HUMAN-COMPUTER INTERACTION, 2001, Volume 16, pp. 39–86

Automated Eye-Movement Protocol Analysis

Dario D. Salvucci and **John R. Anderson**
Carnegie Mellon University

ABSTRACT

This article describes and evaluates a class of methods for performing automated analysis of eye-movement protocols. Although eye movements have become increasingly popular as a tool for investigating user behavior, they can be extremely difficult and tedious to analyze. In this article we propose an approach to automating eye-movement protocol analysis by means of *tracing*—relating observed eye movements to the sequential predictions of a process model. We present three tracing methods that provide fast and robust analysis and alleviate the equipment noise and individual variability prevalent in typical eye-movement protocols. We also describe three applications of the tracing methods that demonstrate how the methods facilitate the use of eye movements in the study of user behavior and the inference of user intentions.

1. INTRODUCTION

In the study of human–computer interaction (HCI) and cognition in general, protocol analysis is a widely popular and successful method of inferring how hu-

Dario Salvucci is a computer scientist with an interest in inferring intentions from actions; he is currently a Research Associate at Cambridge Basic Research and will soon be an Assistant Professor of Math and Computer Science at Drexel University. **John Anderson** is a psychologist with interests in computational modeling and cognitive architectures, particularly in the ACT–R framework; he is a Professor of Psychology at Carnegie Mellon University.

CONTENTS

1. INTRODUCTION
1.1. Eye Movements as Protocols
1.2. Tracing Eye-Movement Protocols
2. BACKGROUND: EYE MOVEMENTS AND PROTOCOL ANALYSIS
2.1. Eye Movements
Saccadic Eye Movements
Eye Movements as Data
Eye-Movement Data Analysis
2.2. Protocol Analysis and Tracing
Methodological Foundations
Automated and Semiautomated Analysis
3. TRACING EYE-MOVEMENT PROTOCOLS
3.1. Tracing Setup
3.2. Tracing Methods
Target Tracing
Fixation Tracing
Point Tracing
3.3. Issues
3.4. Summary
4. CASE STUDIES
4.1. Equation Solving
Experiment
Protocol Analysis
Classifying Strategies
4.2. Reading
Experiment
Protocol Analysis
Comparing Models
Cleaning Data
4.3. Eye Typing
Experiment
Protocol Analysis
Inferring Intent Online
Facilitating User Input
4.4. Summary
5. GENERAL DISCUSSION
5.1. Tracing in HCI Research
5.2. Method Comparison and Recommendations

mans reason and solve problems. Protocols—sequences of actions recorded during the execution of some task—help researchers determine the cognitive strategies involved in performing a task. Most protocol studies to date have emphasized either verbal protocols (e.g., Newell & Simon, 1972; Peck & John, 1992;

Ritter & Larkin, 1994) or protocols of manual actions such as mouse clicks and key presses (e.g., Card, Moran, & Newell, 1983). However, due to improved eye-tracking equipment and better understanding of how to exploit it, eye-movement protocols have enjoyed burgeoning attention as a tool for studying HCI and human behavior in general. For instance, researchers have studied eye movements to understand user behavior in basic interface tasks (Aaltonen, Hyrskykari, & Räihä, 1998; Byrne, Anderson, Douglass, & Matessa, 1999), to reveal how users encode and process information (Lohse & Johnson, 1996), and to infer user intent in real-time interfaces (Goldberg & Schryver, 1995; Jacob, 1991, 1995). However, with notable exceptions (e.g., Goldberg & Kotval, 1998), there has been very little work on specifying an explicit formal methodology for rigorous analysis of eye-movement protocols.

In this article we introduce a class of methods that automate the analysis of eye-movement protocols. The methods analyze eye movements by using a powerful form of protocol analysis called *tracing*—relating observed protocols to the sequential predictions of a process model (Anderson, Corbett, Koedinger, & Pelletier, 1995; Ohlsson, 1990; Ritter & Larkin, 1994). A number of researchers have developed tracing techniques to understand and model user behavior (Card et al., 1983; Ritter & Larkin, 1994), to parse protocol data into more manageable forms (Bhaskar & Simon, 1977; Garlick & VanLehn, 1987; Smith, Smith, & Kuptsas, 1993; Waterman & Newell, 1971), and to infer user intent and knowledge state in a real-time interface (Anderson et al., 1995; Frederiksen & White, 1990). Although their techniques work very well for basic user actions (e.g., mouse clicks, key presses) and reasonably well for verbal protocols, they cannot handle the noise and variability in typical eye-movement data. The proposed methods alleviate this noise and variability to provide efficient and robust analysis of eye-movement protocols.

To demonstrate the uses and benefits of the tracing methods, this article describes case studies of the tracing methods to three application domains: equation solving, reading, and gaze-based interfaces. The equation-solving study evaluates the tracing methods in a domain with a limited number of visual encoding strategies, comparing their interpretations to those of expert human coders. The reading study evaluates the tracing methods in a domain with more complex encoding strategies, using the methods to compare two existing models of reading and to facilitate aggregate analysis of reading data. The gaze-based interface study emphasizes real-time interpretation of user intent to develop more efficient and user-friendly interfaces. Although these case studies certainly do not cover the range of all possible uses for the tracing methods, they nicely illustrate their most important uses both for offline analysis of user actions, to facilitate understanding and modeling of user behavior, and for online inference of user intentions, to facilitate design and implementation of intelligent user interfaces.

1.1. Eye Movements as Protocols

Eye movements provide a wealth of information regarding how people acquire and process information. Eye movements are especially convenient because data can be collected at a fine temporal grain size and users need little instruction and training to produce informative data. Unfortunately, eye-movement protocols are very time-consuming and tedious to analyze for at least three reasons. First, like verbal protocols, several trials of even a simple task can generate enormous sets of eye-movement data, all of which must be coded into some more manageable form for analysis. Second, eye-movement protocols typically include a great deal of equipment noise due to common calibration errors and difficulties that arise with current eye trackers. Third, even if we had highly accurate eye trackers, eye-movement analysis still would be challenging because of the high degree of individual variability in people's visual scanning strategies. For large eye-movement data sets with hundreds or thousands of protocols, it is simply impossible for humans to interpret the data consistently, accurately, and in a reasonable amount of time.

We can illustrate these issues with two sample eye-movement protocols. Figure 1a shows a sample screen and protocol taken from an equation-solving task (detailed later) that might arise in an intelligent tutoring system. In the task, students solved equations of the form $a\mathbf{x}/B = A/b$ by computing the result $\mathbf{x} = (A/a)(B/b)$. For instance, the equation in Figure 1a, $5\mathbf{x}/9 = 35/3$, can be solved as $\mathbf{x} = (35/5)(9/3) = 21$. The eye-movement protocol in the figure comprises the student's alternating fixations (long pauses over informative items) and saccades (rapid movements between fixations). Each point in the protocol represents the student's point of gaze as sampled by an eye tracker; fixation points are drawn larger than saccade points, and earlier points are drawn darker than later points. In this protocol, Fixation 2 appears midway between the two leftmost numbers 5 and 9. Interpreting this off-target fixation can be problematic: A naive interpretation that simply maps fixations to the nearest target might interpret Fixation 2 as the encoding of 5 or 9, or perhaps **x** or /. However, if we take the entire protocol into account along with our knowledge of the task, we note that each of the three numbers besides 5 has a fixation directly over it and that students do not encode the **x** or the operators after solving many such problems (as is the case here). Using these observations, we conclude that the most likely interpretation of the protocol maps Fixation 2 to an encoding of the number 5. In this case, the off-target fixation can be attributed primarily to individual variability as either an overshoot of the target or possibly an intentional parafoveal encoding. (Equipment bias is a less likely possibility, because the other fixations appear directly over their targets.) This protocol illustrates how naive analyses sometimes can be inadequate and how we may need to use various types of information to form sensible interpretations.

Figure 1. **Sample protocols from (a) the equation-solving task and (b) the eye-typing task.**

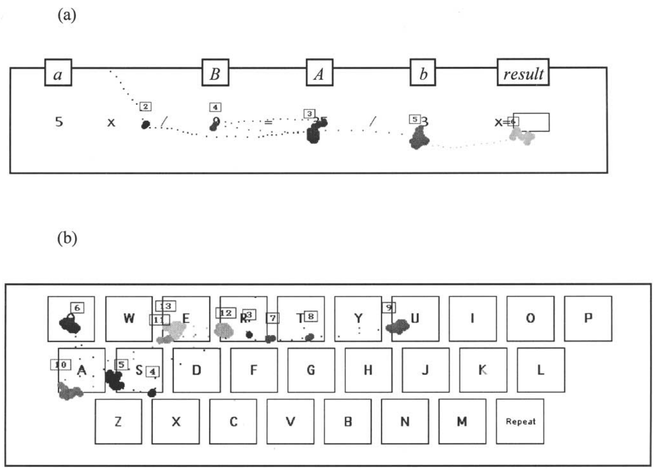


Figure 1b shows another sample protocol taken from an "eye-typing" gaze-based interface (again detailed later) in which users type words by looking at letters on an on-screen keypad. The figure includes the keypad portion of the screen along with a user's eye movements when typing the word *square.* The user first fixated the word to type (not shown) and then fixated the letters *S* (Fixations 4 and 5) and *Q* (Fixation 6). After undershooting the next target with two fixations near *T* (Fixations 7 and 8), the user fixated *U* and *A* (Fixations 9 and 10). Finally, after an incidental fixation near *E* (Fixation 11), the user finally fixated *R* and *E* (Fixations 12 and 13). This protocol nicely illustrates how variability can cause confusion in interpretation: It includes incidental fixations (e.g., 7, 8, and 11) that could be included as extraneous letters and off-center fixations that occur over the boundaries of keys (e.g., 4, 5, and 10) that could cause ambiguous interpretations. As was the case for the equation-solving example, we see that simple approaches to data analysis cannot adequately handle the complexity of typical eye-movement protocols.

1.2. Tracing Eye-Movement Protocols

In this article we explore a new class of methods that analyze eye movements by using a rigorous form of protocol analysis called tracing. Tracing is the process of mapping observed action protocols to the sequential predictions of a cognitive process model. Tracing begins with a cognitive process model capable of predicting sequences of eye movements performed during a task. This process model may be implemented in a number of ways, such as a production system (e.g., Anderson & Lebiere, 1998; Just & Carpenter, 1992; Kieras & Meyer, 1997; Laird, Newell, & Rosenbloom, 1987) or a cognitive grammar (e.g., J. B. Smith et al., 1993). For example, consider a process model for the equation-solving task described earlier (solving equations of the form $a\mathbf{x}/B = A/b$) with the following two strategies:

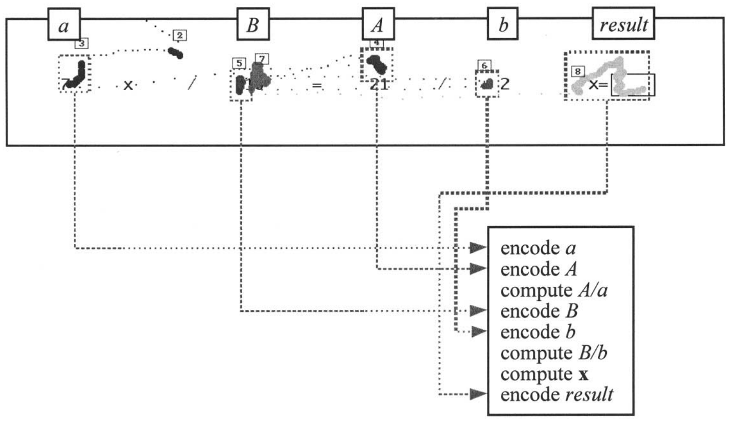
- Encode *a*, encode *B*, encode *A*, compute *A/a*, encode *b*, compute *B/b*, compute **x**.
- Encode *a*, encode *A*, compute *A/a*, encode *B*, encode *b*, compute *B/b*, compute **x**.

The first strategy encodes the four problem values from left to right, performing the three necessary computations as soon as possible. The second strategy encodes the values in pairs and from left to right and computes intermediate values as soon as possible.

Given such a process model, tracing maps an eye-movement protocol to the best fitting process model strategy. Figure 2 shows the mapping of a sample protocol to the preceding process model. (Note that the mapping, as in this case, may be partial rather than complete.) The arrows in the figure represent the trace, or interpretation, of the protocol—that is, the mapping between predicted fixations and their corresponding observed fixations in the protocol. In this case, the protocol best matches the second strategy that encodes values in pairs, and thus the trace shows the mapping from predicted to observed fixations in this strategy. Note that the mapping includes only predicted fixations (Fixations 3–6, 8); unpredicted fixations (Fixations 2 and 7) and actions that do not correspond to observed fixations (e.g., computations) are not mapped. As the example illustrates, tracing takes advantage of both global and local information in forming interpretations of protocols. Local information, namely the location and velocity of each data point, helps separate fixations and saccades and pinpoint the locations of each fixation. Global information, namely the sequence of fixations, helps sort out the assignment of fixations to their intended targets.

In this article, we describe three methods for tracing eye-movement protocols. The first method, *target tracing*, assigns fixations to targets and employs a sequence-matching algorithm to trace the target sequence. The other two methods, *fixation tracing* and *point tracing*, trace protocols by using hidden Markov models (HMMs)—powerful statistical models that have been applied successfully in fields such as speech and handwriting recognition. As we demonstrate in the application studies, these tracing methods interpret eye movements as accurately as human experts in significantly less time.

Figure 2. **Sample protocol, process model strategy, and trace from the equation-solving task.**



2. BACKGROUND: EYE MOVEMENTS AND PROTOCOL ANALYSIS

Before delving into the details of our approach, we first provide an exposition of relevant work in two important related areas: eye movements and protocol analysis. Work in eye movements has provided a foundation for understanding the physical characteristics of visual–motor control and algorithms for analyzing eye-movement data. Work in protocol analysis and tracing has explored techniques for mapping observed data to predicted actions. We now give an overview of these areas and describe how the work relates to our proposed approach.

2.1. Eye Movements

The wealth of eye-movement studies in past decades has provided solid evidence that eye movements can reveal a great deal about underlying cognitive processes (Just & Carpenter, 1984; Rayner, 1995). Researchers have used eye-movement data successfully in a variety of contexts, including menu selection (Aaltonen et al., 1998; Byrne et al., 1999), reading (Just & Carpenter, 1980, 1984; McConkie & Rayner, 1976; O'Regan, 1981; Rayner & Morris, 1990), gaze-based interfaces (Hutchinson, White, Martin, Reichert, & Frey, 1989; Jacob, 1991, 1995; Salvucci, 1999a; Salvucci & Anderson, 2000; Stampe & Reingold, 1995; Zhai, Morimoto, & Ihde, 1999), marketing (Goldberg, Zak, & Probart, 1999; Lohse, 1997), driving (McDowell & Rockwell, 1978), image scanning (Noton & Stark, 1971), air-traffic control (Lee & Anderson, 2001), television viewing (Flagg, 1978), mathematics (Hegarty, Mayer, & Green, 1992; Suppes, 1990), analogy (Salvucci & Anderson, 2001), and mental rotation (Carpenter & Just, 1978). Although some have expressed pessimism over relating eye movements and mental processes (e.g., Harris, Hainline, Abramov, Lemerise, & Camenzuli, 1988; Viviani, 1990), these numerous studies undeniably have made significant contributions to their fields through the study of eye movements.

Saccadic Eye Movements

Research in the low-level physical characteristics of eye movements is essential to understanding and analyzing eye-movement data. This article focuses on saccadic eye movements, the most common and best understood type of eye movement. Saccadic eye movements consist of two alternating events: fixations, where the eye locks on a target while the perceptual systems encodes the image and saccades, where the eye moves rapidly to the next location to fixate. Fixation durations range from approximately 100 msec to more than 1 sec, depending on the difficulty of the given task, and the distribution of fixation durations over particular domains is approximately exponential (Harris et al., 1988; Suppes, Cohen, Laddaga, Anliker, & Floyd, 1983). Saccades are (primarily) ballistic eye movements during which no information can be encoded (Kowler, 1990). Each saccade requires approximately 30 msec for 5° of visual angle and 2 msec longer for each additional degree (Fuchs, 1971). Refer to Fuchs and Kowler for detailed overviews of saccadic eye movements.

Eye Movements as Data

Eye movements can be recorded by using a wide variety of methods (see, e.g., Wolfe & Eichmann, 1997; Young & Sheena, 1975) and are extremely informative as a data source for analysis of human cognition. Perhaps the most important reason for their usefulness is that eye movements indicate, generally speaking, the focus of visual attention. Although visual attention can move independently from eye-gaze location, attention always precedes eye movements: Attention shifts to the location of the next fixation before the eye executes a saccade to that location (Hoffman, 1998). Thus, at the very least, eye movements provide clues to where the visual attention focuses at approximately the same time. Just and Carpenter (1984) even posited an "eye–mind" assumption whereby processing occurs while information is fixated and the fixation continues until processing is completed. Although this claim has been debated (Viviani, 1990), the link between eye movements and visual attention is clearly strong and provides researchers with a convenient window into a person's thought processes.

Another reason for the popularity of eye movements as data is the fine temporal grain size that they represent. Today's eye trackers can typically sample eye movements at a rate of 50 to 1000 Hz, or once every 1 to 20 msec. This fine grain size allows researchers to analyze human behavior at a level of detail unachievable with more common types of data. This push toward lower levels of detail has arisen in the broader research community; for instance, the ACT–R theory of cognition, which typically modeled actions at the 1-sec grain size not long ago (Anderson, 1993), now typically models actions at the 50-msec level (Anderson & Lebiere, 1998). Deeper understanding of cognition requires data that elucidate behavior at deeper levels, making eye movements an excellent choice as a behavioral data source.

The usefulness of eye movement data is further bolstered by the fact that eye-movement data collection (i.e., eye tracking) is mostly nonintrusive and typically does not have significant effects on observed behavior. Eye-movement studies with both tracking and nontracking conditions for the same task have shown no significant differences in performance between conditions (e.g., Salvucci, Anderson, & Douglass, 2000). In addition, eye-movement data collection generally requires no training or coaching. These benefits, in conjunction with fine temporal grain size, allow researchers to collect data easily even for tasks with short trials of a few seconds or less.

Eye-Movement Data Analysis

Considering the vast amount of research using eye movements to study behavior, relatively little research has examined formal methods of eye-movement data analysis; most of the eye-movement literature describes analysis methods informally and with few details. However, some research has investigated formal analysis methods and has demonstrated how the choice of methods can drastically impact subsequent data analysis (Karsh & Breitenbach, 1983). Of this research, the vast majority focuses on the problem of reducing raw eye-movement data to sequences of fixations and saccades (Cabiati, Pastormerlo, Schmid, & Zambarbieri, 1983; Erkelens & Vogels, 1995; Karsh & Breitenbach, 1983; Sen & Megaw, 1984; Stark & Ellis, 1981; Tole & Young, 1981; Widdel, 1984). There have been relatively few attempts to generate more sophisticated interpretations of eye movements, such as for the purposes of strategy classification (e.g., Crosby & Wesley, 1991) or gaze-based interfaces (e.g., Edwards, 1998; Goldberg & Schryver, 1995). In this article we extend this line of work to a general approach relating eye movements and cognitive processes by tracing eye-movement protocols with process models.

2.2. Protocol Analysis and Tracing

Tracing, a rigorous form of sequential protocol analysis, has become increasingly popular in various fields and under various appellations. Cognitive scientists have employed trace-based protocol analysis to develop and refine process models (Ritter & Larkin, 1994). Researchers in HCI have employed tracing, especially sequence comparison techniques, to study the fits of user models (Card et al., 1983). Builders of intelligent tutoring systems have used model tracing (Anderson et al., 1995) or tracking (Frederiksen & White, 1990) to determine the user's solution path through a student model of the domain. This section reviews work related to tracing, including more general work in protocol analysis and sequential data analysis.

Methodological Foundations

Newell and Simon (1972) provided arguably the most influential early contribution to the methodological foundations of tracing. Their work formalized the notion of the problem space and illustrated how subject protocols could help determine a participant's particular solution path through the space. Their work also demonstrated how one can test process models by mapping his or her predictions directly onto the observable actions of human participants. In the context of tasks involving cryptarithmetic and logic problems, Newell and Simon aligned think-aloud protocols with the predictions of task-specific production system models. To a lesser extent, they also discuss eye-movement data in relation with model predictions (as cited in Winikoff, 1967), sometimes to the level of tens of milliseconds. Such detailed analyses have set the tone for this article and have demonstrated the potential benefits of low-level analysis for larger sets of protocol data. Of course, there is a tradeoff between the level of detail in the analysis employed and the amount of data on which the analysis is employed, as Newell and Simon stated:

> Our investigations of human problem solving alternate between two strategies. One is to examine the behavior of an individual problem solving attempt in as much detail as possible. The aim is to determine whether the information processing theory really describes the fine structure of behavior. … The other strategy is to ask more broadly whether the theory fits the facts from a substantial range of situations, trading off breadth for depth. (p. 310)

This article is an attempt to push the envelope in both "breadth" and "depth," allowing for the examination of larger and more complex protocols at lower levels of analysis.

Another major step in the evolution of tracing came in the development of process models of HCI. Card et al. (1983) created a modeling system called

GOMS to facilitate understanding of user behavior at a fine-grained level. In one application to manuscript editing, they traced user actions with GOMS model predictions at the level of individual typing and general reading events. Using these traces, the authors evaluated various models by computing a percentage match between predicted and observed sequences with a sequence-distance metric. Their work stressed the need to use quantitative measures of a trace's goodness of fit along with traditional qualitative analyses. The tracing methods in this article also provide such quantitative metrics and, in fact, use a similar sequence-distance metric for one of the methods.

Although Newell and Simon (1972) and Card et al. (1983) emphasized the substantive aspects of their work in their particular domains, Ohlsson (1990) helped to highlight the significant methodological contributions of their work. Ohlsson formalized the methodology, which he called trace analysis, as a three-step process:

1. Construct the participant's problem space.
2. Identify the participant's solution path by using the sequential information in the protocol.
3. Hypothesize the participant's strategy by inventing problem-solving heuristics that can reproduce the participant's solution path.

Ohlsson's application of the methodology to the domain of spatial reasoning nicely illustrates how trace analysis facilitates both exploratory and confirmatory analysis in prototyping and evaluating process models.

Ritter (1992) and Ritter and Larkin (1994) examined tracing at length and specified a methodology, trace-based analysis, for testing process models' predictions through comparison with verbal and nonverbal protocols. Their formulation of trace-based protocol analysis, reminiscent of Ohlsson's (1990) trace analysis, comprised the following steps:

1. Using a process model, generate a sequence of predicted actions.
2. Compare the model predictions to empirical data by forming a mapping between the predicted action sequence and the observed action sequence.
3. Analyze the fit of the model to the data to see where the model can be improved.
4. Refine the model and iterate.

Using their Soar/MT system, Ritter and Larkin (1994) successfully developed a process model for users of a computer interface. Their article mentions eye movements only in passing, characterizing them as "overt task actions" that can be handled by the system but without explaining in detail how this might be achieved. One significant difference between their work and this article arises in the predictions of the cognitive model: Soar/MT requires that the model's sequence of predicted actions be deterministic, whereas the tracing methods in this article allow for nondeterministic action traces. In general, however, Ritter and Larkin's work is quite applicable in the context of eye movements and maps out a good methodology for developing cognitive models of visual processing.

Automated and Semiautomated Analysis

Because of the tedious nature of tracing protocols, several researchers have attempted to automate the process. For instance, Waterman and Newell (1971, 1973) developed two systems for the automated analysis of verbal reports. Their first system, Protocol Analysis System I (PAS–I), mapped participants' verbal protocols onto a problem behavior graph, which describes the trajectory of a participant's solution through a problem space (Waterman & Newell, 1971). This system ran without user input and was specifically coded for interpreting protocols in the cryptarithmetic task. The second system, PAS–II, generalized the analysis to any task and allowed the user to interactively take part in the analysis: The user "can provide answers to subproblems the system is unable to solve, correct processing errors, and even maintain control over the processing sequence" (Waterman & Newell, 1973). In a similar effort, Bhaskar and Simon's (1977) Semi-Automated Protocol Analysis system provided interactive analysis in the domain of thermodynamics.

Although these systems represented a significant attempt at automating protocol analysis, the systems, as the authors themselves admit, constituted only one component task of the larger picture of protocol analysis. A truly automated verbal protocol analysis would allow that we transform language input directly to an interpretation, which would require, among other things, a complete process theory of the generation of verbal reports. Even with important contributions in that area (e.g., Ericsson & Simon, 1980, 1984), such a theory remains out of reach. Fortunately with regard to eye movements, we know a great deal about how the cognitive and visual systems send commands to the eye and how the eye executes these commands. This crucial difference between verbal and eye-movement data makes eye movements, in this sense at least, more amenable to automated analysis.

In another attempt to automate tracing, Smith et al. (1993) employed cognitive grammars to represent cognitive strategies and parse verbal, keystroke, video, and action protocols. Using a cognitive theory of writing, they implemented three types of cognitive grammars for an expository writing task, all of which could successfully parse subject protocols into a parse tree of higher order cognitive actions, symbolizing the model's interpretation of the observed behavior. Although parsing subject protocols is an intriguing method of protocol analysis and has been effective in the writing domain, its applicability to eye-movement data seems dubious for one primary reason: The parsing must be complete and exact, with no allowance for deviations from the predicted model sequences. The authors claim that to show the validity of some grammar, "the grammar should successfully parse any protocol produced" (p. 139) in a given task. Although we can strive to achieve this ideal, real data, especially eye-movement data, seem simply too noisy for such a strict interpretation. The process of tracing eye-movement protocols must incorporate robust methods that tolerate noise and unexpected or unpredicted actions.

Several other researchers have investigated automated and semiautomated techniques that do not implement tracing per se but do highlight common goals of the previously discussed work and this article. Fisher (1991) constructed the Protocol Analyst's Workbench (PAW) for exploratory data analysis of arbitrary sequential protocol data. PAW provides flexible methods for data encoding, hypothesis testing, and data analysis, with an emphasis on cycle-based analyses such as the determination of so-called Fisher cycles. Sanderson et al. 's (1994) MacSHAPA system allows for similar types of sequential protocol analysis, including sequence comparisons, Fisher cycles, Markov transition statistics, and lag sequential analysis. Lallement (1998) used decision trees to classify data from an air-traffic controller task, showing that such machine learning techniques can provide automated analysis that is more consistent and faster than analysis by hand.

3. TRACING EYE-MOVEMENT PROTOCOLS

This section introduces three automated methods for tracing eye-movement protocols. The first method, target tracing, is an extension of an existing tracing method for generic-action protocols (Card et al., 1983; Ritter & Larkin, 1994) as applied to eye movements. Target tracing transforms eye movements to generic actions and maps observed to predicted action sequences by using a dynamic-programming, sequence-matching algorithm. The second and third methods, fixation tracing and point tracing, trace eye movements by using hidden Markov models. These methods employ probabilistic models to determine the optimal interpretation for a protocol with a given model. Fixation tracing and point tracing are designed specifically for eye-movement protocols and thus generate more accurate interpretations than the generic-action algorithm in target tracing. Together, the three tracing methods as described here represent a significant extension of previous work (Salvucci & Anderson, 1998) and bring together numerous aspects of work in eye movements and protocol analysis outlined in the previous section.

Interpretation of eye-movement protocols encompasses a wide range of potential methods and issues. To restrict the scope of this work to a manageable subset, we make one important assumption in this article: The visual screen for which eye-movement data are collected is relatively static during particular subtasks. We should stress that this assumption does not preclude application of the methodology to dynamic environments; tasks in these dynamic environments often can be broken into subtasks with relatively static displays, even though displays between subtasks may be quite different.

3.1. Tracing Setup

The tracing methods require the specification of two components: a set of target areas that designate relevant regions of interest, and a process model that generates sequences of predicted actions. Target areas, or "regions of interest," indicate relevant units of information that people are likely to encode. For simplicity, we assume that targets areas can be specified as rectangular regions that enclose the visual space in which the target may be encoded; rectangular regions nicely approximate common visual objects such as text, icons, and menus. In addition, because this article is restricted to primarily static task screens, we can assume that the target areas remain fixed throughout a single trial (i.e., the range of one run of the process model).

The process model provides tracing with a specification of the set of possible predicted action sequences. For the purposes of this article, the process model is a *regular grammar* (Sudkamp, 1988) that can generate all possible sequences of fixations on targets. The regular grammar comprises a set of rules of the form **subgoal** → {fixation} * **{subgoal}**. The first subgoal represents a subgoal that must be achieved through some sequence of fixations. When the rule executes, or fires, this subgoal produces zero, one, or more fixations on various target areas and optionally specifies a new subgoal to achieve. When multiple rules appear for the same subgoal, the grammar specifies the probability with which each rule fires.

Let us consider a sample process model for the equation-solving domain discussed earlier. Assume that students solve equations by using one of four strategies, shown as sequences of visual, cognitive, and typing actions in Figure 3. The first two strategies represent two methods of solving the problem: one in which the values are encoded left to right, and one in which the values are encoded in pairs. The second two strategies are identical to the first two except that they do not encode (check) the result after typing. Each strategy has a particular probability of occurring based on a .60 probability of reading left to right and .40 of reading in pairs, and a .75 probability of checking the result and .25 of not checking. Note that we set these probabilities arbitrarily for illustration; in real applications, we generally set them to uniform probabilities

Figure 3. **Sample equation-solving strategies.**

Number	Strategy	Probability
1	encode *a*, encode *B*, encode *A*, encode *b*, compute *A/a*, compute *B/b*, compute **x**, type **x**, encode *result*	.60 × .75
2	encode *a*, encode *A*, encode *B*, encode *b*, compute *A/a*, compute *B/b*, compute **x**, type **x**, encode *result*	.40 × .75
3	encode *a*, encode *B*, encode *A*, encode *b*, compute *A/a*, compute *B/b*, compute **x**, type **x**	.60 × .25
4	encode *a*, encode *A*, encode *B*, encode *b*, compute *A/a*, compute *B/b*, compute **x**, type **x**	.40 × .25

(i.e., all rules are equally likely) unless we have evidence to the contrary, which may arise from exploratory analysis or task requirements (as we see later in the case studies).

We can represent the strategies in Figure 3 as the model grammar shown in Figure 4. This model grammar contains two subgoals: **compute-result** and **type-result**. The **compute-result** subgoal can be executed in one of two ways, represented by the first two rules. The first rule, which executes with probability .60, encodes the values left to right; the second rule, with probability .40, encodes the values in pairs. Both rules then execute the **type-result** subgoal. The **type-result** subgoal either generates a fixation on the result by using the third rule with probability .75 or generates no fixation by using the fourth rule with probability .25. In either case, execution completes due to the absence of a right-hand subgoal for both type-result rules.

3.2. Tracing Methods

The following proposed tracing methods require three inputs: a process model grammar, a set of target areas as described previously, and an eye-movement protocol comprising a sequence of <*x*,*y*> point-of-regard locations collected by an eye tracker. The tracing methods generate two outputs: the trace (i.e., the mapping from the protocol to the sequential predictions of the process model) and a mismatch score (i.e., a goodness-of-fit value that describes how well the protocol fits the model).

Target Tracing

Target tracing is a two-stage process that extends an existing tracing algorithm to eye-movement protocols. Card et al. (1983), Ritter (1992), and Ritter

Figure 4. **Process model grammar for the model strategies in Figure 3.**

Number	Rule	Probability
1	**compute-result** → *a B A b* **type-result**	.60
2	**compute-result** → *a A B b* **type-result**	.40
3	**type-result** → *result*	.75
4	**type-result** →	.25

and Larkin (1994) developed tracing algorithms for generic actions based on a common sequence-matching algorithm (Kruskal, 1983). Their algorithms find the optimal match between two sequences that minimizes the string-edit distance between the sequences—that is, the number of insertions, deletions, and substitutions needed to transform one sequence to the other. This tracing method, which they applied to user actions and verbal protocols, showed that tracing by sequence matching is highly efficient and generates accurate and useful interpretations for data analysis. When combined with target identification, sequence matching can generate the same useful interpretations for eye-movement data.

Fixation Identification and Assignment. The first stage of target tracing transforms a raw eye-movement protocol to a sequence of intended targets by means of *fixation identification* and *assignment*. First, fixation identification separates fixation points from saccade points in the raw protocol and produces a sequence of fixation locations. For this process, target tracing uses a straightforward velocity-based algorithm described in previous work (Salvucci & Anderson, 1998). Next, fixation assignment takes the resulting fixations and maps each fixation to its intended target; that is, it matches each fixation with the visual target assumed to be encoded during that fixation. For this process, target tracing simply assigns each fixation to the nearest target; for example, the protocol in Figure 2 would produce the target sequence [*B a A B b B result*]. This method of fixation assignment implicitly assumes that each fixation is intended to encode the visual target nearest to it. In doing so, it converts fixations to generic actions as required by the sequence-matching algorithm: A fixation near a target becomes the action of encoding the information in that target.

Tracing. The second stage of target tracing matches the observed target sequence resulting from the first stage to a predicted sequence of targets generated by the process model. Target matching first generates all possible target sequences from the process model grammar; for instance, for the grammar in Figure 4, it generates the sequences [*a B A b result*], [*a A B b re-*

sult], [*a B A b*], and [*a A B b*]. Next, it uses the sequence-matching algorithm to determine the best match between the observed target sequence and each predicted target sequence generated from the model. The sequence-matching algorithm compares the observed and predicted sequences with respect to a sequence-distance metric: the number of insertions, deletions, and substitutions needed to transform one sequence to the other (Kruskal, 1983). The predicted sequence with the smallest sequence distance from the observed sequence is chosen as the best-matching sequence. The resulting trace defines a mapping between elements of the observed and predicted sequences. This mapping may contain elements from the observed sequence that are not predicted (i.e., deleted) and elements from the predicted sequence that are not observed (i.e., insertions). In addition, the process generates a goodness-of-fit value as the mismatch between sequences—namely, the number of insertions, deletions, and substitutions present in the mapping.

Figure 5 shows a sample trace for the protocol in Figure 2 and the model in Figure 4. On the left, the figure shows the sequence of observed targets in the protocol. On the right, the figure shows the best-matching sequence of predicted targets from the four possible target sequences described previously. In addition to specifying the most likely strategy, the trace shows that the model does not predict two fixations: the first fixation on *B*, which is likely a missed saccade to the first element of the equation, and the final fixation on *B*, which seems to represent a review of previously encoded information. We require two deletions to transform the observed sequence to the predicted sequence, so the mismatch (i.e., sequence distance) between the two sequences is 2.

Two issues arise when applying the sequence-matching algorithm in target tracing. First, the model grammar used in tracing may be circular in that certain subgoals may repeat indefinitely. In such a case, target matching cannot generate all possible grammar sequences because there exist an infinite number. Thus, when generating grammar sequences, target matching must ensure that circularity is cut off at some convenient point. Second, ties sometimes occur in sequence matching when two or more model sequences have the same sequence distance from the observed sequence. To break ties in these cases, target tracing (somewhat arbitrarily) favors mapping in which matches occur earlier rather than later in the sequences. Alternatively, target tracing could break ties by favoring more likely sequences as determined by the rule probabilities in the generating grammar.

Evaluation. Target tracing is fairly accurate and efficient, especially for smaller, noncircular process models with few enumerated strategies. However, because of the assignment of fixations to targets in the first stage, target tracing sometimes can have difficulty tracing noisy protocols. Figure

Figure 5. **Trace of the Figure 2 protocol using target tracing and the process model grammar in Figure 4.**

Observed Targets	Mapping	Predicted Targets
B		
a	→	*a*
A	→	*A*
B	→	*B*
b	→	*b*
B		
result	→	*result*

1a illustrates such a protocol. When we look at the protocol as a whole, it closely corresponds to the strategy in which values are encoded in pairs, so Fixation 2 should be attributed to the target *a*. However, target tracing would assign Fixation 2 to its closest target *B*, resulting in an observed target sequence [*B A B b*] after the first stage. This observed sequence matches equally well with two predicted target sequences—namely, [*a B A b*] and [*a A B b*]. The crucial problem here is the loss of information between stages: Once the first stage assigns a fixation to a target, the second stage must rely only on the assigned target for tracing and cannot access information concerning where the fixation may have fallen. A second problem with target tracing involves the lack of ability to use rule probabilities in tracing. Although we could imagine heuristics that may help—for example, multiplying the mismatch score by (1 – strategy probability)—target tracing has no parsimonious and straightforward way of incorporating strategy probabilities.

Fixation Tracing

Fixation tracing, like target tracing, is a two-stage process that comprises identification and tracing. However, instead of assuming that fixations map to the nearest target, fixation tracing allows the model to influence the mapping by tracing fixations by using HMMs. HMMs are powerful probabilistic models that have been applied extensively in the development of speech and handwriting recognition systems (e.g., Lowerre & Reddy, 1980). Just as these systems infer intended speech or writing from observed input, fixation tracing infers intended encoding and fixation patterns from observed eye movements. In previous work, researchers have used simple Markov models in analyses of transition probabilities from one fixation target area to another (e.g., Stark & Ellis, 1981; Suppes, 1990). These models shed light on the sequential nature of eye-movement protocols but ignore more global information by assuming that transitions do not depend on the prior sequence of fixations. A few researchers have used HMMs to represent velocity distributions in smooth eye movements (Kowler, Martins, & Pavel, 1984) and to model explicit foveal sequences (Rimey & Brown, 1991). We extend this work by developing a more rigorous approach that takes advantage of the full expressive power of HMMs.

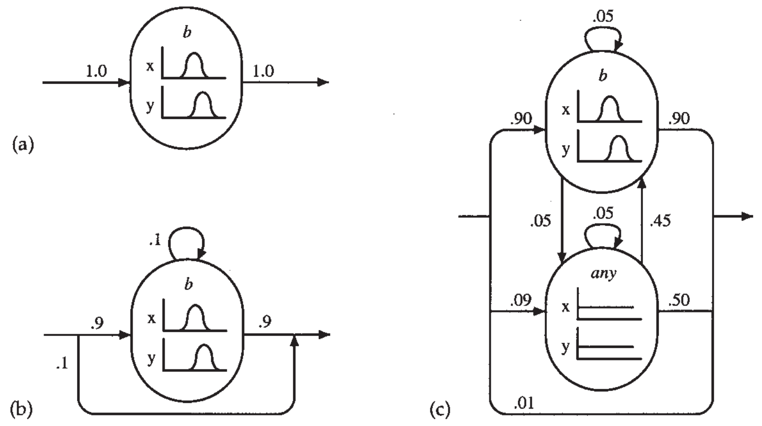
Fixation Identification. The first stage of fixation tracing transforms a raw eye-movement protocol into a sequence of fixations by means of fixation identification. Like target tracing, fixation tracing uses a velocity-based algorithm (Salvucci & Anderson, 1998) that produces a sequence of fixations represented as locations $<x,y>$. However, unlike target tracing, fixation tracing makes no assumptions about the intended targets for each fixation in the first stage; this assignment occurs as part of the tracing process in the second stage.

Tracing. The second stage of fixation tracing decodes, or interprets, the identified fixation sequence with respect to a *tracer HMM*—an HMM that probabilistically represents the process model. We first describe how to construct the tracer HMM from the specified process model and target areas. We then describe how fixation tracing traces a protocol by determining the alignment of observed fixations and tracer HMM states that maximizes the probability of the fixation sequence with respect to the HMM.

We construct the tracer HMM in two steps.[1] Because the tracer HMM decodes (interprets) a fixation sequence, each state in the HMM must represent a fixation. Thus, we first construct *fixation HMMs* for each target area that represent the likely x and y locations for fixations intended to encode that target. Figure 6a shows a sample fixation HMM for the equation-solving target area b. This fixation HMM contains only one state that includes probability distributions for expected x and y coordinates for fixations intended to encode b; the distributions center around the coordinate centers of the target area, and their standard deviations lie near the boundaries of the target area. The transition probabilities for the HMM force the state to be traversed exactly once. Although this HMM is one possible fixation HMM, there are any number of other possibilities, such as those included in Figure 6 that allow for repeating and skipping of fixations (Figure 6b) and unpredicted intervening fixations on any target with uniform x and y distributions (Figure 6c). Overall, the fixation HMM should represent expected fixations around a particular target given

1. In practical use, the tracer HMM need only be constructed once and can be stored for subsequent traces.

Figure 6. **Sample fixation HMMs for fixation tracing.**

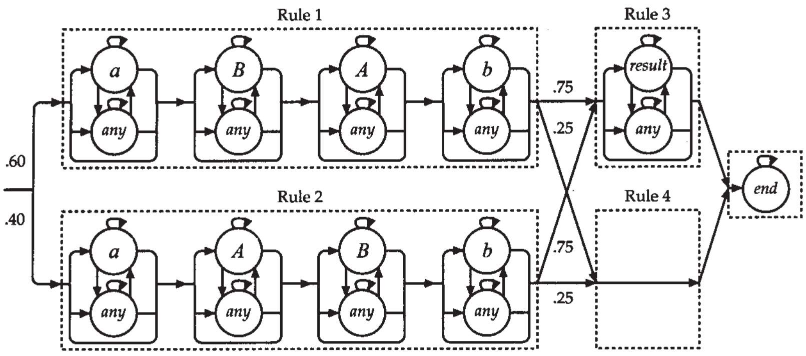


that the person intends to look at that target. The choice of fixation HMM can depend on the given domain, particularly whether the model predicts fixations or gazes (consecutive fixations on the same target); for instance, the HMM in Figure 6a represents only a single fixation, whereas those in Figures 6b and 6c can include multiple fixations that make up a gaze.

Once the fixation HMMs have been defined, fixation decoding uses the HMMs and the process model grammar to construct the tracer HMM, as shown in Figure 7 for the grammar in Figure 4. Fixation decoding first converts each grammar rule to a *rule HMM* that includes serially linked fixation HMMs for each predicted target in the rule. For instance, Figure 7 contains a rule HMM for Rule 1 in Figure 4 in which the fixation HMMs (from Figure 6c) for *a*, *B*, *A*, and *b* are linked together serially; it also contains a null rule HMM for Rule 4. Next, the rule HMMs are linked together according to the nonterminals (subgoals) present in the rules: Each rule HMM with a right-hand-side nonterminal (i.e., rule HMMs 1 and 2) is linked to each other rule HMM with that nonterminal in the left-hand side (i.e., rule HMMs 3 and 4); each rule HMM with no right-hand-side nonterminal (i.e., rule HMMs 3 and 4) is linked to a *terminal HMM* that represents some special end observation. The probabilities of the transition links are dictated by the probabilities of the rules to which the transitions go. Note that because the fourth rule HMM is null, the transitions link the first and second rule HMMs directly to the terminal HMM.

By using the tracer HMM, fixation tracing traces a given protocol by determining the best alignment of observed fixations to tracer HMM states. This process, called *HMM decoding*, finds the mapping from fixations to states that maximizes the probability of the fixation sequence given the tracer HMM.

Figure 7. **Tracer HMM for fixation tracing and the process model in Figure 4.**



(See Rabiner, 1989, for a detailed description of HMM decoding.) Decoding produces a trace of the protocol in that observed fixations (and thus raw data points) are mapped to their intended targets, as embodied in the tracer HMM states. The mismatch score produced by fixation tracing is defined as follows:

$$mismatch = \frac{-\log P}{L} \tag{1}$$

Here P is the probability of the observed fixations given the tracer HMM as provided by the HMM decoding process—that is, the probability of the data given the model—and L is the length of the observed fixation sequence. Thus, the equation represents the mismatch score as the average log probability per fixation, with a perfect match between model and data producing a mismatch score of 0. In general, the actual magnitude of the mismatch score is less important than its magnitude relative to mismatch scores for other models, allowing us to determine which model better fits the given data. The case studies, particularly for the reading domain, illustrate this model comparison process.

Evaluation. Fixation tracing, because of the incorporation of fixation assignment in the tracing process, allows for more flexible and accurate interpretations than target tracing. In the Figure 1 protocol, although target tracing assigns Fixation 2 to target B before tracing, fixation tracing uses the fixation's location and the overall sequence of fixations to attribute the fixation accurately to the target a. By using HMMs for tracing, fixation tracing is able to employ higher level global information, namely the sequence of fixations as predicted by the process model, to influence the interpretation of lower level aspects of the protocol, namely the assignment of fixations to targets. It also uses grammar rule probabilities, unlike target tracing. Nevertheless, the two-stage process of fixation tracing still results in some loss of information between stages, namely the determination of where fixations occur in the protocol. Figure 8a shows a protocol in which the fixation identification stage has failed to identify an evident fixation on target *b* (at the direction reversal on the right side of the protocol). This has happened because of the fixation's short duration and the effects of time-averaged sampling. We know there is a fixation on *b* by considering the larger protocol, but the cognitive model in the second stage cannot influence the determination of what is and is not a fixation.

Point Tracing

Point tracing identifies fixations and traces the protocol in a single stage, thus avoiding the loss of information between stages suffered by target and fixation tracing. The point-tracing algorithm relates very closely to the fixation-tracing algorithm: It constructs an HMM from the process model and decodes the protocol with this HMM. However, rather than decoding a sequence of fixation locations, point tracing decodes the raw sequence of eye-movement data. The one-stage process allows the process model to influence both the identification of fixations and the assignment of fixations to intended targets.

Tracing. Point tracing begins with the construction of a tracer HMM used to decode the given protocol. This process is, for the most part, identical to the construction of the tracer HMM in fixation tracing: It generates fixation HMMs for each possible target, builds rule HMMs for each rule in the model grammar, and links the rule HMMs as described by the model grammar. However, whereas the fixation HMMs in fixation tracing represent expected fixation locations, the fixation HMMs in point tracing represent expected raw eye-movement data. Figure 9 shows one possible fixation HMM for target *b* analogous to that for fixation tracing in Figure 6a. The first state represents the saccade points present before the fixation; the state includes uniform *x* and *y* distributions and a *v* velocity distribution centered around high velocities. The velocity distributions can be estimated from existing protocol data (Salvucci & Anderson, 1998). The second state represents fixation points with *x* and *y* distributions centered around the *b* target and a *v* distribution centered around low velocities. Of course, we could also design alternative fixation HMMs analogous to those in Figures 6b and 6c.

By using the constructed tracer HMM, point tracing decodes the given protocol and determines the optimal protocol trace. First, it augments the $<x,y>$

Figure 8. **Illustrative protocol traced with (a) fixation tracing and (b) point tracing.**

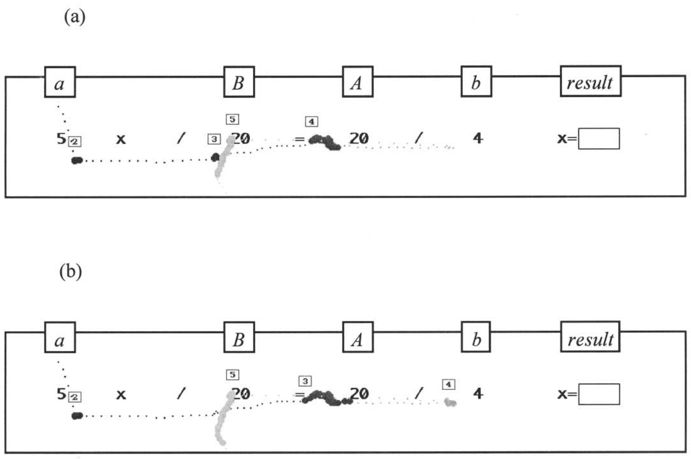
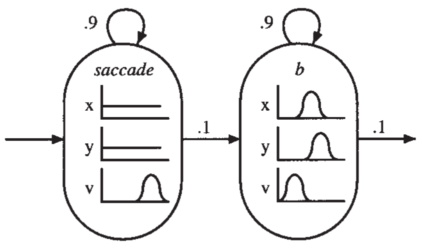


Figure 9. **Sample fixation HMM for point tracing.**



eye-movement protocol with point-to-point velocities to generate sequences of $<x,y,v>$ tuples. Second, it decodes these tuples with the tracer HMM to find the optimal trace, or mapping, from observed to predicted fixations. The trace provides a mapping between data points and tracer HMM states, which can be used to collapse the raw protocol into fixations or gazes. The mismatch score for point tracing can be defined exactly as that for fixation tracing, as shown in Equation 1. Note that the mismatch scores between fixation and point tracing are not directly comparable because the mismatch for point tracing represents the negative log of the average probability per raw data point rather than fixation.

Evaluation. Because point-tracing HMMs can be much larger than fixation-tracing HMMs and must decode much more data, point tracing is typically much less efficient than fixation tracing. This loss in efficiency is counterbalanced somewhat by gains in information from tracing a raw protocol directly: Point tracing allows the process model to influence both fixation identification and assignment. For instance, Figure 8b shows the same protocol as Figure 8a traced with point tracing; here the model determines that the bend in data points nearest the *b* target must correspond to a fixation, even though the point velocities fail to indicate a fixation. Unfortunately, point tracing occasionally fails to identify incidental or extraneous fixations as we would expect: Rather than assigning them to some *any* state, point tracing tends to identify points for incidental fixations as saccade points rather than fixation points—for instance, Fixation 3 in Figure 8a that disappears in Figure 8b. This problem sometimes causes difficulty when analysis requires the determination of incidental fixations.

3.3. Issues

Fixations represent a fairly natural grain size for typical analysis and modeling of eye movements. However, for some applications or domains, many researchers have used *gazes* as the basic units of eye-movement analysis (e.g., Just & Carpenter, 1984; Rayner & Morris, 1990). A gaze typically is defined as a sequence of one or more consecutive fixations on the same target area. Gazes allow researchers to focus their analyses on eye movements between visual targets and to ignore eye movements within targets. All three tracing methods allow a user to trace eye movements at the level of either fixations or gazes. The decision to use fixations or gazes impacts the specification of the process model and the tracing algorithms themselves. First, the process model should predict scanning behavior at the level of the desired analysis: If we are interested in tracing gazes in a physics task, our model should predict individual gazes; if we are interested in tracing fixations in a reading task, our model should predict fixations. Second, the tracing algorithms as described generally assume analysis at the fixation level. At the gaze level, the algorithms require slight modification. For target tracing, target identification must collapse consecutive, repeated fixations on target areas into single gazes before passing the target sequence to target tracing. For fixation and point tracing, the fixation HMMs should allow for repeated fixations on their respective targets to model the multiple fixations that can compose a gaze. The case studies presented later illustrate analysis at both the fixation and gaze levels.

The tracing methods use the locations of fixations and the sequence in which they occur. Additional information present in an eye-movement protocol—namely, fixation durations and onset times—potentially can facilitate interpretation of the protocol. Fixation durations can indicate the amount of cognitive processing dedicated to particular fixations or gazes. This information could help determine whether a fixation is intended or incidental, or whether computation takes place during a fixation; for instance, fixations in the equation-solving task during which the person performs a mathematical operation are typically longer than other fixations. Fixation onset times can help us to understand the interleaving of encoding and other actions, such as typing or mouse movements. By comparing the onset times of fixations with the times of these other types of actions, tracing methods potentially could distinguish between various strategies with similar eye movements but dissimilar behavior in these actions.

Another issue is that some process models of visual attention may not predict the exact target of every fixation but rather may posit that a fixation lands in some area on "any" target in that area. For instance, in the equation-solving domain, students sometimes move their eyes down onto the equation from the start target area to orient themselves as to where elements of the equation lie; the fixation is not intended for a particular target value but more so for the entire equation as a whole. Although such fixations occasionally could be modeled with larger target areas, it is often convenient to use a special *any* target area that captures fixations on any target. All three tracing methods can handle the *any* target area: Target tracing requires a slight modification of the sequence-matching algorithm, and fixation and point tracing require fixation HMM states in which the x and y distributions area fairly uniform (flat) and thus do not center over a particular target.

3.4. Summary

Figure 10 provides a summary of the tracing methods with respect to model influences and complexity. The model influences illustrate the power of the various methods in terms of what aspects of the tracing process are influenced by the specified process model. The complexities of the different methods depend on four factors: the length of the raw protocol T, the number of fixations F, the number of enumerated strategies M, and the number of tracer HMM states N. Not surprisingly, all the methods depend on the protocol length T. The dependency of target tracing on M is related closely to the dependency of fixation and point tracing on N. For simpler models, M and N may be near equal, but N may be significantly smaller than M for more complex, hierarchical process models. Also, the factor of $O(N^2)$ in fixation and point tracing typically behaves more like $O(N)$ for process models that generate sparse tracer HMMs. Note that the complexities for fixation and point tracing do not include the time needed for tracer HMM construction, which typically is performed once per process model and reused for all traces with the model.

The tracing methods, like the fixation identification methods, are difficult to compare in the absence of an application domain. The next section provides detailed quantitative and qualitative evaluations and comparisons of the

Figure 10. **Summary of the interpretation methods.**

Method	Model Influences	Complexity[a]
Target tracing	Trace	$O(T + MF^2)$
Fixation tracing	Fixation assignment, trace	$O(T + N^2F)$
Point tracing	Fixation identification, fixation assignment, trace	$O(N^2T)$

[a]The complexity parameters are interpreted as follows: T is the length of the raw protocol, F is the number of fixations, M is the number of enumerated strategies for the process model, and N is the number of states in the tracer hidden Markov models.

tracing methods in the context of application domains. The subsequent general discussion includes an overview of these results and a number of recommendations for choosing an appropriate tracing method for future applications.

4. CASE STUDIES

We have applied the tracing techniques and methodology to three complementary domains: equation solving, reading, and gaze-based interfaces. These applications serve two purposes. First and foremost, the applications help to evaluate the accuracy and speed of the tracing methods. As yet we have only described how the methods form interpretations of eye-movement protocols; in this section we show that the methods indeed form accurate interpretations and can produce them significantly faster than human experts. Second, the applications demonstrate the uses and benefits of tracing for the offline study of user behavior and the online inference of user intentions. Although the chosen applications involve relatively basic tasks, these tasks nicely allow for rigorous testing and evaluation of the tracing methods and demonstrate numerous ways in which the methods could be used in larger scale applications.

To apply the tracing methods to the studied domains, we have developed a testbed system called EyeTracer[2] that embodies the three tracing methods described in the previous section as well as a number of other algorithms used by and with the tracing methods. EyeTracer facilitates both exploratory and confirmatory analysis of eye-movement protocols. For exploratory analysis, the system can replay protocols in real time or any arbitrary speed, which provides an extremely useful tool for an initial understanding of behavior (Smith

2. EyeTracer is publicly available on the World Wide Web at http://www.cbr.com/~dario/EyeTracer

et al., 1993). EyeTracer also can display static plots of an eye-movement protocol with various aids for analysis; for instance, EyeTracer can plot points at sizes that correspond to their velocities, mark and number fixations in the plot, and mark when other (non-eye-movement) events take place (e.g., keystrokes). Such displays incorporate little to no bias in the interpretation of the protocols, thus allowing the user to view essentially unadulterated data. In addition, given the postprocessed protocols, EyeTracer can be customized to perform any type of exploratory analysis in which the user may be interested, such as fixations counts and durations, Fisher cycles, or Markov transition matrices. For confirmatory analysis, EyeTracer can display eye-movement protocols highlighted according to a trace with respect to a cognitive model; for instance, the system can highlight predicted fixations in blue and unpredicted fixations in red. The functionality of EyeTracer thus enables its users to better understand behavior, construct prototype cognitive models, and subsequently refine the models based on the match and mismatch between predicted and observed behavior.

4.1. Equation Solving

We begin with an application of the tracing methods to equation solving (Salvucci & Anderson, 1998). The equation-solving task involves solving equations of the form $a\mathbf{x}/B = A/b$ by computing the result $\mathbf{x} = (A/a)(B/b)$. Our study of equation solving emphasizes one use of tracing in particular: the classification of protocols into given model strategies. The study evaluates both the accuracy and speed of the tracing methods in classifying protocols.

The equation-solving study tests the methods' accuracy and speed in two ways. The first way involves comparing their interpretations to the "correct" interpretations as generated in an instructed-strategy paradigm. The instructed-strategy paradigm asks students to solve problems by using a particular strategy; this instructed strategy specifies the exact sequence of steps to perform in solving the problem, including visual accesses (i.e., look at and encode this value), computational cognitive processes (e.g., divide A by a), and motor actions (e.g., type the result). The instruction provides a "correct" answer for interpreting the protocol—that is, we knew what strategy a student should be using for a given protocol. By asking each tracing method to interpret the protocols as one of the instructed strategies, we can compare their interpretations to the correct strategies and determine how accurately they perform.

The second way of testing the tracing methods entails comparing their performance to that of expert human coders. Regardless of the power of the tracing methods, it is reasonable to assume that they may not interpret all protocols accurately because of user variability, data noise, or both. However, because the tracing methods are intended to automate the analysis task typically performed by humans, we are primarily interested in how accurately they interpret compared with human coders. This comparison controls for protocols in the data set that simply cannot be interpreted because of the student failing to perform the instructed strategy or because of extreme data noise.

Experiment

In the experiment, students solved equations in five 1-hr sessions. In the first session, students could solve the equations however they wished. In each of the next four sessions, students were instructed to solve the equations by using one of the following strategies: read values left to right [*a B A b*], read values left to right in pairs [*a A B b*], read values right to left [*b A B a*], and read values right to left in pairs [*b B A a*]. In all strategies, intermediate results (*A/a* and *B/b*) were computed as soon as possible. Note that students used only one strategy in a particular session to avoid confusion between strategies. Our analysis includes the data from four students for a total of 369 trial protocols.

For this experiment and all others described in this article, we collected eye-movement data by using an ISCAN, Inc.® (Burlington, MA) RK726/RK520 eye tracker with a Polhemus FASTRAK™ head tracker. The ISCAN eye tracker uses the pupil-center and corneal-reflection points to estimate the angle of the left eye's gaze. The head tracker determines the position of the head and eye with respect to the visual field. The system collects eye and head information to produce a stream of on-screen point-of-regard locations. Data were sampled at a rate of 120 Hz. Participants were seated approximately 30 in. from the computer display screen. Characters in the equation-solving stimuli were approximately ¼ in. in height (0.48° of visual angle) with approximately 1 in. (1.91° of visual angle) between them.

Protocol Analysis

To analyze the protocols by using the tracing methods, we require a process model grammar that describes student behavior in the task. Fortunately, the task specifies the model for us: The model is a straightforward translation of the instructed strategies, as shown in Figure 11. The grammar incorporates two additional features not included in the original strategies. First, Rule 1 (the rule that fires first) produces a fixation on the start area, as students were required to do. Second, Rules 6 through 8 fixate any item or items as review (Rule 6), fixate the result area and terminate (Rule 7), or simply terminate (Rule 8). Because students were instructed to perform the strategies with equal frequency, the rule probabilities are distributed evenly among rules for each subgoal.

Figure 11. **Process model grammar for the constrained equation-solving task.**

Number	Rule			Probability
1	**start-trial**	$\rightarrow$	*start* **compute-result**	1
2	**compute-result**	$\rightarrow$	*a B A b* **type-result**	1/4
3	**compute-result**	$\rightarrow$	*a A B b* **type-result**	1/4
4	**compute-result**	$\rightarrow$	*b A B a* **type-result**	1/4
5	**compute-result**	$\rightarrow$	*b B A a* **type-result**	1/4
6	**type-result**	$\rightarrow$	*any* **type-result**	1/3
7	**type-result**	$\rightarrow$	*result*	1/3
8	**type-result**	$\rightarrow$		1/3

Classifying Strategies

We first examine how the tracing methods classify protocols as one of the given instructed strategies. To compare the methods' performance to that of human coders, we extracted a test set of the student protocols and had two coders classify each protocol as one of the instructed strategies. The test set comprised the last two protocols from each strategy and each participant ($2 \times 4 \times 4 = 32$ protocols total). The coders viewed the protocols as printed eye-movement displays, with numbered fixations to clarify their temporal sequencing. Both coders (a professor and graduate student in cognitive psychology) were highly experienced in studying such printed displays and in developing cognitive process models. Figure 12 shows the percentage agreement between the tracing methods, human coders, and "correct" interpretations for the 32-protocol test set. The tracing methods achieved an accuracy of 87.5% to 93.7%, whereas the human coders achieved an accuracy of 78.1% to 90.6%. Fixation and point tracing achieved the highest accuracy of 93.7%. Interestingly, for five of six cases, the agreement between the tracing methods and the human coders was as great as or greater than the agreement between the human coders. These results demonstrate that, on average, the tracing algorithms interpreted the protocols at least as accurately as the human coders. Not surprisingly, the tracing methods interpreted the protocols in significantly less time: As Figure 12 shows,[3] the tracing methods averaged 1 sec or less per protocol, whereas the human coders averaged approximately 1 min per protocol.

As a better comparison among the tracing methods themselves, we also can compare the tracing methods' interpretations for the entire protocol data set.

3. All reported times for the tracing methods were collected on a 200 MHz Power Macintosh, and times for fixation and point do not include the (small) one-time cost of building the tracer HMM.

Figure 12. **Percentage agreement and interpretation time per protocol for the tracing methods, human coders, and "correct" interpretations in the constrained equation-solving task.**

	Target	Fixation	Point	Human 1	Human 2	Correct
Target	—	84.4	84.4	81.2	90.6	87.5
Fixation		—	93.7	81.2	84.4	93.7
Point			—	78.1	90.6	93.7
Human 1				—	81.2	78.1
Human 2					—	90.6
Time (sec)	.038	.045	1.01	67.5	60.0	

Target tracing interpreted 91.9% of the protocols correctly, fixation tracing 93.2%, and point tracing 94.3%. These results reflect the fact that as the methods use more information for tracing (see Figure 10), their accuracy increases accordingly. We should note that we cannot expect 100% accuracy from any method: The inherent noise and human variability in certain protocols simply preclude correct interpretation, as attested by the accuracy of the human coders. Overall, the results of the constrained study demonstrate that the tracing methods do indeed form reliable and accurate interpretations of eye-movement protocols.

4.2. Reading

The second application examines reading, a domain in which eye movements have received a great deal of attention in both empirical work (e.g., Just & Carpenter, 1984; McConkie & Rayner, 1976; O'Regan, 1981) and modeling work (e.g., Just & Carpenter, 1980; Legge, Klitz, & Tjan, 1997; Reichle, Pollatsek, Fisher, & Rayner, 1998). The study and modeling of eye movements in reading typically have addressed two types of information in the data: temporal information in the form of fixation durations (and related measures) and spatial information in the form of fixation or gaze locations. Although these types of information provide a good picture of reading behavior, research in reading has paid less attention to another important aspect of eye movements: sequential information. It may seem surprising at first that sequential information is an important component of eye movements in reading; one might have the impression that the eye simply moves left to right with few interesting deviations. On the contrary, this sequential information can be quite complex. When considering only left-to-right eye movements at the word level, the eye sometimes refixates words or skips some words entirely. These possibilities produce interesting transitions where, from a given word, the eye may move to the same word, the next word, or any subsequent word with some likelihood. Sequential information nicely complements temporal and spatial information and provides a fuller picture of human reading behavior.

The reading study emphasizes two uses of tracing: quantitative evaluation of process models and cleaning up noisy eye-movement data for subsequent data analysis. The first use takes advantage of the mismatch score (i.e., the goodness-of-fit measure) provided by tracing and compares two existing process models of reading: E-Z Readers 3 and 5 (Reichle et al., 1998). E-Z Readers 3 and 5 represent two of five E-Z Reader models developed to predict eye movements in reading at the word level. Reichle et al. fit all five E-Z Reader models to several aspects of a previously collected data set (Schilling, Rayner, & Chumbley, 1998), specifically to fixation durations and probabilities on words of different frequencies. E-Z Readers 3 and 5 were found to provide the best fits with respect to these measures. The essential difference between the two models was that E-Z Reader 5 included an assumption that words farther in the periphery are more difficult to process. However, because both models provided equally good fits to these measures, Reichle et al. could rely only on qualitative examination to discern which model was indeed the more accurate model of reading behavior. We demonstrate how tracing facilitates quantitative comparison of the two models.

The second use of tracing emphasized in the reading study is "cleaning up" noisy eye-movement data to facilitate later analyses. The reading experiment described here is a replication of a previous experiment, which we term the "SRC" experiment after the authors (Schilling et al., 1998). Although the original SRC data were available for use, they included only duration, word position, and letter position of each fixation; they did not include the millisecond-by-millisecond raw data required for the tracing algorithms. The experiment in this study, which we term the "New" experiment, gathers these raw data and, at the same time, replicates the experimental results with a different eye tracker and subject population. However, the eye tracker used in the New experiment was somewhat less accurate than that used in the SRC experiment (accuracies of ½ to 1° for New vs. $\frac{1}{6}$° for SRC) and had a worse temporal resolution (sampling at 120 Hz for New vs. 400 Hz for SRC). Nevertheless, as we later see, tracing helps make sense of the less accurate data where naive algorithms produce inconclusive results.

The reading tasks and the equation-solving tasks have an important difference that concerns the possible strategies in the two tasks. For equation solving, the number of potential strategies that students could employ is relatively low, allowing us to enumerate every strategy fully in the model grammar. For reading, the number of potential strategies—that is, possible sequences of word fixations—can be extremely high; for instance, for left-to-right strategies on a 10-word sentence in which each word may be fixated or skipped, there may be up to $2^{10} = 1{,}024$ strategies. It is generally infeasible to enumerate this many strategies fully in a model grammar. Thus, we must reduce the full grammar to a more feasible grammar that, although it contains less information, incorporates as much of the important aspects of the original grammar as possible; for instance, we use a “first-order” grammar in which one state represents each word and the transitions between states represents eye movements between words. The reading domain and the eye-typing domain in the next section provide good illustrations of how this reduction can be realized.

Because of the nature of the reduced model grammars, the reading study uses only the HMM-based tracing methods—namely, fixation tracing and point tracing. Target tracing, due to its basis in sequence matching, requires that grammars be expanded into lists of every possible strategy (given the algorithm presented here); even with a threshold on the number of cycles or length of strategies, such a process would generate an infeasible strategy list for the reduced reading grammars. Thus, although target tracing provided good if not excellent results in the equation-solving studies, we cannot employ it at all in the reading study. This study, like the eye-typing study, illustrates how the novel HMM-based tracing methods facilitate analysis even for domains in which sequence-matching methods are entirely inapplicable.

Experiment

The reading experiment is a replication of the original SRC experiment (Schilling et al., 1998). In this experiment, participants read 48 medium-length (8- to 14-word) sentences and occasionally answered questions to ensure that they had processed the information. Our analysis includes data from 16 students, all of whom were native English speakers.

Protocol Analysis

As mentioned, enumerating every possible strategy (i.e., fixation sequence) for each sentence would create infeasibly large model grammars. Thus, it is necessary to build smaller model grammars that capture interesting aspects of the models’ behavior while eliminating much of the unnecessary or less interesting information. A straightforward way to build such grammars is to translate the predictions of the models to first- and second-order grammars—that is, grammars that describe only the first- or second-order transitions between words. These grammars assume that the next word to be fixated depends only on the current word (for a first-order grammar) or the current and previous word (for a second-order grammar). Although the grammars clearly have less information than one that contains all enumerated strategies, they embody the most important sequential information in the models and provide an excellent starting point for tracing the reading data.

The first- and second-order grammars were generated from simulations of the E-Z Reader models. Both E-Z Reader models comprise two primary component processes: lexical access, which checks the familiarity of a word and then retrieves its semantic representation, and saccade programming and execution, which plans when and where the eyes move and then executes the movement (under certain conditions). During a simulation run, E-Z Reader begins a familiarity check on the first word and then, when the check is completed, initiates full lexical access of the word as well as a saccade program to the next word. Depending on how the subsequent processes run, the programmed eye movement sometimes runs to completion and executes a saccade; however, the program is sometimes canceled by a subsequent program to another word. Thus, E-Z Reader provides an account of the indirect relation between the movement of attention (i.e., lexical processing) and the movement of the eyes. Consult Reichle et al. (1998) for a complete description of the simulation and running of the E-Z Reader models.

Simulations of the E-Z Reader models were used to generate the first- and second-order grammars as follows. First, for each sentence, the models were run for 300 simulations, and the resulting fixation sequences were collapsed into first- and second-order transitions; simulations with regressions to previous words were discarded. Because the E-Z Reader models start by fixating the first word, the model grammars assume that the first fixation always occurs on this word. The transitions for the first-order grammar comprised first-order transitions from each word to itself and every word to its right. The transitions for the second-order grammar comprised first-order transitions from the first word to all other words and second-order transitions from a pair of words to the second word and words to its right. Probabilities for the transitions were derived directly from the model simulations; in other words, each transition probability represented the number of times the transition occurred divided by the number of opportunities in which the transition could occur. This entire process results in the creation of a first- and second-order grammar for each of the 48 sentences.

Comparing Models

We first explore how the tracing methods provide a quantitative comparison of E-Z Readers 3 and 5 with respect to sequential information. By tracing the protocols with the first- and second-order grammars for each model, the tracing methods provide the mismatch score between each grammar and the observed data. Figure 13 shows the average mismatch scores per sentence for all grammars and for fixation and point tracing. In all cases, E-Z Reader 5 exhibits a lower mismatch score and thus provides a better fit to the data. Repeated measures analyses of variance show that the differences are

Figure 13. **Mismatch scores for the first-order and second-order reading models using fixation tracing and point tracing.**

First-Order Grammar				Second-Order Grammar			
Fixation		Point		Fixation		Point	
EZR 3	EZR 5	EZR 3	EZR 5	EZR 3	EZR 5	EZR 3	EZR 5
11.824	11.780	13.961	13.959	11.913	11.851	13.962	13.959

Note. Scores are not directly comparable between methods or grammars. EZR = E-Z Reader.

moderately significant for the first-order grammar with point tracing ($p < .1$) and very significant for all other grammars and methods ($p < .0001$). The differences are generally quite small because the models have only one important difference: E-Z Reader 5 posits that words farther in the periphery are more difficult to process, and E-Z Reader 3 does not. This distinction makes transitions to farther words slightly less likely for E-Z Reader 5 than for E-Z Reader 3, leading to small differences in the predictions of the models and thus in the goodness-of-fit scores. Nevertheless, even these small differences hold for almost all individual readers, producing significant within-readers effects. Note that scores are not directly comparable between methods, because of the different fixation HMMs, or between grammars, because of lower transition probabilities in the second-order grammar. This quantitative comparison of model fit thus supports Reichle et al.'s (1998) qualitative argument that E-Z Reader 5 is the better model of reading behavior.

Cleaning Data

Another interesting line of examination illustrates how tracing cleans up the data and facilitates analysis of temporal measures. In their original treatment of E-Z Reader, Reichle et al. (1998) examined the SRC data with three measures: *gaze duration*, or the sum of durations of consecutive fixations on a word; *first-fixation duration*, or the duration of only the first fixation on a word; and *single-fixation duration*, or the duration of only single fixations on a word. Figure 14 shows their SRC results for five classes of words where Class 5 words are most frequent and Class 1 words are least frequent. The overall result was that readers spent less time on more frequent words (i.e., with a higher class) than on less frequent words. To gather analogous results from the New data, we first analyzed these data without tracing by simply mapping fixations to the nearest words. With this analysis, as Figure 14 shows, the "Untraced New" data failed to manifest these effects due primarily to the noisier eye-movement data. Figure 14 illustrates this point by showing the correlation between the

Figure 14. **Mean gaze durations for the SRC data, Untraced New data, and Traced New data (in milliseconds).**

	Gaze			First Fixation			Single Fixation		
Class	SRC	Untraced New	Traced New	SRC	Untraced New	Traced New	SRC	Untraced New	Traced New
1	293	276	251	248	228	226	265	247	240
2	272	253	232	234	220	212	249	235	222
3	256	243	230	228	223	218	243	236	227
4	234	242	225	223	227	216	235	240	223
5	214	232	208	208	224	205	216	230	206
R	—	.93	.96	—	.22	.87	—	.82	.95
Span	79	44	43	40	8	21	49	17	34

Note. The Traced New data were analyzed with fixation tracing and the first-order grammar. $R =$ correlation between the SRC data and the New data; Span = the difference between the minimum and maximum gaze durations.

SRC and Untraced New results and the *span* of each result as defined by the difference between the smallest and largest duration. Although the Untraced New gaze durations closely resembled the SRC durations with a high correlation and reasonable span, the first-fixation durations did not match, and the single-fixation durations matched only to a lesser degree.

Tracing considerably cleans up the New data and provides a much closer match with the results of the SRC experiment. The traced results for the New data as interpreted by fixation tracing and the first-order grammar are denoted in Figure 14 as Traced New. In essence, tracing assigns some fixations to words other than the nearest words given the statistics of the model; for instance, if a fixation lies between two words, tracing would assign the fixation to the more likely of the two words (e.g., the word with the lower frequency of occurrence). The gaze duration results changed little, but the results for the other two measures were improved significantly in both the correlations and the size of the spans. These results, like the equation-solving results in the previous section, show how tracing facilitates understanding of temporal information given sequential model predictions. It may seem in some sense that we have simply massaged the data to produce the desired results; after all, tracing influences interpretation of the data according to the given model. However, we emphasize that tracing uses only sequential information in its interpretations; there is no a priori reason that the temporal aspects of the traced protocols should correlate better with the SRC data. The fact that the traced results match closely with the SRC results provides indirect but sound evidence that the tracing methods form accurate interpretations of the data.

4.3. Eye Typing

The third application explores the use of the tracing methods in an intelligent user interface (Salvucci, 1999a). In particular, we use a type of *gaze-based* interface in which users communicate with a computer by means of their eye movements—in our case, an "eye-typing" interface in which users type characters by looking at characters on an on-screen keypad (see Figure 1b). Researchers recently have developed gaze-based interfaces for a wide variety of tasks, including word processing (e.g., Frey, White, & Hutchinson, 1990; Gips, 1998; Hutchinson et al., 1989; Stampe & Reingold, 1995), operating system control (Salvucci & Anderson, 2000), musical composition (Gips, 1998), and control of robotic wheelchairs (Yanco, 1998). Although gaze-based interfaces may appear to have little in common with the previous two domains of equation solving and reading, the domains in fact share a critical commonality—the need to interpret human actions by mapping actions to the thoughts that produced them. However, gaze-based interfaces must interpret actions online and in real time, making them an excellent application domain in which to test both the speed and accuracy of tracing.

The eye-typing study emphasizes the use of the tracing methods to infer intent in a real-time interface and to provide more efficient, user-friendly user input. Previous eye-typing systems (e.g., Frey et al., 1990; Gips, 1998; Hutchinson et al., 1989; Stampe & Reingold, 1995) have enjoyed good success, enabling physically disabled users to type using their eye movements alone. However, previous systems used a fairly simple scheme to interpret eye movements: The user had to fixate precisely over a letter for some minimum time, or dwell threshold, for the letter to be typed. This scheme led to several required restrictions—such as large spacing between letters and long dwell thresholds—that facilitated interpretation but also limited their usability and design. This study examines a prototype eye-typing interface that minimizes these restrictions but, as expected, makes the interpretation of user input extremely difficult. The study uses this interface to explore how difficult interpretation becomes when restrictions are removed and how the tracing methods can assist in interpreting this input. The study then investigates the speed–accuracy tradeoffs of the various tracing methods and discusses their applicability to real-time interfaces.

Experiment

The eye-typing experiment involved typing words using the eyes on an on-screen keyboard, as shown in Figure 1b. For each trial, the user read the given word, typed the word by fixating its letters in order, and ended the trial by fixating the space bar. Repeated letters were typed by first fixating the letter and then fixating a special Repeat key at the bottom-right corner of the keyboard. On completion of the trial, the interface interpreted the trial protocol by using the full model and 12-word vocabulary (described subsequently) and displayed its interpretation for 1 sec in the space bar box. (Note that although the interface used only this model for online interpretation, the subsequent discussion describes offline interpretation using all the various models.) Seven users typed 12 different words four times each for a total of 336 trials.

Protocol Analysis

As for the reading domain, the eye-typing allows for several possible model grammars that incorporate differing amounts of sequential information. Three types of process models were used: a full model that enumerated every word as a sequence of letters, a second-order model that included only second-order transitions, and a first-order model with first-order transitions. These models enable tracing to make use of the sequential information in the vocabulary; for instance, after seeing a fixation on *Q*, the models would bias interpretation to favor interpreting the next fixation as *U*. In addition, we created models for three vocabularies of 12, 100, and 1,000 words, each of which included the 12 original typed words. The different model grammars and vocabularies allow us to explore the various speed–accuracy tradeoffs that arise with the tracing methods.

Inferring Intent Online

Figure 15a shows the percentage accuracy with which fixation tracing interpreted user protocols for each model. The figure includes results for each of the three models and for three vocabulary sizes; for comparison, it also includes results for a "no-model" condition in which fixations simply map to the nearest letter. In addition, because only the full model guarantees an interpretation that maps to a vocabulary word, we performed a sequence match on the no-model, first-order, and second-order interpretations to translate them to the nearest vocabulary word. As expected, models with more sequential information outperformed models with less information. The full model was highly accurate for all vocabularies. The second-order and first-order models showed good accuracy for the smaller vocabularies but degraded somewhat for larger vocabularies, whereas accuracy for no-model was worst for all vocabularies. Figure 15b shows the average time needed to interpret one protocol, in milliseconds. The accuracy of the more sophisticated models goes hand in hand with slower performance; these models are larger and thus require more time to interpret protocols. With one exception (the full model of the 1,000-word vocabulary), tracing interprets the protocols in real time—that is, faster than the time taken to generate the protocol. These times can be improved signifi-

Figure 15. **Eye-typing results showing (a) interpretation accuracy and (b) interpretation time (in milliseconds) for the full, second-order, and first-order model grammars and the no-model condition.**

(a)

Vocabulary Size	Full	Second-Order	First-Order	No Model
12	.99	.98	.94	.89
100	.99	.91	.83	.79
1,000	.95	.75	.74	.70

(b)

Vocabulary Size	Full	Second-Order	First-Order	No Model
12	124	106	70	58
100	867	483	157	136
1,000	9,104	1,771	866	883

Note. All values represent results derived by using fixation tracing and subsequent sequence matching as described in the text.

cantly by using common techniques such as a beam search; Salvucci (1999b) provided a detailed discussion of these techniques and their results.

These results have two important implications. First, the tracing methods clearly form accurate interpretations of the data; as for equation solving, we have direct evidence that the methods' interpretations correspond to known user strategies (i.e., the given words to be typed). Second, the tracing methods can provide a great deal of flexibility for designing and implementing online interpretation algorithms for use in intelligent user interfaces. Whereas existing methods primarily use the simplest algorithm (our no-model condition) for interpretation, the tracing methods allow for a wide range of possible models that provide different speed–accuracy characteristics. Thus, developers can trade off speed and accuracy as needed to achieve a good balance for a particular system.

Facilitating User Input

The inference of user intentions goes hand in hand with the facilitation of user input: By producing intelligent interpretations of user actions, an interface can more easily understand user intentions and thus provide faster and more accurate responses to these actions. We compared the average typing time in our eye-typing system to those of previous systems to evaluate whether our tracing-enhanced interface allowed for better user performance. For our system, users required 821 msec per character on average to eye-type the words; the fastest user averaged 430 msec, whereas the slowest user averaged

1,255 msec. These times are faster than those reported for other eye-typing systems (e.g., Hutchinson et al., 1989, 85 min/page; Stampe & Reingold, 1995, 1,870 msec/character). Users also showed improvement over the two halves of the experiment, averaging 870 msec in the first half and 772 msec in the second half. Thus, although our minimally restrictive interface increased the difficulty of interpreting protocols, the tracing methods compensated for this difficulty and allowed users to type at their desired speed without the need for restrictions such as dwell times and widely spaced keyboards.

4.4. Summary

The three domain applications provide a rigorous test for the tracing methods and illustrate how the tracing methods facilitate a number of aspects of protocol analysis and cognitive model development. The equation-solving and eye-typing domains provide direct evidence that the tracing methods generate accurate interpretations by comparing their interpretations to known "correct" interpretations. The reading and equation-solving domains provide indirect evidence of the methods' accuracy by showing how they facilitate higher level analysis of temporal aspects of the data. The eye-typing and equation-solving domains demonstrate that the tracing methods can produce interpretations much faster than human experts and in real time. The equation-solving domain illustrates tracing with a small number of enumerated strategies; the other two domains illustrate tracing with reduced grammars that represent a large number of strategies. Thus, the domains nicely complement one another and provide a good picture of how the tracing methods benefit different types of applications.

5. GENERAL DISCUSSION

The major contribution of this article is a class of tracing methods for interpreting eye-movement protocols with process models. The domain applications have shown that the tracing methods interpret eye movements as accurately as human experts in significantly less time. They also demonstrate the flexibility with which researchers can use different methods and process models to best suit their needs. Overall, the tracing methods, as well as tracing methodology in general, serve as a powerful tool that facilitates both offline analysis and modeling of user behavior and online interpretation of user intentions.

5.1. Tracing in HCI Research

Although we have discussed a number of uses for tracing in the context of the application domains, it is worthwhile to review the most important uses for research in HCI. Figure 16 summarizes these uses as well as their correspon-

Figure 16. **Illustrative uses for the tracing methods and their correspondence with the application domains.**

Uses	Equation Solving	Reading	Eye Typing
1. Classifying strategies	√		√
2. Coding protocols	+	+	+
3. Developing cognitive models	+		
4. Comparing models	+	√	
5. Cleaning data	+	√	
6. Inferring intent online			√
7. Facilitating input			√

Note. Check marks (√) represent uses and applications described in this article; plus signs (+) represent additional uses and applications described in Salvucci (1999b).

dence with the studied domains; the figure includes both correspondences that are described in this article (√) as well as additional correspondences realized and described elsewhere (+), particularly in Salvucci (1999b).

The first two uses of eye-movement tracing in Figure 16 address the offline interpretation of protocols for rigorous study and understanding of behavior. Strategy classification (Use 1) assigns protocols to one of a set number of model strategies. Protocol coding (Use 2), a more generalized form of strategy classification, maps observed eye movements to predicted fixations and subgoals as described by the model. The classified strategies or hierarchies of predicted subgoals enable a variety of subsequent analyses; for instance, they may be used to study strategy frequencies, strategy adaptation, metastrategies, and between-user and within-user variability. The automated nature of the tracing methods enables such analyses to encompass a much larger set of protocols than hand classification or coding could allow.

Use 3 involves tracing for the purposes of developing a cognitive model of user behavior. As described earlier, several researchers, most notably Ritter and Larkin (1994), have formalized a methodology for developing models by using an iterative trace-based approach. Clearly, the eye-movement tracing methods described here fit directly into this framework, allowing the same type of development for models of eye movements and visual attention. We ourselves have developed a model of behavior in an “unconstrained” equation-solving task in which students solve equations however they choose. The unconstrained task provided no a priori process model like the instructed strategies in the described equation-solving task. To develop such a model, we began with an exploratory analysis of the data, using tracing to identify fixations and map them to the nearest targets (i.e., with no model bias). This analysis led to an initial model of behavior that we iteratively evaluated and refined: At each stage of the iteration, we traced the data with the model, determined where the model failed to predict behavior, and refined the model accordingly. The final model enabled a higher level analysis that showed a significant effect of computation on students' gaze durations; this analysis was only possible because of the protocol coding provided by tracing.

The next two uses relate closely to model development and higher level analysis. Tracing allows for model comparison (Use 4) by providing a mismatch score, or goodness-of-fit measure, that describes how well the model predicts the data. As we did in the reading study, comparing mismatch scores for different models gives a quantitative comparison of the models with respect to sequential information—that is, how well the different models predict the occurrences of fixations and transitions between fixations. Tracing also cleans up data (Use 5) to facilitate higher level analysis and thus obtain a more accurate sense of both the behavior in the task and the model's accuracy in accounting for this behavior. Like the unconstrained equation-solving analysis and the reading study described earlier, tracing alleviates noise by using sequential information such that analysis of other measures, such as temporal information, more accurately reflects behavior.

The final two uses address tracing in the context of intelligent user interfaces. Because of its speed and accuracy, tracing can provide intelligent interfaces with real-time estimations of user intent (Use 6); that is, tracing determines the user's intentions based on a noisy input signal that represents the user's actions. As opposed to the offline nature of coding and model development, online tracing is essential for user interfaces that react to user actions as they occur. For instance, intelligent tutoring systems typically trace student actions to infer their current state of knowledge. In such systems, eye movements help disambiguate problem-solving strategies that cannot be inferred solely from more typical data such as keystrokes (Gluck, 1999). As another example, gaze-based user interfaces such as our eye-typing interface allow users to execute commands with explicit or implicit eye movements. As a consequence of accurate inference of user intent, tracing facilitates user input (Use 7) by minimizing interface restrictions and allowing the user to control the interface more freely. The eye-typing study demonstrates that, although allowing freer input can lead to serious difficulties in interpretation, the tracing methods alleviate these difficulties and perform as well as or better than existing interpretation techniques. Thus, better eye-movement tracing algorithms enable intelligent interfaces to better understand user intentions and thus provide more efficient, usable interaction.

5.2. Method Comparison and Recommendations

Based on our experiences with the tracing methods, we can offer a general comparison of their advantages and disadvantages along with recommendations for their use. Target tracing provides fast interpretations with reasonable accuracy and robustness to protocol noise. Its primary advantage is the ease in which a user can understand and implement the algorithm. Its accuracy, although not as good as the other algorithms, is likely adequate for many simple applications. Target tracing has three major disadvantages that hinder its use in practical applications. First and foremost, the algorithm cannot handle complex process models with many possible enumerated strategies. Hierarchical or circular process models with even a small number of rules can generate enough enumerated strategies to make target tracing infeasible. Second, target tracing cannot incorporate rule probabilities in any systematically meaningful way, forcing process models to consider every strategy equally likely. Third, target tracing produces a fairly coarse mismatch score that can have only integer values. Nevertheless, target tracing is a more than adequate tracing algorithm that should serve well in simple applications.

Fixation tracing generates more accurate and robust interpretations than target tracing with only a minor loss of speed. In addition, unlike target tracing, fixation tracing can handle hierarchical and circular process models and strategies of different probabilities. Its only real disadvantage, the difficulty of implementing the algorithm, can be alleviated through the use of a public or commercial package for HMM construction and decoding. In all, fixation tracing stands out as the most powerful and usable of the three tracing algorithms. Its balance of speed and accuracy, plus its flexibility in acceptable process models, makes it an excellent choice for many applications.

Point tracing is the least efficient of the tracing algorithms but also provides the best accuracy, with reservations. Although it allows the process model to influence all aspects of tracing, its major difference with fixation tracing—the ability of the model to influence fixation identification—does not seem to have a major impact on tracing; in reality, fixation identification is often fairly robust, and thus sequential information in a process model helps very little in the process. In fact, this difference sometimes causes a problem with finding incidental fixations because point tracing is more likely to identify them as saccades and not count them as incidental fixations. Thus, the extra predictive power of point tracing does not significantly improve its performance and sometimes may even degrade the usefulness of its resulting traces. Overall, fixation tracing seems to serve best as the tracing method of choice for most applications.

NOTES

Background. This article is based on Dario Salvucci's dissertation research under the direction of John Anderson.

Acknowledgments. We thank Ken Koedinger, Herb Simon, Keith Rayner, Erik Reichle, Scott MacKenzie, the ACT–R research group, and three anonymous reviewers for many insightful comments and informative discussions in the development of this research.

Support. This work was supported in part by National Science Foundation Grant CDA–9720359 and Office of Naval Research Grant N00014–96–I–0491.

Authors' Present Addresses. Dario Salvucci, Department of Math and Computer Science, Drexel University, 3141 Chestnut Street, Philadelphia, PA 19104. E-mail: salvucci@drexel.edu. John Anderson, Department of Psychology, Carnegie Mellon University, Pittsburgh, PA 15213. E-mail: ja@cmu.edu.

HCI Editorial Record. First manuscript received April 12, 2000. Accepted by Scott MacKenzie. Final manuscript received September 29, 2000. — *Editor*

REFERENCES

Aaltonen, A., Hyrskykari, A., & Räihä, K. (1998). 101 spots, or how do users read menus? *Human Factors in Computing Systems: CHI'98 Conference Proceedings,* 132–139. New York: ACM.

Anderson, J. R. (1993). *Rules of the mind.* Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

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

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

Bhaskar, R., & Simon, H. A. (1977). Problem solving in semantically rich domains: An example from engineering thermodynamics. *Cognitive Science, 1,* 193–215.

Byrne, M. D., Anderson, J. A., Douglass, S., & Matessa, M. (1999). Eye tracking the visual search of click-down menus. *Human Factors in Computing Systems: CHI'99 Conference Proceedings,* 402–409. New York: ACM.

Cabiati, C., Pastormerlo, M., Schmid, R., & Zambarbieri, D. (1983). Computer analysis of saccadic eye movements. In R. Groner, C. Menz, D. F. Fisher, & R. A. Monty (Eds.), *Eye movements and psychological functions: International views* (pp. 19–29). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Card, S., Moran, T., & Newell, A. (1983). *The psychology of human–computer interaction.* Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Carpenter, P. A., & Just, M. A. (1978). Eye fixations during mental rotation. In J. W. Senders, D. F. Fisher, & R. A. Monty (Eds.), *Eye movements and the higher psychological processes* (pp. 115–133). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Crosby, M. E., & Wesley, W. (1991). Using eye movements to classify search strategies. *Proceedings of the Human Factors and Ergonomics Society 35th Annual Meeting,* 1476–1480. Santa Monica, CA: Human Factors and Ergonomics Society.

Edwards, G. (1998). A tool for creating eye-aware applications that adapt to changes in user behavior. *Proceedings of ASSETS'98,* 67–74. New York: ACM.

Ericsson, K. A., & Simon, H. A. (1980). Verbal reports as data. *Psychological Review, 87,* 215–251.

Ericsson, K. A., & Simon, H. A. (1984). *Protocol analysis: Verbal reports as data.* Cambridge, MA: MIT Press.

Erkelens, C. J., & Vogels, I. M. L. C. (1995). The initial direction and landing position of saccades. In J. M. Findlay, R. Walker, & R. W. Kentridge (Eds.), *Eye movement research: Mechanisms, processes, and applications* (pp. 133–144). New York: Elsevier Science.

Fisher, C. (1991). *Protocol analyst's workbench: Design and evaluation of computer-aided protocol analysis.* Unpublished doctoral dissertation, Department of Psychology, Carnegie Mellon University, Pittsburgh, PA.

Flagg, B. N. (1978). Children and television: Effects of stimulus repetition on eye activity. In J. W. Senders, D. F. Fisher, & R. A. Monty (Eds.), *Eye movements and the higher psychological processes* (pp. 279–291). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

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, Inc.

Frey, L. A., White, K. P., & Hutchinson, T. E. (1990). Eye-gaze word processing. *IEEE Transactions on Systems, Man, and Cybernetics, 20,* 944–950.

Fuchs, A. F. (1971). The saccadic system. In P. Bach-y-Rita, C. C. Collins, & J. E. Hyde (Eds.), *The control of eye movements* (pp. 343–362). New York: Academic.

Garlick, S., & VanLehn, K. (1987). *CIRRUS: An automated protocol analysis tool* (Technical Report 6). Pittsburgh, PA: Department of Psychology, Carnegie Mellon University.

Gips, J. (1998). On building intelligence into EagleEyes. In V. O. Mittal, H. A. Yanco, J. Aronis, & R. Simpson (Eds.), *Assistive technology and artificial intelligence* (pp. 50–58). Berlin: Springer-Verlag.

Gluck, K. A. (1999). *Eye movements and algebra tutoring.* Unpublished doctoral dissertation, Department of Psychology, Carnegie Mellon University, Pittsburgh, PA.

Goldberg, J. H., & Kotval, X. P. (1998). Computer interface evaluation using eye movements: Methods and constructs. *International Journal of Industrial Ergonomics, 24,* 631–645.

Goldberg, J. H., & Schryver, J. C. (1995). Eye-gaze determination of user intent at the computer interface. In J. M. Findlay, R. Walker, & R. W. Kentridge (Eds.), *Eye movement research: Mechanisms, processes, and applications* (pp. 491–502). New York: Elsevier Science.

Goldberg, J. H., Zak, R. E., & Probart, K. (1999). Visual search of food nutrition labels. *Human Factors, 41,* 425–437.

Harris, C. M., Hainline, M., Abramov, I., Lemerise, E., & Camenzuli, C. (1988). The distribution of fixation durations in infants and naive adults. *Vision Research, 28,* 419–432.

Hegarty, M., Mayer, R. E., & Green, C. E. (1992). Comprehension of arithmetic word problems: Evidence from students' eye fixations. *Journal of Educational Psychology, 84,* 76–84.

Hoffman, J. E. (1998). Visual attention and eye movements. In H. Pashler (Ed.), *Attention* (pp. 119–153). Hove, England: Psychology Press.

Huang, X. D., Ariki, Y., & Jack, M. A. (1990). *Hidden Markov models for speech recognition.* Edinburgh, Scotland: Edinburgh University Press.

Hutchinson, T. E., White, K. P., Martin, W. N., Reichert, K. C., & Frey, L. A. (1989). Human–computer interaction using eye-gaze input. *IEEE Transactions on Systems, Man, and Cybernetics, 19,* 1527–1534.

Jacob, R. J. K. (1991). The use of eye movements in human–computer interaction techniques: What you look at is what you get. *ACM Transactions on Information Systems, 9,* 152–169.

Jacob, R. J. K. (1995). Eye tracking in advanced interface design. In W. Barfield & T. A. Furness (Eds.), *Virtual environments and advanced interface design* (pp. 258–288). New York: Oxford University Press.

Just, M. A., & Carpenter, P. A. (1980). A theory of reading: From eye fixations to comprehension. *Psychological Review, 87,* 329–354.

Just, M. A., & Carpenter, P. A. (1984). Using eye fixations to study reading comprehension. In D. E. Kieras & M. A. Just (Eds.), *New methods in reading comprehension research* (pp. 151–182). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Just, M. A., & Carpenter, P. A. (1992). A capacity theory of comprehension: Individual differences in working memory. *Psychological Review, 99,* 122–149.

Karsh, R., & Breitenbach, F. W. (1983). Looking at looking: The amorphous fixation measure. In R. Groner, C. Menz, D. F. Fisher, & R. A. Monty (Eds.), *Eye movements and psychological functions: International views* (pp. 53–64). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

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

Kowler, E. (1990). The role of visual and cognitive processes in the control of eye movement. In E. Kowler (Ed.), *Eye movements and their role in visual and cognitive processes* (pp. 1–70). New York: Elsevier Science.

Kowler, E., Martins, A. J., & Pavel, M. (1984). The effect of expectations on slow oculomotor control IV: Anticipatory smooth eye movements depend on prior target motions. *Vision Research, 24,* 197–210.

Kruskal, J. B. (1983). An overview of sequence comparison: Time warps, string edits, and macromolecules. *SIAM Review, 25,* 201–234.

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

Lallement, Y. (1998). A hierarchical ensemble of decision trees applied to classifying data from a psychological experiment. *Proceedings of the Eleventh International FLAIRS Conference,* 230–234. Sanibel Island, FL: AAAI Press.

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.

Legge, G. E., Klitz, T. S., & Tjan, B. S. (1997). Mr. Chips: An ideal-observer model of reading. *Psychological Review, 104,* 524–553.

Lohse, G. L. (1997). Consumer eye movement patterns on yellow pages advertising. *Journal of Advertising, 26,* 61–73.

Lohse, G. L., & Johnson, E. J. (1996). A comparison of two process tracing methods for choice tasks. *Organizational Behavior and Human Decision Processes, 68,* 28–43.

Lowerre, B., & Reddy, R. (1980). The HARPY speech understanding system. In W. Lea (Ed.), *Trends in speech recognition* (pp. 340–346). Englewood Cliffs, NJ: Prentice Hall.

McConkie, G. W., & Rayner, K. (1976). Asymmetry of the perceptual span in reading. *Bulletin of the Psychonomic Society, 8,* 365–368.

McDowell, E. D., & Rockwell, T. H. (1978). An exploratory investigation of the stochastic nature of the drivers' eye movements and their relationship to roadway geometry. In J. W. Senders, D. F. Fisher, & R. A. Monty (Eds.), *Eye movements and the higher psychological processes* (pp. 329–345). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

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

Noton, D., & Stark, L. (1971). Scanpaths in saccadic eye movements while viewing and recognizing patterns. *Vision Research, 11,* 929–942.

Ohlsson, S. (1990). Trace analysis and spatial reasoning: An example of intensive cognitive diagnosis and its implications for testing. In N. Frederiksen, R. Glaser, A. Lesgold, & M. G. Shafto (Eds.), *Diagnostic monitoring of skill and knowledge acquisition* (pp. 251–296). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

O'Regan, K. (1981). The "convenient viewing position" hypothesis. In D. F. Fisher, R. A. Monty, & J. W. Senders (Eds.), *Eye movements: Cognition and visual perception* (pp. 289–298). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Peck, V. A., & John, B. E. (1992). The Browser model: A computational model of a highly interactive task. *Proceedings of the CHI'92 Conference on Human Factors in Computer Systems,* 165–172. New York: ACM.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE, 77,* 257–286.

Rayner, K. (1995). Eye movements and cognitive processes in reading, visual search, and scene perception. In J. M. Findlay, R. Walker, & R. W. Kentridge (Eds.), *Eye movement research: Mechanisms, processes, and applications* (pp. 3–21). New York: Elsevier Science.

Rayner, K., & Morris, R. K. (1990). Do eye movements reflect higher order processes in reading? In R. Groner, G. d'Ydewalle, & R. Parham (Eds.), *From eye to mind: Information acquisition in perception, search, and reading* (pp. 179–190). New York: Elsevier Science.

Reichle, E. D., Pollatsek, A., Fisher, D. L., & Rayner, K. (1998). Toward a model of eye movement control in reading. *Psychological Review, 105,* 125–157.

Rimey, R. S., & Brown, C. M. (1991). Controlling eye movements with hidden Markov models. *International Journal of Computer Vision, 7,* 47–65.

Ritter, F. E. (1992). *A methodology and software environment for testing process models' sequential predictions with protocols.* Unpublished doctoral dissertation, Department of Psychology, Carnegie Mellon University, Pittsburgh, PA.

Ritter, F. E., & Larkin, J. H. (1994). Developing process models as summaries of HCI action sequences. *Human–Computer Interaction, 9,* 345–383.

Salvucci, D. D. (1999a). Inferring intent in eye-movement interfaces: Tracing user actions with process models. *Human Factors in Computing Systems: CHI'99 Conference Proceedings,* 254–261. New York: ACM.

Salvucci, D. D. (1999b). *Mapping eye movements to cognitive processes.* Unpublished doctoral dissertation, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Salvucci, D. D., & Anderson, J. R. (1998). Tracing eye movement protocols with cognitive process models. *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society,* 923–928. Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

Salvucci, D. D., & Anderson, J. R. (2000). Intelligent gaze-added interfaces. *Human Factors in Computing Systems: CHI'2000 Conference Proceedings,* 273–280. New York: ACM.

Salvucci, D. D., & Anderson, J. R. (2001). Integrating analogical mapping and general problem solving: The path-mapping theory. *Cognitive Science, 25,* 67–110.

Salvucci, D. D., Anderson, J. R., & Douglass, S. (2000). *Interleaving visual attention and problem solving.* Manuscript in preparation.

Sanderson, P., Scott, J., Johnston, T., Mainzer, J., Watanabe, L., & James, J. (1994). MacSHAPA and the enterprise of exploratory sequential data analysis (ESDA). *International Journal of Human–Computer Studies, 41,* 633–681.

Schilling, H. E. H., Rayner, K., & Chumbley, J. I. (1998). Comparing naming, lexical decision, and eye fixation times: Word frequency effects and individual differences. *Memory & Cognition, 26,* 1270–1281.

Sen, T., & Megaw, T. (1984). The effects of task variables and prolonged performance on saccadic eye movement parameters. In A. G. Gale & F. Johnson (Eds.), *Theoretical and applied aspects of eye movement research* (pp. 103–111). Amsterdam: Elsevier.

Smith, J. B., Smith, D. K., & Kuptsas, E. (1993). Automated protocol analysis. *Human–Computer Interaction, 8,* 101–145.

Stampe, D. M., & Reingold, E. M. (1995). Selection by looking: A novel computer interface and its application to psychological research. In J. M. Findlay, R. Walker, & R. W. Kentridge (Eds.), *Eye movement research: Mechanisms, processes, and applications* (pp. 467–478). New York: Elsevier Science.

Stark, L., & Ellis, S. R. (1981). Scanpath revisited: Cognitive models of direct active looking. In D. F. Fisher, R. A. Monty, & J. W. Senders (Eds.), *Eye movements: Cognition and visual perception* (pp. 193–226). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Sudkamp, T. A. (1988). *Languages and machines.* New York: Addison-Wesley.

Suppes, P. (1990). Eye-movement models for arithmetic and reading performance. In E. Kowler (Ed.), *Eye movements and their role in visual and cognitive processes* (pp. 455–477). New York: Elsevier Science.

Suppes, P., Cohen, M., Laddaga, R., Anliker, J., & Floyd, R. (1983). A procedural theory of eye movements in doing arithmetic. *Journal of Mathematical Psychology, 27,* 341–369.

Tole, J. R., & Young, L. R. (1981). Digital filters for saccade and fixation detection. In D. F. Fisher, R. A. Monty, & J. W. Senders (Eds.), *Eye movements: Cognition and visual perception* (pp. 247–256). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Viviani, P. (1990). Eye movements in visual search: Cognitive, perceptual, and motor control aspects. In E. Kowler (Ed.), *Eye movements and their role in visual and cognitive processes* (pp. 353–393). New York: Elsevier Science.

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

Waterman, D. A., & Newell, A. (1973). *PAS–II: An interactive task-free version of an automatic protocol analysis system* (Technical Report 73–34). Pittsburgh, PA: Department of Computer Science, Carnegie Mellon University.

Widdel, H. (1984). Operational problems in analysing eye movements. In A. G. Gale & F. Johnson (Eds.), *Theoretical and applied aspects of eye movement research* (pp. 21–29). New York: Elsevier Science.

Winikoff, A. (1967). *Eye movements as an aid to protocol analysis of problem solving.* Unpublished doctoral dissertation, Department of Psychology, Carnegie Mellon University, Pittsburgh, PA.

Wolfe, B., & Eichmann, D. (1997). A neural network approach to tracking eye position. *International Journal of Human–Computer Interaction, 9,* 59–79.

Yanco, H. A. (1998). Wheelesley: A robotic wheelchair system: Indoor navigation and user interface. In V. O. Mittal, H. A. Yanco, J. Aronis, & R. Simpson (Eds.), *Assistive technology and artificial intelligence* (pp. 256–268). Berlin: Springer-Verlag.

Young, L. R., & Sheena, D. (1975). Survey of eye movement recording methods. *Behavioral Research Methods and Instrumentation, 7,* 397–429.

Zhai, S., Morimoto, C., & Ihde, S. (1999). Manual and gaze input cascaded (MAGIC) pointing. *Human Factors in Computing Systems: CHI'99 Conference Proceedings,* 246–253. New York: ACM.