C., Gertner, A., & VanLehn, K. (2002). Using Bayesian networks to manage uncertainty in student modeling. *Journal of User Modeling and User-Adapted Interaction*, *12*.

Frederiksen, J. R., & White, B. Y. (1990). Intelligent tutors as intelligent testers. In N. Frederiksen, R. Glaser, A. Lesgold, & M. G. Shafto (Eds.), *Diagnostic Monitoring of Skill and Knowledge Acquisition* (pp. 1-25). Hillsdale, NJ: Lawrence Erlbaum Associates.

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., & Rommelse, K. (1998). The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence* (pp. 256-265). San Francisco, CA: Morgan Kaufmann.

Kuge, N., Yamamura, T., Shimoyama, O., & Liu, A. (2000). A driver behavior recognition method based on a drive model framework. In *Proceedings of the Society of Automotive Engineers World Congress 2000*.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, *33*, 1-64.

Lee, F. J., & Anderson, J. R. (2001). Does learning of a complex task have to be complex? A study in learning decomposition. *Cognitive Psychology*, *42*, 267-316.

Meyer, D. E., Glass, J. M., Mueller, S. T., Seymour, T. L., & Kieras, D. E. (2001). Executive-process interactive control: A unified computational theory for answering twenty questions (and more) about cognitive ageing. *European Journal of Cognitive Psychology*, *13*, 123-164.

Meyer, D. E., & Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms. *Psychological Review*, *104*, 3-65.

Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice Hall.

Pentland, A., & Liu, A. (1999). Modeling and prediction of human behavior. *Neural Computation*, *11*, 229-242.

Salvucci, D. D. (2001). Predicting the effects of in-car interface use on driver performance: An integrated model approach. *International Journal of Human-Computer Studies*, *55*, 85-107.

Salvucci, D. D., & Anderson, J. R. (2001). Automated eye-movement protocol analysis. *Human-Computer Interaction*, *16*, 39-86.

Salvucci, D. D., Boer, E. R., & Liu, A. (2001). Toward an integrated model of driver behavior in a cognitive architecture. *Transportation Research Record*, 1779.

Waterman, D. A., & Newell, A. (1971). Protocol analysis as a task for artificial intelligence. *Artificial Intelligence*, *2*, 285-318.