

Mapping Eye Movements to Cognitive Processes

Dario D. Salvucci
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Doctoral Thesis Summary

May 10, 1999

Abstract

Eye movements provide a rich and informative window into a person's thoughts and intentions. In recent years researchers have increasingly employed eye movements to study cognition in psychological experiments, to understand behavior in user interfaces, and even to control computers through eye-based input devices. Unfortunately, like speech and handwriting, eye movements generate vast amounts of data with significant individual variability and equipment noise. Thus, the analysis of eye-movement data—that is, determining what people are thinking based on where they are looking—can be extremely tedious and time-consuming. Typical eye-movement data sets are simply too large and complex to be analyzed by hand or by naive automated methods.

This thesis formalizes a new class of algorithms that provide fast and robust automated analysis of eye-movement data. Specifically, the thesis describes three novel algorithms for *tracing* eye movements—mapping eye-movement protocols to the sequential predictions of a cognitive process model. Two algorithms, *fixation tracing* and *point tracing*, employ hidden Markov models to determine the best probabilistic interpretation of the data given the model. The third algorithm, *target tracing*, extends an existing tracing algorithm based on sequence matching to eye movements. The thesis also formalizes several algorithms for identifying fixations in raw eye-movement protocols and provides a working system, EyeTracer, that embodies the proposed tracing and fixation-identification algorithms.

To demonstrate the power of the proposed algorithms, the thesis applies them in three real-world domains: equation solving, reading, and eye typing. The equation-solving studies show how the algorithms can code, or interpret, eye-movement protocols as accurately as expert human coders in significantly less time. The studies also illustrate how the algorithms facilitate the prototyping and refinement of cognitive models. The reading study demonstrates how the algorithms help to evaluate and compare two existing computational models of reading and clear up temporal aspects of reading data using sequential aspects of the data. The eye-typing study shows how the algorithms can interpret eye movements in real time and help eliminate usability restrictions imposed by existing eye-based interfaces.

1. Introduction

Our eyes reveal a great deal about us; whether or not our eyes are “windows to the soul,” as the common saying goes, they are certainly windows to the mind. The connection between eye movements and thoughts has enjoyed great attention in recent years in a number of major research areas. In psychology and the cognitive sciences, researchers have studied eye-movement protocols—recorded sequences of actions—to help elicit and analyze the cognitive processes in a variety of task domains, such as reading (e.g., Just & Carpenter, 1980, 1984; Schilling, Rayner, & Chumbley, 1998), arithmetic (Suppes, 1990), and word problems (e.g., Hegarty, Mayer, & Green, 1992), to name a few. In neurobiology and vision research, researchers have focused on the underlying mechanisms of eye movements (e.g., Erkelens & Vogels, 1995; Fuchs, 1971; Vaughan, 1982; Zingale & Kowler, 1987) and laid a solid foundation for understanding fundamental characteristics of human visual processing. In human-computer interaction, researchers have studied eye movements to better understand interface use (e.g., Aaltonen, Hyrskykari, & Riih , 1998) and to develop successful eye-driven user interfaces (e.g., Hutchinson et al., 1989; Stampe & Reingold, 1995).

There are a number of important reasons why eye movements have become so popular in so many research fields. Eye-movement protocols—sequences of recorded eye-gaze locations—represent actions at a fine temporal grain size, typically on the order of 10 milliseconds. At this grain size, eye movements yield important clues to human behavior, including what information people use in problem solving and when they use it; how much time people need to process various pieces of information; and when people forget and review previously encoded information. Also, humans need little if any instruction or training to produce informative data; in most applications, eye-movement data is collected non-intrusively, such that data collection in no way affects task performance. In addition, eye movements can serve as the sole source of data or as a supplement to other sources like verbal protocols. Thus, while eye movements certainly do not entirely reveal a person’s thoughts, their flexibility and informativeness make them an excellent data source for many studies and applications.

Although eye movements are extremely flexible and informative, they are also very time-consuming and tedious to analyze. 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. In addition, eye-movement protocols typically include a great deal of noise due to human and equipment variability. For large eye-movement data sets with hundreds or thousands of trial protocols, it is simply implausible for humans to code the data consistently, accurately, and in a reasonable amount of time. Computer tools that assist protocol analysis allow investigators to analyze larger, more complex data sets in a consistent, detailed manner that would otherwise be impossible.

This thesis explores a new class of algorithms that analyze eye movements 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 has become an extremely useful tool for analyzing data in many contexts. Cognitive scientists have employed tracing to analyze verbal protocols in various problem-solving domains (e.g., Newell & Simon, 1972). Builders of intelligent tutoring systems have used tracing to determine a student’s solution path through a student model of the learning domain (e.g., Anderson et al., 1990). Researchers in human-computer interaction have employed tracing to study the fits of user models (e.g., Card, Moran, & Newell, 1983) and to develop intelligent interfaces that analyze and react to user actions.

Automated tracing of eye movements has a number of applications in numerous areas of research including those mentioned above. For instance, in the cognitive sciences, tracing assists in prototyping and evaluating low-level models of visual attention using large sets of eye movement protocols. In human-computer interaction, tracing provides important low-level information on users’ encoding strategies and help interfaces with eye-based input devices to determine user intentions. In intelligent tutoring systems, eye movements help disambiguate problem-solving strategies which cannot be inferred solely from the student’s mouse clicks and key presses.

The following subsections provide further details on the problem to be solved and the proposed approach to solving it. The next section gives a full overview of the thesis and describes its major results and contributions.

1.1 The Problem

The central problem in this thesis is the *tracing* of eye movements, that is, the analysis and interpretation of eye movement protocols with respect to some cognitive model. We will operate under the following assumptions:

- The eye-movement protocol consists of a sequence of point-of-regard data, each of which gives the planar coordinates of the estimated look point on a computer screen. We assume that the point data are sampled at a constant rate, which depends on the specific eye-tracking equipment used.
- The cognitive process model using in tracing is a model capable of generating one or more predicted action sequences, such as a production system (e.g., Anderson & Lebiere, 1998) or cognitive grammar (e.g., Smith, Smith, & Kuptsas, 1993). We assume that the model predicts visual processing at the level of fixations or gazes. In addition, we allow the model to be non-deterministic, such that each possible predicted sequence occurs with some probability.
- The task environment in which eye-movement data are collected is (at least for the most part) static, with few if any changes occurring during any individual trial. The display should have easily-defined visual targets, such as numbers or words, that roughly translate to rectangular regions of interest. (The issues of tracing in dynamic environments and inferring regions of interest are addressed as future extensions.)

Under these assumptions, tracing eye movements corresponds to relating the observed eye movement protocol with one of the possible action sequences predicted by the cognitive model. The trace thus maps each point of the protocol to a predicted action in the sequence. Such a mapping can be used for both exploratory and confirmatory purposes to discover, evaluate, and refine the model.

The automated tracing of eye movement protocols is very difficult for several reasons. First, equipment such as eye trackers can introduce a significant amount of noise, sometimes rendering protocols difficult or impossible to interpret sensibly. Second, subjects often take shortcuts or err in executing their intended actions; for instance, they may produce a saccade that falls short of its intended destination. Third, there are no guarantees that every trial actually corresponds to one of the predefined strategies; the subject may exhibit some combination of strategies or a new one altogether. A successful automated analysis system must take serious care to address such problems when tracing eye movement protocols.

1.2 The Approach

This thesis proposes three algorithms that automate the tracing of eye movements: target tracing, fixation tracing, and point tracing. As input, the algorithms take an eye-movement protocol, a cognitive process model, and a set of targets (regions of interest). As output, the algorithms generate a mapping from eye movements to model predictions as well as a measure of “goodness of fit.” Each algorithm trades off speed and accuracy in different ways: the fastest provides only moderate accuracy, while the slowest provides very high accuracy. All together, they represent a powerful class of algorithms that allow developers to balance the speed-accuracy tradeoff as necessary for particular applications.

The first algorithm, target tracing, starts with a sequence of fixation locations and maps these locations to their nearest respective targets. The target sequence can optionally be transformed such that consecutive gazes on the same target are collapsed into a single gaze. Target tracing then matches the resulting target sequence to the predicted action sequences of the cognitive model. The sequences are compared using a sequence-distance metric (Card, Moran, & Newell, 1983; Kruskal, 1983) based on the number of inserted and deleted targets from one sequence to the other. The interpretation of the target

sequence is simply the predicted model sequence that best matches the target sequence. This interpretation includes a mapping from the observed to the predicted sequence, which helps identify whether or not individual gazes are predicted by the model. Target tracing provides lesser interpretation accuracy compared to the other algorithms but can run significantly faster.

The other two algorithms, fixation tracing and point tracing, use hidden Markov models (HMMs) to interpret eye movements. HMMs are probabilistic finite state machines that have been used effectively in many areas of study, most notably in the fields of speech and handwriting recognition (see Rabiner, 1989). In many ways, the analysis of eye movements has much in common with speech and handwriting recognition. These recognition systems take a person's speech or handwriting input and determine the most likely interpretation of this input given a model of the person's possible intentions. The tracing algorithms described here perform the analogous task for eye movements, taking a person's eye movements and determining the most likely sequence of intended fixations. Fixation and point tracing borrow a number of techniques from these well-developed fields and carry them over to eye-movement data.

Fixation tracing utilizes HMMs to map fixations onto predicted model sequences. Fixation tracing begins by creating a *tracer HMM* that embodies a grammar representing all possible model sequences. The tracer HMM comprises a number of *submodel HMMs* that represent single fixations on particular targets; the submodel HMMs contain probability distributions for the possible x and y locations of fixations over the target, as well as probability distributions for transitions through and around the submodel. The tracer HMM thus contains a probabilistic representation of possible model sequences and possible fixation locations for each target in those sequences. Fixation tracing then determines the most likely interpretation of a fixation sequence given a tracer HMM; in other words, it maps fixations in the fixation sequence onto states in the HMM. Fixation tracing provides more accurate interpretations than target tracing due to its probabilistic representation: Although it tends to assign fixations to their nearest targets, as does target tracing, fixation tracing has the freedom to guide interpretation to other targets if they are deemed more likely by the given model.

Point tracing operates much like fixation tracing, except that it combines fixation identification and tracing into a single step that maps raw eye-movement data directly to predicted model sequences. Point tracing also forms a tracer HMM, but its submodel HMMs represent properties of data points rather than fixations. Specifically, each submodel HMM includes two states: one that represent high-velocity saccade points, and one that represents low-velocity fixation points. Point tracing then determines the most likely interpretation of a given data point sequence given the model HMM. Point tracing, like fixation tracing, has the freedom to guide interpretation of points to non-nearest targets. However, unlike fixation tracing, point tracing allows the model to influence whether particular data points are identified as fixations or saccades. Point tracing is thus the most powerful of the three algorithms but can incur substantially more computation.

2. Thesis Summary and Contributions

This thesis offers a number of important contributions to the fields of computer science, cognitive science, psychology, human-computer interaction, and related areas. Most importantly, it introduces and formalizes three tracing algorithms for interpreting eye movements with cognitive process models. It also collects and formalizes a number of existing and novel algorithms for identifying fixations in raw eye-movement protocols. The thesis evaluates the tracing and fixation-identification algorithms in three real-world application domains and demonstrates the benefits of the algorithms for each domain. All algorithms are implemented in a working system, EyeTracer, that allows for eye-movement protocol manipulation, visualization, and analysis.

2.1 Algorithms for Tracing Eye Movements

The first and most important contribution of this thesis is a class of tracing algorithms for interpreting eye movements with cognitive process models (Chapter 4). The thesis describes and formalizes three tracing algorithms: target tracing, fixation tracing, and point tracing. Target tracing is essentially an

extension of an existing tracing algorithm that utilizes sequence matching to trace generic action protocols (Card, Moran, Newell, 1983). Fixation and point tracing are novel algorithms that apply hidden Markov models (HMMs) to tracing eye movements. Altogether, the three represent a powerful toolkit of algorithms that can be utilized for a variety of purposes, including protocol coding, model prototyping and refinement, model evaluation and comparison, and real-time inference of user intentions.

2.1.1 Evaluating Tracing Accuracy and Speed

To evaluate the accuracy and speed of the tracing algorithms, the thesis applies them to three domains—equation solving, reading, and eye-typing—and tests them directly and indirectly in the context of these domains. The constrained equation-solving and eye-typing studies provide the most direct tests of tracing accuracy. The constrained equation-solving study uses an instructed-strategy paradigm to give a “correct” interpretation of subject protocols. The interpretations of the tracing algorithms correspond very well to the “correct” interpretations; in fact, they are as accurate or more accurate than those of expert human coders. The eye-typing study is very similar in that the “correct” interpretation of a user’s eye movements is known, namely the letters of the given word to be typed. Again the tracing algorithms provide accurate interpretations when compared to the “correct” interpretations. In both studies, point tracing and fixation tracing gave the best accuracy with target tracing close behind.

The unconstrained equation-solving and reading studies provide a more indirect evaluation of tracing accuracy. In the unconstrained equation-solving study, traced interpretations of protocols clear up subjects’ intended fixation targets to better reveal when subjects perform intermediate computation. In the reading study, traced interpretations provide a better fit to expected results with respect to fixation durations and probabilities. Both studies demonstrate that the sequential information utilized by tracing helps sort out temporal information in the protocols, thus giving indirect evidence that the tracing algorithms generate sensible interpretations of the protocols. Amongst each other, the tracing algorithms differ little with respect to their ability to sort out temporal information.

The constrained equation-solving and eye-typing studies evaluate the speed of the tracing algorithms. The constrained equation-solving study shows that all the tracing algorithms interpret protocols approximately an order of magnitude faster than expert human coders. The eye-typing study shows that all algorithms can run in real-time, at least for smaller vocabularies of 1000 words or less. In addition, the study demonstrates that the algorithms differ greatly in terms of their speed-accuracy tradeoffs: target tracing is the fastest but least accurate algorithms, point tracing is the slowest but (generally) most accurate algorithm, and fixation tracing is almost as fast as target tracing and almost as accurate as point tracing.

2.1.2 Conclusions

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, while 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 goodness-of-fit metric 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. Its speed can be improved with memoization of tracer HMMs and process models that generate sparse HMMs. In addition, unlike target tracing, fixation tracing can handle hierarchical and circular process models, overlapping and embedded target areas, and strategies of

different probabilities. The difficulty of implementing the algorithm, its one major disadvantage, can be alleviated through the use of a public or commercial package for HMM construction and decoding. Overall, 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 cognitive process models, makes it an excellent choice for many applications. In addition, as we discuss shortly, fixation tracing is the most amenable to future extensions such as the incorporation of fixation duration information and the generalization to dynamic task environments.

Point tracing is the least efficient of the tracing algorithms but also provides the best accuracy. Although it allows the cognitive 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.

2.2 Algorithms for Identifying Fixations

Another important contribution of the thesis is the collection and formalization of a set of useful algorithms for fixation identification (Chapter 3). While a number of researchers have developed such algorithms, only a few have compared the various algorithms (e.g., Karsh & Breitenbach, 1983) and to a very limited extent. The thesis gathers several existing and novel algorithms and collects them into a single presentation. I-VT (velocity-threshold identification), I-DT (dispersion-threshold identification), and I-TA (target-area identification) are formalizations of algorithms employed in previous work. I-HMM (hidden Markov model identification), like fixation and point tracing, is a novel application of HMMs to eye-movement data analysis. Like the tracing algorithms, these identification algorithms represent a useful toolkit of methods for different needs and applications.

2.2.1 Evaluating Identification Accuracy and Speed

The thesis evaluates the accuracy and speed of the four identification algorithms in the context of the application domains. The evaluation of identification accuracy arises primarily in the constrained equation-solving and eye-typing studies, where each algorithm is tested for its ability to facilitate target tracing and produce accurate tracing results. Both studies have similar results: I-HMM, I-VT, and I-DT give more accurate and robust interpretations than I-TA. The evaluation of identification speed, primarily in the constrained equation-solving study, shows that all four algorithms are fairly similar in terms of real-time efficiency, with I-HMM requiring slightly more time than the other three algorithms.

2.2.2 Conclusions

The choice of fixation-identification algorithm depends greatly on whether data analysis focuses on fixations or gazes (aggregations of consecutive fixations on the same target). For gaze analysis, I-HMM, I-VT, and I-DT all give good results; I-TA tends to miss more fixations and requires the specification of target areas, making it the least favorable choice. For fixation analysis, I-HMM and I-DT give the best results; I-TA cannot account for multiple fixations, and I-VT sometimes identifies too many fixations when point velocities hover near threshold. I-HMM may be the favorable choice because all its parameters can be estimated directly from data; however, I-DT may be preferred because it does not require implementation of HMM reestimation and decoding procedures.

2.3 Domain Applications

The third major contribution of this thesis is the application of the tracing and identification algorithms to three real-world domains: equation solving, reading, and eye typing (Chapters 6-8). As described above, the studies of these domains help evaluate the accuracy and speed of the algorithms.

However, they also illustrate how the algorithms, and tracing more generally, can greatly benefit work in each domain. The three domains are highly complementary, and thus together allow for a comprehensive demonstration of the power and usefulness of the tracing algorithms.

2.3.1 Equation Solving

The equation-solving studies demonstrate how tracing assists in coding experimental eye-movement protocols and in developing cognitive process models (Chapter 6). The constrained study uses an “instructed-strategy” paradigm in which subjects are given specific problem-solving strategies and asked to utilize these strategies in the equation-solving task. As is often done with verbal protocols, the resulting eye-movement protocols can be coded by categorizing each protocol as one of the given strategies. Results show that the tracing algorithms can code protocols as accurately or more accurately than expert human coders, but in much less time. These results manifest the great promise for these and similar tracing algorithms to code eye-movement protocols and thus facilitate higher-level analyses.

The unconstrained study focuses on modeling subject behavior in the unconstrained task, in which subjects could solve problems in whatever way they choose. The study first performs an exploratory analysis on the data set using frequency trees and protocol visualization to identify common strategies. Starting with a prototype model grammar that embodies these strategies, the study then performs three iterations of trace-based protocol analysis (Ritter & Larkin, 1994), where each iteration comprises: tracing the data set with the model, evaluating the model and identifying model-data mismatches, and refining the model accordingly. After the three iterations, the study shows how the traced data help understanding of subject behavior, especially with respect to temporal aspects of behavior (e.g., gaze durations). The study also takes the final model grammar and from it derives an ACT-R/PM (Anderson & Lebiere, 1998) cognitive model of subject behavior. This model predicts both the cognitive steps involved in the task and the time of each observable action (i.e., gaze or keystroke). Thus, the unconstrained study illustrates the use of the eye-movement tracing algorithms in trace-based analysis for the creation and refinement of a cognitive process model.

2.3.2 Reading

The reading study demonstrates how tracing facilitates the evaluation and comparison of existing cognitive process models (Chapter 7). The study focuses on two computational process models of reading, E-Z Reader 3 and 5 (Reichle et al., 1998). It first derives first- and second-order model grammars from the E-Z Reader models that represent their first- and second-order transition characteristics, respectively. With these grammars, the study compares the E-Z Reader models and finds that the qualitatively better model, E-Z Reader 5, produces a better quantitative fit to collected data. In addition, the study shows that, like in the unconstrained equation-solving study, tracing cleans up temporal aspects of the data (i.e., fixation durations), producing results that better conform to earlier reported results. The study gives a good illustration of how the tracing algorithms help compare cognitive process models and sort out temporal information by means of sequential information.

2.3.3 Eye Typing

The eye-typing study illustrates how tracing assists in designing and implementing eye-based user interfaces (Chapter 8). The experiment in the study uses a prototype eye-typing interface in which users can type words by looking at letters on an on-screen keypad. This interface differs from previous interfaces in that it eliminates two major restrictions imposed by these interfaces: large spacing between letters and long dwell times to actuate a typed letter. A naive algorithm that simply maps fixations to the nearest letter does an extremely poor job at interpreting user eye movements. However, the tracing algorithms, for different models and vocabularies, provide reasonable to excellent tracing accuracy with corresponding tradeoffs in tracing speed. Overall, the study shows that the tracing algorithms help compensate for the difficulty of eliminating the two common interface restrictions. It also shows that the algorithms can be used for real-time interpretation in eye-based interfaces or any interfaces that may benefit from tracing eye movements, such as intelligent tutoring systems.

2.4 The EyeTracer System

A fourth contribution of this thesis is the EyeTracer system, a development environment in which a user can manipulate and analyze eye-movement protocols (Chapter 5). EyeTracer embodies all the algorithms described in the thesis and provides both a testbed for the algorithms and a working system for anyone interested in tracing eye-movement protocols. The system has a menu-driven interface in which a user can view and analyze protocols for both exploratory and confirmatory data analysis. The system is also modular such that a user can simply enter information specific to a particular data set and run the algorithms on the data set with no further implementation. EyeTracer runs in the Macintosh Common Lisp environment and is publicly available at the thesis web site.

2.5 Extensions and Future Work

The final component of the thesis (Chapter 9) describes a number of issues, recommendations, and extensions pertaining to this work. First, the chapter addresses a number of issues raised in the thesis, including how to develop effective cognitive models for tracing, how to improve cognitive models using hidden Markov model training, and how to implement the various algorithms efficiently. Second, the chapter includes recommendations pertaining to the applicability of the algorithms in different contexts—for instance, which algorithms can feasibly operate in real-time as part of an on-line system. Third, the chapter discusses possible future extensions for this work, including incorporation of fixation duration information, extension to dynamic task environments, and generalization to multimodal data.

References

- Aaltonen, A., Hyrskykari, A., & Rähkä, K. (1998). 101 spots, or how do users read menus?. In *Human Factors in Computing Systems: CHI 98 Conference Proceedings* (pp. 132-139). New York: ACM Press.
- Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Anderson, J. R., Boyle, C. F., Corbett, A. T., & Lewis, M. W. (1990). Cognitive modeling and intelligent tutoring. *Artificial Intelligence*, 42, 7-49.
- Card, S., Moran, T., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- 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 Publishing.
- 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 Press.
- 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.
- 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.
- 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*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- 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.
- Kruskal, J. B. (1983). An overview of sequence comparison: Time warps, string edits, and macromolecules. *SIAM Review*, 25, 201-234.
- Newell, A., & Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice Hall.

- Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77, 257-286.
- 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.
- Ritter, F. E., & Larkin, J. H. (1994). Developing process models as summaries of HCI action sequences. *Human-Computer Interaction*, 9, 345-383.
- 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.
- 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 Publishing.
- 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 Publishing.
- Zingale, C. M., & Kowler, E. (1987). Planning sequences of saccades. *Vision Research*, 27, 1327-1341.