WHAT YOU LOOK AT IS WHAT YOU GET: EYE MOVEMENT-BASED INTERACTION TECHNIQUES

Robert J.K. Jacob

Human-Computer Interaction Lab
Naval Research Laboratory
Washington, D.C.

ABSTRACT

In seeking hitherto-unused methods by which users and computers can communicate, we investigate the usefulness of eye movements as a fast and convenient auxiliary user-to-computer communication mode. The barrier to exploiting this medium has not been eye-tracking technology but the study of interaction techniques that incorporate eye movements into the user-computer dialogue in a natural and unobtrusive way. This paper discusses some of the human factors and technical considerations that arise in trying to use eye movements as an input medium, describes our approach and the first eye movement-based interaction techniques that we have devised and implemented in our laboratory, and reports our experiences and observations on them.

KEYWORDS: Eye movements, eye tracking, interaction techniques, human-computer interaction, input.

INTRODUCTION

Current user-computer dialogues tend to be one-sided, with the bandwidth from the computer to the user far greater than that from user to computer. A fast and effortless mode of communication from a user to a computer would help redress this imbalance. We therefore investigate the possibility of introducing the movements of a user's eyes as an additional input medium. While the technology for measuring eye movements in real time has been improving, what is needed is appropriate *interaction techniques* that incorporate eye movements into the user-computer dialogue in a convenient and natural way. This paper discusses some of the human factors and technical considerations that arise in trying to use eye movements as an input medium, describes our approach and the first eye movement-based interaction techniques that we have devised and implemented in our laboratory, and reports our experiences and observations on them.

BACKGROUND

Methods for Measuring Eye Movements

Available techniques for measuring eye movements range from the not-quite-sublime to the almost-ridiculous. First, note that our goal is to measure *visual line of gaze,* that is, the absolute position in space at which the user's eyes are pointed, rather than, for example, the position of the eyeball in space or the relative motion of the eye within the head [14].

The simplest eye tracking technique is electronic recording, using electrodes placed on the skin around the eye to measure changes in the orientation of the potential difference that exists between the cornea and the retina. However, this method is more useful for measuring relative eye movements than absolute position. Perhaps the least user-friendly approach uses a contact lens that fits precisely over the bulge at the front of the eyeball and is held in place with a slight suction. This method is extremely accurate, but suitable only for laboratory studies. More practical methods use remote imaging of a visible feature located on the eyeball, such as the boundary between the sclera and iris, the outline of the pupil, or the corneal reflection of a light shone at the eye. All these require the head to be held absolutely stationary (a bite board is customarily used), to be sure that any measured movement represents movement of the eye, not the head. However, by tracking two features of the eye simultaneously, it is possible to distinguish head movements (the two features move together) from eye movements (the two move with respect to one another), and the head

need not be rigidly fixed. This is currently the most practical method for use in a conventional computer-and-user setting, since the eye tracker sits several feet from the user, nothing contacts him or her, and the head need not be clamped. In our laboratory, we use an Applied Science Laboratories (Waltham, Mass.) Model 3250R eye tracker [9,14]. Figure 1 shows the components of this type of eye tracker. It simultaneously tracks the corneal reflection (from an infrared light shining on eye) and the outline of the pupil (illuminated by same light). Visual line of gaze is computed from the relationship between the two tracked points.

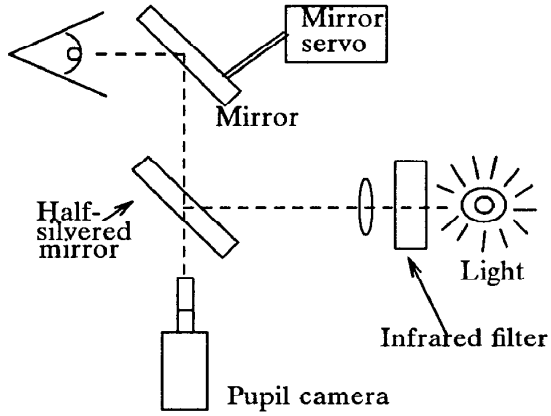

Figure 1. Illustration of components of a corneal reflection-plus-pupil eye tracker. The pupil camera and illuminator operate along the same optical axis, via a half-silvered mirror. The servo-controlled mirror is used to compensate for the user's head motions.

Previous Work

While technology for measuring visual line of gaze is adequate, there has been little research on *using* this information in real time. There is a considerable body of research using eye tracking, but it has concentrated on eye movement data as a tool for studying motor and cognitive processes by recording the eye movements and subsequently analyzing them [7,10]. Real-time eye input has been used most frequently for disabled (quadriplegic) users, who can use only their eyes for input [4,8]. Our interest is, instead, on dialogues that combine real-time eye movement data with other, more conventional modes of user-computer communication. Richard Bolt did some of the earliest work in this particular area and demonstrated several innovative uses of eye movements [1,2]. Floyd Glenn [5] used eye movements for several tracking tasks involving moving targets. Ware and Mikaelian [13] reported an experiment in which simple target selection and cursor positioning operations were performed substantially faster with an eye tracker than with any of the more conventional cursor positioning devices.

Characteristics of Eye Movements

To see an object clearly, it is necessary to move the eyeball so that the object appears on the fovea, a small area at the center of the retina. Because of this, a person's eye position provides a rather good indication (to within the one-degree width of the fovea) of what specific portion of the scene before him he is examining. The most common way of moving the eyes is a sudden, ballistic, and nearly instantaneous saccade. It is typically followed by a fixation, a 200-600 ms. period of relative stability during which an object can be viewed. During a fixation, however, the eye still makes small, jittery motions, generally covering less than one degree. Smooth eye motions, less sudden than saccades, occur only in response to a moving object in the visual field. Other eye movements, such as nystagmus, vergence, and torsional rotation are relatively insignificant in a user-computer dialogue.

The overall picture of eye movements for a user sitting in front of a computer is a collection of steady (but slightly jittery) fixations connected by sudden, rapid saccades. The eyes are rarely entirely still. They move during a fixation, and they seldom remain in one fixation for long. Figure 2 shows a trace of eye movements (with jitter removed) for a user using a computer for 30 seconds. Compared to the slow and deliberate way people operate a mouse or other manual input device, eye movements careen madly about the screen. During a fixation, a user generally thinks he is looking steadily at a single object–he is not consciously aware of the small, jittery motions. This suggests that the human-computer dialogue should be constructed so that it, too, ignores those motions, since, ultimately, it should correspond to what the user *thinks* he is doing, rather than what his eye muscles are actually doing.

"Midas Touch" Problem

The most naive approach to using eye position as an input might be as a direct substitute for a mouse: changes in the user's line of gaze would cause the mouse cursor to move. This is an unworkable (and annoying) approach, because people are not accustomed to operating devices just by moving their eyes. They expect to be able to look at an item without having the look "mean" something. Normal visual perception requires that the eyes move about, scanning the scene before them. It is not desirable for each such move to initiate a computer command.

At first, it is empowering simply to look at what you want and have it happen. Before long, though, it becomes like the Midas Touch. Everywhere you look, another command is activated; you cannot look anywhere without issuing a command. The challenge in building a useful eye

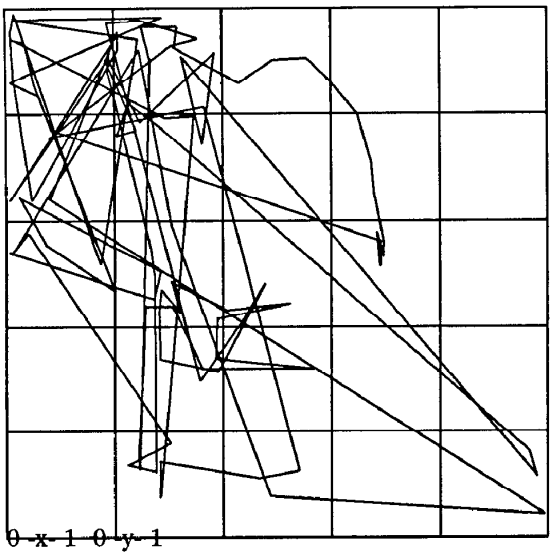

Figure 2. A trace of a computer user's eye movements over approximately 30 seconds, while performing normal work (i.e., no eye-operate interfaces) using a windowed display. Jitter within each fixation has been removed from this plot.

tracker interface is to avoid this Midas Touch problem. Ideally, the interface should act on the user's eye input when he wants it to and let him just look around when that's what he wants, but the two cases are impossible to distinguish in general. Instead, we investigate interaction techniques that address this problem in specific cases.

EXPERIENCE WITH EYE MOVEMENTS

Configuration

We use an Applied Science Laboratories corneal reflection eye tracker. The user sits at a conventional (government-issue) desk, with a Sun computer display, mouse, and keyboard, in a standard chair and office. The eye tracker camera/illuminator sits on the desk next to the monitor. Other than the illuminator box with its dim red glow, the overall setting is thus far just like that for an ordinary office computer user. In addition, the room lights are dimmed to keep the user's pupil from becoming too small. The eye tracker transmits the x and y coordinates for the user's visual line of gaze every 1/60 second, on a serial port, to a Sun 4/260 computer. The Sun performs all further processing, filtering, fixation recognition, and some additional calibration and also implements the user interfaces under study.

Observation: The eye tracker is, strictly speaking, non-intrusive and does not touch the user in any way. Our setting is almost identical to that for a user of a conventional office computer. Nevertheless, we find it is difficult to ignore the eye tracker. It is noisy; the dimmed room lighting is unusual; the dull red light, while not annoying, is a constant reminder of the equipment; and, most significantly, the action of the servo-controlled mirror, which results in the red light following the slightest motions of user's head gives one the eerie feeling of being watched.

Accuracy and Range

A user generally need not position his eye more accurately than the width of the fovea (about one degree) to see an object sharply. Finer accuracy from an eye tracker might be needed for studying the operation of the eye muscles but is not useful for our purposes. The eye's normal jittering further limits the practical accuracy of eye tracking. It is possible to improve accuracy by averaging over a fixation, but not in a real-time interface.

Observation: Despite the mechanisms for following the user's head, we find that the steadier the user holds his head, the better the eye tracker works. We find that we can generally get two degrees accuracy quite easily, and sometimes can achieve one degree. The eye tracker should thus be viewed as having a resolution much coarser than that of a mouse or other typical devices, perhaps more like a touch screen. A further problem is that the range over which the eye can be tracked with this equipment is fairly limited. In our configuration, it can barely cover the surface of a 19" monitor at a 24" viewing distance.

Using the Eye Tracker Data

Our approach to processing eye movement data is to partition the problem into two stages. First we process the raw eye tracker data in order to filter noise, recognize fixations, compensate for local calibration errors, and generally try to reconstruct the user's more conscious intentions from the available information. This processing stage converts the continuous, somewhat noisy stream of raw eye position reports into tokens that are claimed to approximate more closely the user's intentions in a higher-level user-computer dialogue. Then, we design generic interaction techniques based on these tokens as inputs.

Observation: Because eye movements are so different from conventional computer inputs, we achieve success with a philosophy that tries, as much as possible, to use natural eye movements as an implicit input, rather than to train a user to move the eyes in a particular way to operate the system. We try to think of eye position more as a piece of information available to the user-computer dialogue involving a variety of input devices than as the intentional actuation of an input device.

Local Calibration

The eye tracker calibration procedure produces a mapping that is applied uniformly to the whole screen, but we found small calibration errors appear in portions of the screen, rather than systematically across it. We introduced an additional layer of calibration into the chain, which allows the user to make local modifications to the calibration dynamically. If the user feels the eye tracker is not responding accurately in some area of the screen, he can at any point move the mouse cursor to that area, look at the cursor, and click a button.

Observation: Surprisingly, this had the effect of increasing the apparent response speed for object selection and other interaction techniques. The reason is that, if the calibration is slightly wrong in a local region and the user stares at a target in that region, the eye tracker will report the eye position somewhere slightly outside the target. If he continues to stare at it, though, his eyes will in fact jitter around to a spot that the eye tracker will report as being on the target. The effect feels as though the system is responding too slowly, but it is a problem of local calibration.

Fixation Recognition

After improving the calibration, we still observed erratic behavior in the user interface, even when the user thought he was staring perfectly still. This comes from both the normal jittery motions of the eye during fixations and from artifacts introduced when the eye tracker momentarily fails to obtain an adequate video image of the eye.

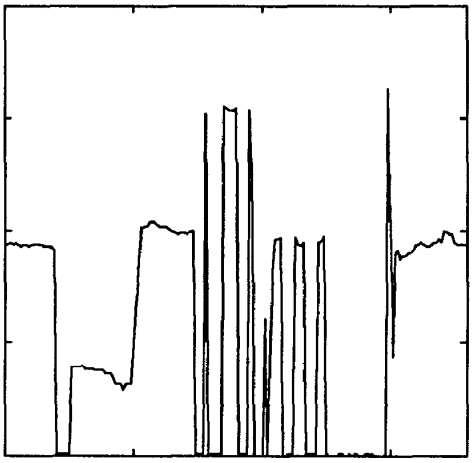

Figure 3. Illustration of erratic nature of raw data from the eye tracker. The plot shows one coordinate of eye position vs. time, over a somewhat worse-than-typical three second period.

Figure 3 shows the type of data obtained from the eye tracker. It plots the x coordinate of the eye position output against time over a relatively jumpy three-second period. Zero values on the ordinate represent periods when the eye tracker could not locate the line of gaze. This might be caused by eye tracker artifacts, such as glare in the video camera, lag in compensating for head motion, or failure of the processing algorithm, or by actual user actions, such as blinks or movements outside the range of the eye tracker. During the period represented by Figure 3, the subject thought he was simply looking around at a few different points on a CRT screen. The difference is attributable not only to the eye tracker artifacts but to the fact that much of the fine-grained behavior of the eye muscles is not intentional. To make a reasonable input to a user-computer dialogue from the eye tracker data, we must filter out that behavior to recover the "intentional" component of the eye motions.

We return to the picture of a computer user's eye movements as a collection of jittery fixations connected by essentially instantaneous saccades. We start with an *a priori* model of such saccades and fixations and then attempt to recognize and quickly report the start, approximate position, and end of each recognized fixation. Blinks of up to 200 ms. may occur during a fixation without terminating it. At first, blinks seemed to present a problem, since, obviously, we cannot obtain eye position data during a blink. However (equally obviously in retrospect), the screen need not respond to the eye during that blink period, since the user can't see it anyway. After applying this algorithm, the noisy data shown in Figure 3 are found to comprise about 6 fixations, which more accurately reflects what the user thought he was doing (rather than what his eye muscles plus the eye tracking equipment actually did). Figure 4 shows the same data, with a horizontal line marking each recognized fixation at the time and location it would be reported.

Observation: Applying the fixation recognition approach to the real-time data coming from the eye tracker yielded a significant improvement in the user-visible behavior of the interface. Filtering the data based on an *a priori* model of eye motion is an important step in transforming the raw eye tracker output into a user-computer dialogue.

User Interface Management System

We next turn the output of the recognition algorithm into a stream of *tokens* for use as input to an interactive user interface. We report tokens for eye events considered meaningful to the dialogue, much like tokens generated by mouse or keyboard events. We then multiplex the eye tokens into the same stream with those generated by the mouse and keyboard and present the overall token stream as input to our user interface management system. The desired user interface is specified to the UIMS as a collection of concurrently executing interaction objects [6]. The operation of each such object

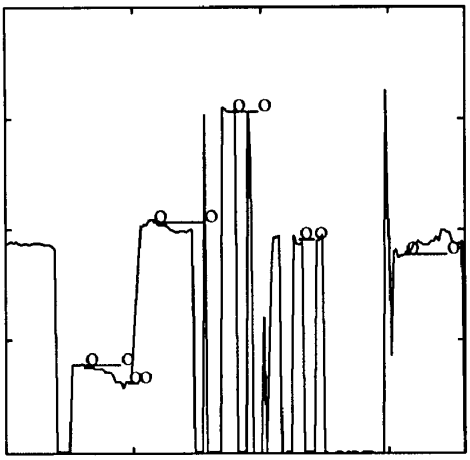

Figure 4. Result of applying the fixation recognition algorithm to the data of Figure 3. A horizontal line beginning and ending with an **o** marks each fixation at the time and coordinate position it would be reported.

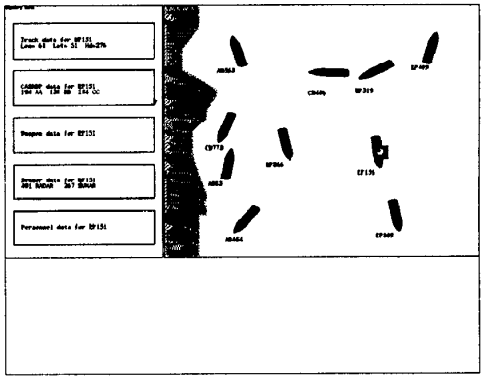

Figure 5. Display from eye tracker testbed, illustrating object selection technique. Whenever the user looks at a ship in the right window, the ship is selected and information about it is displayed in left window. The square eye icon at the right is used to show where the user's eye was pointing in these illustrations; it does not normally appear on the screen. The actual screen image uses light figures on a dark background to keep the pupil large.

is described by a state transition diagram that accepts the tokens as input. Each object can accept any combination of eye, mouse, and keyboard tokens, as specified in its own syntax diagram.

INTERACTION TECHNIQUES

An interaction technique is a way of using a physical input device to perform a generic task in a human-computer dialogue [11]. It represents an abstraction of some common class of interactive task, for example, choosing one of several objects shown on a display screen. This section describes the first few eye movement-based interaction techniques that we have implemented and our initial observations from using them.

Object Selection

The task here is to select one object from among several displayed on the screen, for example, one of several file icons on a desktop or, as shown in Figure 5, one of several ships on a map in a hypothetical "command and control" system. With a mouse, this is usually done by pointing at the object and then pressing a button. With the eye tracker, there is no natural counterpart of the button press. We reject using a blink for a signal because it detracts from the naturalness possible with an eye movement-based dialogue by requiring the user to think about when he or she blinks. We tested two alternatives. In one, the user looks at the desired object then presses a button on a keypad to indicate that the looked-at object is his choice. In Figure 5, the user has looked at ship "EF151" and caused it to be selected (for attribute display, described below). The second uses dwell time–if the user continues to look at the object for a sufficiently long time, it is selected without further operations. The two techniques can be implemented simultaneously, where the button press is optional and can be used to avoid waiting for the dwell time to expire, much as an optional menu accelerator key is used to avoid traversing a menu.

Observation: At first this seemed like a good combination. In practice, however, the dwell time approach is much more convenient. While a long dwell time might be used to ensure that an inadvertent selection will not be made by simply "looking around" on the display, this mitigates the speed advantage of using eye movements for input and also reduces the responsiveness of the interface. To reduce dwell time, we make a further distinction. If the result of selecting the wrong object can be undone trivially (selection of a wrong object followed by a selection of the right object causes no adverse effect–the second selection instantaneously overrides the first), then a very short dwell time can be used. For example, if selecting an object causes a display of information about that object to appear and the information display can be changed instantaneously, then the effect of selecting wrong objects is immediately undone as long as the user eventually reaches the right one. This approach, using a 150-250 ms. dwell time gives excellent results. The lag between eye movement and system response (required to reach the dwell time) is hardly detectable to the user, yet long enough to accumulate sufficient data for our fixation recognition and processing. The subjec-

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.