Intelligent Gaze-Added Interfaces

Dario D. Salvucci
Cambridge Basic Research
Four Cambridge Center
Cambridge, MA 02142
+1 617 374 9669
dario@cbr.com

John R. Anderson
Department of Psychology
Carnegie Mellon University
Pittsburgh, PA 15213 USA
+1 412 268 2788
ja+@cmu.edu

ABSTRACT
We discuss a novel type of interface, the intelligent gaze-added interface, and describe the design and evaluation of a sample gaze-added operating-system interface. Gaze-added interfaces, like current gaze-based systems, allow users to execute commands using their eyes. However, while most gaze-based systems replace the functionality of other inputs with that of gaze, gaze-added interfaces simply add gaze functionality that the user can employ if and when desired. Intelligent gaze-added interfaces utilize a probabilistic algorithm and user model to interpret gaze focus and alleviate typical problems with eye-tracking data. We extended a standard WIMP operating-system interface into a new interface, IGO, that incorporates intelligent gaze-added input. In a user study, we found that users quickly adapted to the new interface and utilized gaze effectively both alone and with other inputs.

Keywords
Gaze-added interfaces, gaze-based interfaces, intelligent interfaces, eye movements, user models.

INTRODUCTION
In the quest to facilitate human-computer interaction, a number of researchers have developed *gaze-based* interfaces in which a user controls the computer using his/her eye movements [e.g., 2, 3, 11]. Gaze-based interfaces have proven especially useful for physically-disabled users, for whom gaze control is the only, or easiest, available method of input. Such interfaces cover a wide range of applications, including typing and word processing [e.g., 2, 5, 10] and locomotion and control [e.g., 12]. While gaze-based interfaces have also shown promise for able-bodied users in specific contexts [e.g., 10, 13], they have yet to make a significant impact on the design and development of today's most common user interfaces.

In this paper, we describe the design and evaluation of a special type of gaze-based interface that we call the *gaze-added* interface. Most existing gaze-based interfaces replace the functionality of certain input(s), such as the mouse, with gaze input. In contrast, gaze-added interfaces provide exactly the same functionality as similar standard (non-gaze) interfaces but also add the ability to utilize gaze input when desired. In doing so, gaze-added interfaces give users more flexibility in choosing when and how to employ gaze input. The few existing systems that could be categorized as gaze-added interfaces [e.g., 13] have manifested the large potential benefits for these types of interfaces.

All gaze-based interfaces must deal with a significant problem in both their design and implementation: the difficulty of interpreting user eye movements. The interpretation of gaze input requires an assignment of each gaze to the visual target to which the user is attending during the gaze. This assignment is often complex for at least two reasons: inherent noise in the eye-tracking equipment, and the dissociation between the gaze point and the user's visual attention. To alleviate this problem, we developed an *intelligent* gaze-added interface that incorporates a probabilistic model of user behavior. The model helps to guide the interface to more likely interpretations of observed gazes, thus creating a significantly more responsive and user-friendly interface.

To demonstrate the power of intelligent gaze-added interfaces, we extended a standard WIMP (Window, Icon, Menu, Pointer) operating-system interface to incorporate intelligent gaze-added input. The WIMP operating-system interface was chosen for several reasons. Such interfaces are extremely common in today's most popular operating systems, such as Apple Mac OS and Microsoft Windows. In addition, these interfaces typically do not require the accurate pointing that interfaces for other applications, such as word processing or drawing, normally require; this feature facilitates the use of gaze input given the noise and variability in eye-tracking data. Our novel operating-system interface, IGO (Intelligent Gaze-added Operating system), thus serves as a realistic but manageable example of a gaze-added interface that has important implications for the development of future gaze-based interfaces.

INTERFACE DESIGN AND IMPLEMENTATION
IGO, the intelligent gaze-added operating-system interface, is modeled primarily on the Mac OS system but has a great degree of overlap with Windows and related systems. We now describe IGO in three components: the basic non-gaze interface, the gaze-added interface, and the intelligent gaze interpretation as determined by the probabilistic user model.

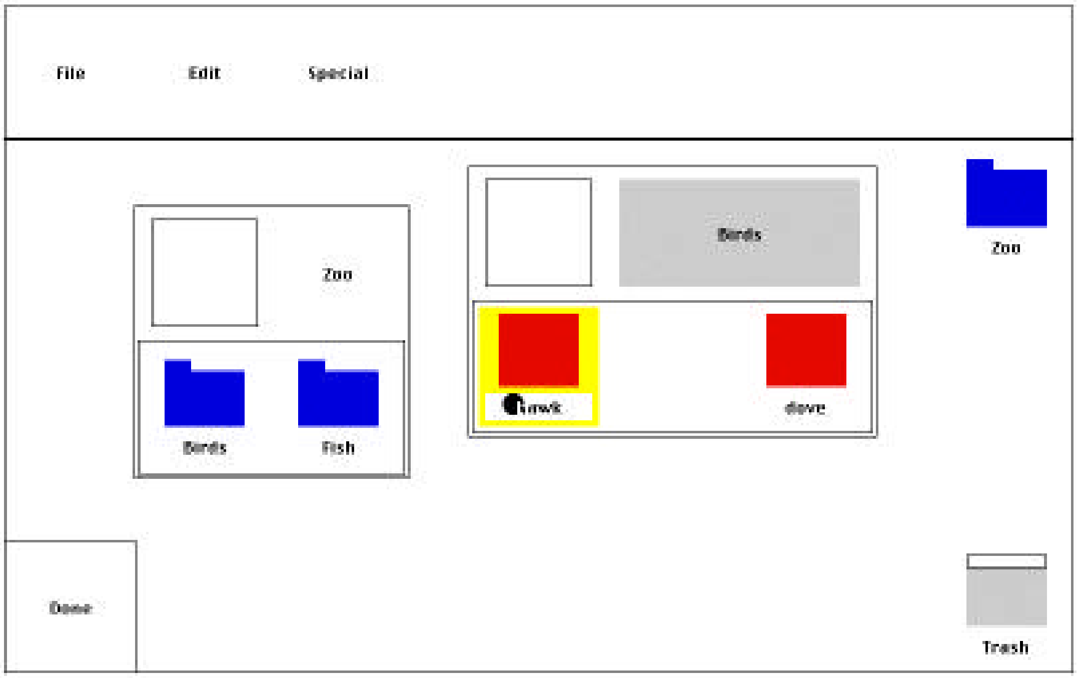

Figure 1: Sample screen from IGO. The inverted circle on the **hawk** icon marks the user's current gaze point, and the light (yellow) highlighting around the icon indicates that this icon is the current gaze focus.

The Basic Non-Gaze Interface

In essence, IGO is extremely similar to common existing interfaces with respect to the look and feel of the interface. A sample interface screen is shown in Figure 1. The screen includes windows, icons for folders (e.g., **Birds**) and files (e.g., **hawk**), and a menu bar at the top of the screen. Windows include a title bar at the top with a close box in the upper-left. Icons are color-coded such that folders appear in red and files in blue. Note that some items (e.g., the close boxes) are larger than in typical operating-system interfaces because current limitations in eye-tracking equipment preclude closely-spaced items; however, the design will generalize nicely as eye trackers improve and allow for more tightly-spaced items.

The basic, non-gaze functionality of IGO is identical to that in typical WIMP operating systems. By controlling an on-screen pointer with a mouse, users can select icons, drag icons to other parts of the screen, open and close folder windows, and select various menu options. Opening folders requires a double-click on a folder icon, and dragging requires a click-down on the icon to be dragged and a click-up at the new location of the icon. Menu selection can be performed in two ways: with a click-down on the menu name and click-up on the menu item, or with two separate clicks on the menu name and item ("sticky menus" in Macintosh terminology). Icons can be renamed by selecting the icon and typing its new name on the keyboard.

The interface as currently implemented contains three menus based on the Mac OS menu set. The File menu includes three items: Create Folder, which creates an untitled folder in the current window; Print, which is currently unused; and Duplicate, which duplicates a selected icon. The Edit menu includes three items: Cut, Copy, and Paste. The Special menu contains a single item, Empty Trash, which clears the contents of the special **Trash** folder. These menu items are clearly a subset of those needed for a full-fledged system but do provide a diverse set of commands for use in the evaluation stage.

The Gaze-Added Interface

IGO augments the basic interface to allow for gaze input in a way fully analogous to mouse input. The system employs an eye tracker (described later) to send gaze information to the interface. As the user looks around the screen, the item to which the user is attending is highlighted with a yellow background; this highlighting is analogous to the on-screen mouse pointer in that it shows the current focus of attention for the gaze modality. The user can then use the Control key to actuate a command for the highlighted item. This *gaze button* is completely analogous to the mouse button and has the exact same functionality — that is, single clicks for mouse correspond to single clicks for gaze, double-clicks correspond to double-clicks, drags to drags, etc.

To better illustrate the gaze-based functionality of IGO, let us briefly consider how a user might perform the task of throwing away a file by dragging it to the **Trash** folder. To open the appropriate folder(s), the user first looks at the folder's icon (thus highlighting it) and double-clicks the gaze button to open its window. Then, to drag a file to the trash, the user looks at the file's icon, clicks and holds the gaze button, looks at the trash icon, and release the gaze button. Finally, the user closes all windows by looking at their close boxes and clicking the gaze button. Thus, the task is identical to that with the mouse except that gaze controls the current focus and the gaze button executes a command for the current focus.

We implemented IGO on the Macintosh platform in Macintosh Common LISP. This system interfaces with an IScan (Cambridge, MA) eye tracker to capture gaze data. This eye tracker includes a head-mounted camera and specialized hardware and produces a sampling rate of 60 Hz and an accuracy of approximately 1° of visual angle.

Intelligent Gaze Interpretation

IGO requires some way of interpreting user gazes — that is, mapping gaze points to the items to which the user is likely attending. Although we could incorporate a naive algorithm that maps gazes to the nearest items, this algorithm often fails because of equipment noise and individual variability [9]. Thus, we employ an intelligent probabilistic algorithm that determines the best interpretation based on two criteria: the location of the gaze point and the current task context. The algorithm is somewhat similar in essence to algorithms previously employed in building gaze-based interfaces [10] and in analyzing gaze data from psychological experiments [9]. This body of research has shown that such probabilistic algorithms can interpret gaze data as accurately as human experts in real time [9], thus making them an excellent choice for IGO.

The interpretation algorithm takes a given gaze location g and target items $i \in I$ and returns the item i_{best} that most likely corresponds to g. More formally, the algorithm finds the item i_{best} that maximizes the probability $\Pr(i|g)$:

$$\begin{aligned} i_{best} &= \underset{i \in I}{\operatorname{argmax}} \left[\Pr(i \mid g) \right] \\ &= \underset{i \in I}{\operatorname{argmax}} \left[\frac{\Pr(g \mid i) \cdot \Pr(i)}{\Pr(g)} \right] \\ &= \underset{i \in I}{\operatorname{argmax}} \left[\Pr(g \mid i) \cdot \Pr(i) \right] \end{aligned}$$

The first step in this process involves calculating the probabilities $\Pr(g|i)$ of producing a gaze at g given the intention to attend each item i. The gaze location g simply denotes the estimated point-of-regard coordinates $<x,y>$ as determined by the eye tracker. Each item i can be described as a rectangle $<cx,cy,sx,sy>$, where cx and cy describe the coordinates of the center of the rectangle and sx and sy describe the size of the rectangle as the distance from cx and cy to the edges of the rectangle. To compute $\Pr(g|i)$, we multiply the probability of each coordinate given a Gaussian distribution around the item's rectangle:

$$\Pr(g \mid i) = G(x, cx, sx) \cdot G(y, cy, sy)$$

Here the function $G(x,\mu,\sigma)$ denotes the probability of observing the value x in a Gaussian distribution with mean μ and standard deviation σ.

The second step in the process involves calculating prior probabilities $\Pr(i)$ of attending each item i. We compute these probabilities from the current state of the interface by assigning *prior scores* to various items and then normalizing the scores into probabilities. The various items are assigned scores as follows:

- File menu: 1 if the last action was an open or select, 1/5 otherwise. This models the fact that users tend to use this menu immediately after opening or selecting folders or files.
- Edit menu: 1/5. This models the fact that the menu is unused in the current implementation.
- Special menu: 1 if there are items in the Trash, 1/5 otherwise. This models the fact that users only empty the trash if items have already been thrown away.
- Window close box: 1/5 if the last action selected an icon in the window, 1 otherwise. This models the fact that users typically do not close a window immediately after selecting something within it.
- Window: 1/25. This models the fact that windows are very infrequently the focus of attention, with the exception of dragging an icon into a window.
- Other: 1. This models the default case.

Thus, while items have a default prior score of 1, the score is reduced for items that are unlikely in the current situation. The prior scores are then normalized to produce prior probabilities $\Pr(i)$. While this preliminary design estimated these priors informally in pilot studies of the interface, a more rigorous design could determine better priors empirically by observing long-term behavior in the interface.

Given $\Pr(g|i)$ and $\Pr(i)$, we can determine i_{best} — the i that maximizes $\Pr(i|g)$. However, we would like to give the interface the option of not assigning a gaze to any item if this probability is too low. For this purpose, we ensure that the value $log\ \Pr(i_{best}|g)$ is above a minimum threshold; below threshold, the interface considers there to be no current focus. In pilot testing we found that a threshold of –20 works well for our implementation.

To illustrate the behavior of the gaze interpretation algorithm, let us consider the situation where a gaze point falls near in or around a window close box. Figure 2 shows the assignment of gazes at various points for two possible cases; filled circles represent gaze points assigned to the close box, open circles to the icon, and X's to nothing. In Figure 2(a), the close box and the icon have equal prior scores, and thus are given equal weight in the interpretation algorithm. In this case gaze points are assigned to the nearest item above threshold, which even allows points somewhat far from the close box to be assigned to the close box. In Figure 2(b), the close box has only 1/5 the prior score of the icon (assuming that the last action selected an icon in the window). Here gaze points between the close box and the icon are more likely assigned to the icon, and gaze points farther from the close box are no longer assigned to it. Thus, as the prior probability of the close box decreases, the area in which gaze points are assigned to it shrinks and other assignments become more likely.

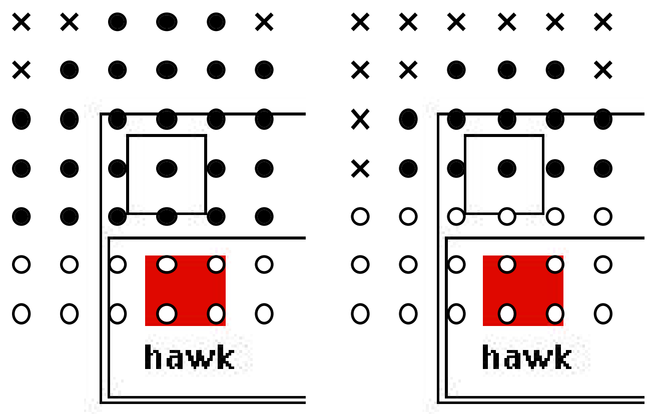

Figure 2: Probabilistic assignment for the close box at various points for (a) the default prior and (b) a smaller prior. Filled circles represent points assigned to the close box, open circles to the icon, and X's to no item.

Discussion

We would like to emphasize three important aspects of IGO. First, we have not removed any functionality from the mouse or other inputs; instead, we have simply added the ability to utilize gaze as a complementary source of input. In contrast, most existing gaze-based interfaces replace the functionality of the mouse and/or keyboard with gaze, typically because the interfaces are intended for physically-disabled people who have difficulty using, or cannot use, other forms of input [e.g., 3, 5]. Some existing systems intended more for able-bodied users add gaze functionality that is always "on" [e.g., 13]. IGO allows for total flexibility in using the gaze functionality as little or as much as desired.

Another important aspect of the interface involves the use of a gaze button to invoke commands. All gaze-based interfaces must deal with the so-called "Midas touch" problem [6]: users focus on many items, some of which are intended for commands and some of which are not. Of the ways of managing this problem (see [6] for a review), the use of a "dwell threshold" is the most common in current interfaces [e.g., 2, 11]. However, gaze buttons, in various forms, are also employed in some interfaces [e.g., 6]. We chose to use a gaze button for two primary reasons: it allows gaze functionality fully analogous to the mouse, and it can be implemented on a keyboard to provide fast and convenient access.

A third important aspect of the interface is its interpretation algorithm, which utilizes a probabilistic model of gaze location and context to assign gazes to a current focus of attention. The vast majority of existing gaze-based interfaces employ a naive method of gaze interpretation, namely mapping gazes to the nearest targets. A few systems employ probabilistic models to choose what commands to present next but not actually to interpret gaze input [e.g., 2]. Only one system incorporates a sophisticated model of user behavior, implemented as a hidden Markov model [10]; however, this system requires a detailed sequential model and a division of continuous input into subsequences for analysis. The probabilistic model presented here represents a balance of the naive and complex probabilistic algorithms, allowing for fast and robust interpretation of continuous input while avoiding the complexities of more fully-specified models.

INTERFACE EVALUATION

Several pilot trials with IGO showed that, with little practice, users could successfully use the system and perform the basic functions with the gaze modality easily and efficiently. To evaluate IGO more quantitatively, we ran a study with several interface tasks and asked users to perform the tasks as quickly and accurately as possible. We had two primary goals in this study. First, we wished to train users on the gaze and mouse modalities separately to determine how their performance improves at various points in the training. Second, we wanted to analyze how users integrate gaze input with mouse input after a period of training in each modality. While more rigorous evaluation would require a long-term study of interface use, our study sheds light on a number of interesting aspects of the system that can guide future development of this and similar systems.

Interface Setup and Tasks

We set up IGO so that users could perform a number of basic tasks. We first specified the file system as a three-level hierarchy of folders and files. The following lists each folder in the system along with its contents; folders are capitalized, files are in lower-case:

- **Zoo**: { **Birds** , **Fish** }
- **Birds**: { **hawk** , **dove** }
- **Fish**: { **minnow** , **trout** }

Given this file hierarchy, we defined five tasks for users to perform. The tasks, along with sample instructions as given to users and the basic actions required in the tasks, are shown in Table 1. For instance, the Move task involves dragging a file from one folder to another; this task comprises five basic actions: two open actions (for the **Zoo** and **Fish** folders), one drag action (from **Fish** to **Birds**), and two close actions. Similarly, the Create task involves creating and naming a new folder; the Duplicate task, duplicate a file through the File menu; the Rename task, selecting and renaming a file; and the Trash task, dragging a file to the trash and emptying the trash contents through the Special menu. All tasks involve one or two open actions, followed by some combination of the drag, select, type, and menu actions, followed by one or two close actions.

Method

Subjects

Ten users (three women and seven men) successfully participated in the experiment. An additional three users participated but were omitted from data analysis because of extreme noise in their eye-tracking data. All participants had at least three years of experience with either the Mac OS or Windows operating system. None of the participants had any prior experience with a gaze-based interface or with eye-tracking equipment.

Table 1: Interface tasks with instructions and actions.

Task	Sample Instructions	Actions
Move	In **Zoo**/ **Fish**/ , Move **trout** to **Zoo**/ **Birds**/ and close all windows.	Open, open, drag, close, close
Create	In **Zoo**/ , Create folder **Dogs** and close all windows.	Open, menu, type, close
Duplicate	In **Zoo**/ **Fish**/ , Duplicate **trout** and close all windows.	Open, open, menu, close, close
Rename	In **Zoo**/ **Birds**/ , Rename **hawk** to **owl** and close all windows.	Open, open, select, type, close, close
Trash	In **Zoo**/ **Birds**/ , Trash **dove**, empty trash, and close all windows.	Open, open, drag, menu, close, close

Materials

The experiment included two stages: a *training stage* and a *free stage.* The training stage comprised eight blocks of 10 trials each. The blocks alternated between *gaze blocks* using gaze input alone and *mouse blocks* using mouse input alone; the starting block type was counterbalanced across users. The free stage comprised two *free blocks* of 10 trials each where users could employ both inputs freely as desired. All blocks included two instances of each of the five tasks in a randomized order.

Procedure

Users were first introduced to the workings of the eye-tracking equipment and the gaze-based interface. After being calibrated on the eye tracker, the users completed five mouse trials followed by five gaze trials with help from the experimenter to become acquainted to the interface and experimental tasks. Finally, they completed the 10 blocks of experimental trials. Comments were gathered from users after the experiment to note their impressions of the ease of the system and any specific strategies they may have utilized.

Each trial comprised two parts: first, the user would read the on-screen instructions and click the gaze or mouse button when the instructions were understood; and second, the user would perform the task and click in a special **Done** region in the lower-left of the screen. Although the instructions remained on the screen in the second part, users were encouraged to use them as little as possible to provide a more accurate estimate of performance.

Results

Training Stage Results

The training stage allowed users to improve their skills with each modality separately. We examine user behavior in this stage beginning with how accurately users performed the given tasks. We utilize two criteria for determining whether a user's task behavior was correct: whether the user's actions include all the necessary actions for the task (as shown in Table 1), and whether there were two or less actions in addition to the necessary actions. Figure 3 shows the percentage of tasks classified as correct for the four blocks in the training stage. (We analyze the free stage in the next section.) Users consistently make more errors in the gaze trials than in the mouse trials and the number of errors remains fairly constant throughout the four blocks. A repeated-measures ANOVA with within-user factors of modality and block confirms these observations, showing a significant effect of modality, $F(1,9)=22.07$, $p<.01$, but no effect of block or their interaction, $p>.5$.

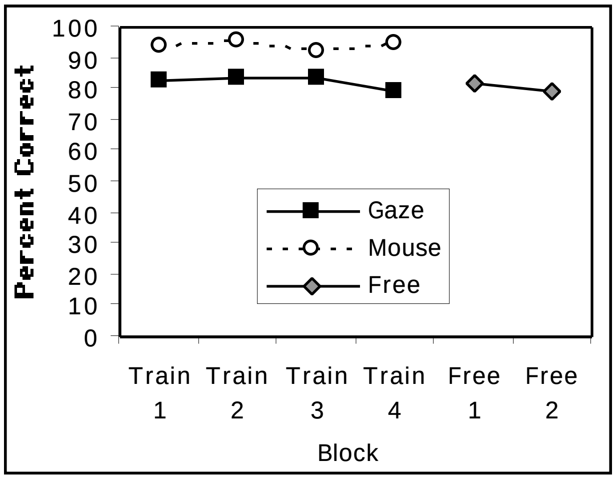

Figure 3: Percent correct across experiment blocks.

User errors with the gaze modality in both the gaze and free trials could largely be attributed to a specific type of error that we call a "leave-before-click" error. This error occurred when the user looked at an item, tried to click the gaze button, but looked away before the button was actually pressed. Ironically, users were *less* prone to commit this error as total novices because they fixated items more deliberately; however, as their confidence in the interface grew, they performed actions faster and became more prone to the error. Further practice helped users to understand the temporal interaction of gazing and clicking and better cope with the problem.

Considering only correct trials, Figure 4 shows the average time needed for subjects to complete a single task in each of the training blocks. For both modalities, users exhibit a nice learning curve, with rapid improvement in the first blocks and more gradual improvement in later blocks. Although the gaze blocks show consistently longer times than the mouse blocks, a repeated-measures ANOVA shows that this difference is not significant, $p>.1$. The effect of block is very significant, $F(1,9)=22.16$, $p<.001$, confirming the learning trend. Although we might expect that, given users' familiarity with the mouse, the gaze learning curve would be steeper than the mouse curve, the modality-block interaction is not significant, $p>.2$ — in other words, users improved at approximately the same rate with both modalities. Thus, much of the learning in the interface seems to have arisen from familiarity with the screen and tasks rather than use of the gaze or mouse input, suggesting that users had little trouble becoming adept with the gaze modality.

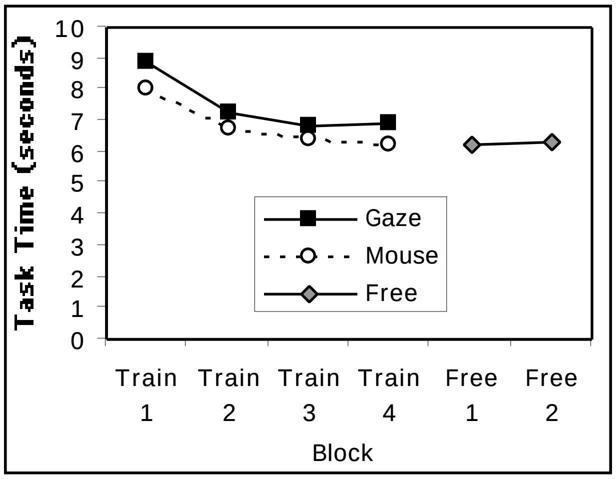

Figure 4: Task times across experiment blocks, in seconds.

Taking this analysis a step further, it is interesting to look at the performance of individual users in the different modalities. Figure 5 shows the average task times for each user and modality in the final two training blocks. Three users (3, 4, 7) required an additional one or more seconds with the gaze modality. However, four users (1, 8, 9, 10) actually exhibited faster times with the gaze modality — even with little practice in this modality and years of practice with the mouse modality. Again we have evidence that users quickly and easily learned to employ the gaze modality in IGO.

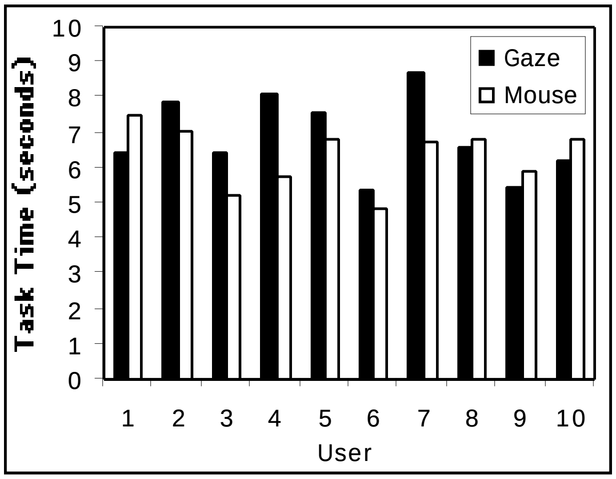

Figure 5: Task times for individual users in the final two training blocks.

We should note that these results comparing gaze and mouse performance in IGO do not necessarily reflect the raw physical performance of the eyes and hands. Rather, the results reflect a complex interaction between raw performance and a number of other factors, including the interface's ability to interpret focus (e.g., using intelligent gaze interpretation), the user's ability to coordinate focus with the gaze/mouse button, and user's ability to adapt to all these factors. In the general discussion, we mention how we can utilize our understanding of raw physical performance to improve the system through detailed cognitive modeling and analysis.

One final important aspect of user's interaction with IGO involves the effect of intelligent interpretation: did the intelligent algorithm have a significant impact on how gazes were interpreted? To answer this question, we re-ran user protocols under two other versions of IGO with simpler methods of gaze interpretation: *basic* interpretation, where gazes directly over a target are assigned to the target and gazes not over a target are assigned to nothing (i.e., the same algorithm used for the mouse); and *no-context* interpretation, where the prior scores are eliminated from the probabilistic intelligent algorithm. We then analyzed correctness (as defined earlier) of user protocols when interpreted by these methods versus the proposed intelligent method. While 83% of the protocols were correct with intelligent interpretation, 65% were correct with no-context interpretation, and only 17% were correct with basic interpretation. Thus, the intelligent algorithm greatly assisted in interpreting user gaze and thus facilitated interaction with the system.

Free Stage Results

The free stage allowed users to employ either modality whenever and however they wished. Overall, users clearly liked the gaze modality, employing it in 67% of all task actions. Figures 2 and 3 include the percent correct and task times, respectively, for the two free blocks. Users' task times in the free blocks were slightly faster than the final two blocks in for mouse and gaze alone. Thus, users were successfully able to integrate the two modalities in terms of overall speed. Users' percent correct in the free blocks was nearest that in the gaze training blocks. This result is due in part to users' extensive use of the gaze modality. Also, users experienced some difficulty in dealing with two foci (i.e., gaze and mouse), causing them to use the wrong button for the intended focus (e.g., using the mouse button when only gaze focus was over an item).

We can also look at how individual users employed the two modalities in the free stage. Figure 6 shows each user's gaze use defined as the percentage of task actions using the gaze modality. A number of users (e.g., 1, 6, 10) employ gaze in a vast majority of actions. Only two users (5, 7) employ gaze less than half the time, and one of these users (7) avoids gaze completely. Users thus exhibited a fair amount of variability in the amount of their gaze use. However, their gaze use was not completely random: Gaze use was closely correlated to the difference between task times for gaze and mouse, *R*=.70; thus, users who performed better with gaze relative to mouse in the training blocks generally preferred gaze in the free blocks. While some of users' gaze use could be attributed to experimenting with the two modalities, it is clear that users appreciated the gaze modality and made good use of it to achieve fast performance.

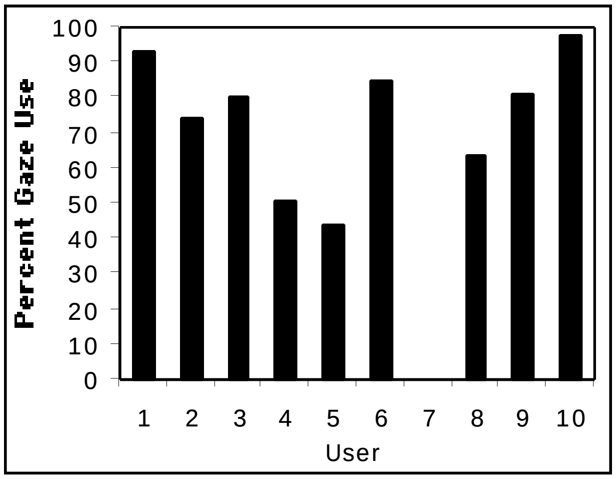

Figure 6: Percent gaze use for individual users in the free stage.

Because the task actions varied in complexity, we can analyze how this complexity affected gaze use. Figure 7 shows average gaze use for each of the five action types. The action types are shown left-to-right in terms of increasing complexity: Select and Close require a single click, Open requires a double-click, Menu requires a drag or two clicks, and Move requires a drag. As the figure shows, users exhibited a clear tendency to favor gaze use for simpler actions over complex actions. Users employed gaze approximately 75% of the time for the simplest actions, Select and Close. They employed gaze approximately 40% of the time for the most complex action, Move, and 50-60% of the time for actions in between, Open and Menu.

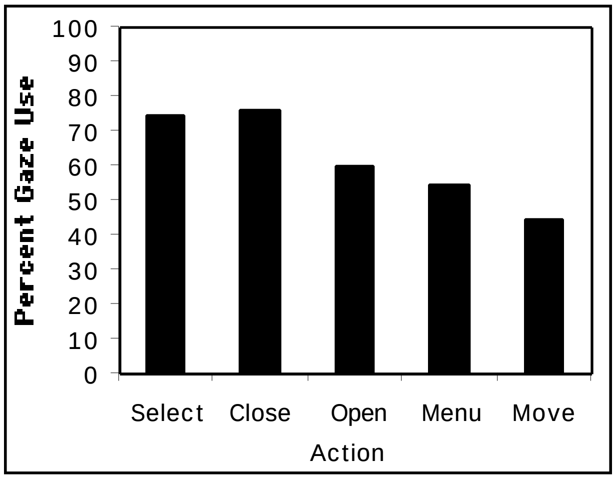

Figure 7: Percent gaze use for the five action types in order of decreasing complexity.

User comments after the experiment and our own qualitative impressions of their behavior manifested a number of interesting aspects of the system. Several users reported specific strategies to employ gaze only for simpler actions and to avoid it for complex actions, particularly dragging: “the main difficulty [with gaze] is the drag-and-drop”; “I didn't like dragging things [with gaze]”. Users often seemed to utilize a strategy in which they used mouse when the mouse pointer was near the focus of the intended action and gaze otherwise; one user explicitly noted this strategy and termed it “selection based on distance”. Users also tended to use gaze after typing on the keyboard to avoid long latencies to shift from keyboard to mouse: “it took a long time to move to the mouse”. In addition, they often employed gaze until it “failed” them (i.e., they made an error), then switched to mouse temporarily, then returned to gaze after a short time. Thus, users derived several interesting strategies both implicitly and explicitly to help them cope with the integration of the gaze and mouse modalities.

GENERAL DISCUSSION

Beyond WIMP Interfaces

The overall success of IGO demonstrates the power of intelligent gaze-added interfaces for everyday applications, such as those that utilize WIMP interfaces. However, the potential for intelligent gaze-added interfaces goes well beyond WIMP interfaces. Jacob et al. [7] and others have outlined a variety of “non-WIMP” interfaces that allow for more flexible interface design and use. These researchers cite a number of features important to non-WIMP interfaces, including multimodal interaction, parallel input streams, and continuous-valued input.

Intelligent gaze-added interfaces offer exciting potential for future non-WIMP interfaces. As IGO and other systems [e.g., 13] demonstrate, users can learn to integrate gaze input with other modalities quickly and naturally. Gaze input is inherently stream-like, and thus provides a range of possibilities for uses in parallel with other input/output streams. In addition, gaze input can be incorporated into a “non-command” interface that acts on implicit rather than explicit user commands [6]. However, like speech and handwriting input, gaze input is not only continuous-valued but also noisy. This fact often makes it difficult to infer user intent, emphasizing the need for intelligent interpretation algorithms such as that in IGO or similar algorithms [e.g., 10].

Importance of Intelligent Gaze Interpretation

IGO, like all gaze-based interfaces, is only as user-friendly as the eye-tracking equipment allows. When we were able to calibrate users accurately on the eye tracker, they reported that the gaze modality felt smooth and seamless. However, when our calibration was somewhat problematic, users reported some amount of difficulty and frustration with the system. Other researchers [e.g., 6] have noted similar experiences with other gaze-based interfaces. In IGO, intelligent gaze interpretation clearly helps a great deal in alleviating these problems with the eye tracker. In the near future, we hope to conduct a detailed study that quantitatively measures the effect of intelligent gaze interpretation on user performance and ease of use.

The problem of interpreting gaze goes beyond the accuracy of eye tracking, however. Even with perfect eye tracking, we cannot know exactly what users are attending to based on the estimated gaze point, since users often view items in the parafovea and periphery (roughly speaking, farther than

1° of visual angle outside the line of sight). This dissociation between gaze and attention makes gaze interpretation significantly more difficult with tighter spacing between items and faster user input [9, 10]. Thus, systems must make use of as much predictive information as possible to maximize the likelihood of correct interpretations. The interpretation algorithm instantiated in IGO demonstrates the power of probabilistic algorithms for building intelligent, accurate, and more user-friendly gaze-based systems.

Gaze-Based Interfaces and Cognitive Modeling

The design and implementation of gaze-based interfaces such as IGO incorporates a number of major and minor design decisions, some of which can be difficult to make. For instance, for IGO, we considered incorporating a lag in which gaze focus on an item would last some time after the gaze has left the item. This change might help to alleviate the "leave-before-click" problem but also might decrease the responsiveness of the system as observed by users. The impact of such design options is often unclear, leaving the system developer to choose between user studies or ad hoc implementation decisions.

As an alternative to these choices, we have started considering how to employ cognitive modeling to improve current gaze-based interfaces. Cognitive modeling provides a rigorous way to express user behavior and test design options without the need for a full-scale user study. CPM-GOMS [8] is one framework that allows for fast, convenient modeling of the "critical paths" in an interface. Such a framework may help to identify and eliminate bottlenecks in the processes required by gaze-based interfaces based on knowledge of characteristics of raw eye and hand performance. ACT-R [1] is another framework that allows for detailed modeling of behavior at the level of keystrokes and eye movements. Using ACT-R, system designers can not only evaluate the times needed for various actions but also predict user learning trends and performance improvement with practice. We hope to soon model user behavior in IGO to attempt to determine how modeling can inform the design of this and similar interfaces.

ACKNOWLEDGMENTS

This work was done at Carnegie Mellon University and was supported in part by Office of Naval Research grant N00014-95-10223 awarded to John R. Anderson. We thank Rob Jacob for helpful comments in the early development of this work.

REFERENCES

1. Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Hillsdale, NJ: Lawrence Erlbaum Associates.
2. Frey, L. A., White, K. P., & Hutchinson, T. E. (1990). Eye-gaze word processing. *IEEE Transactions on Systems, Man, and Cybernetics*, *20*, 944-950.
3. 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.
4. 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 Publishing.
5. 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.
6. 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.
7. Jacob, R. J. K., Deligiannidis, L., & Morrison, S. (1999). A software model and specification language for non-WIMP user interfaces. *ACM Transactions on Computer-Human Interaction*, *6*, 1-46.
8. John, B. E. (1990). Extensions of GOMS analyses to expert performance requiring perception of dynamic visual and auditory information. In *Proceedings of CHI 90* (pp. 107-115). New York: ACM Press.
9. Salvucci, D. D. (1999). *Mapping eye movements to cognitive processes*. Doctoral Dissertation, Department of Computer Science, Carnegie Mellon University.
10. Salvucci, D. D. (1999). Inferring intent in eye-movement interfaces: Tracing user actions with process models. In *Human Factors in Computing Systems: CHI 99 Conference Proceedings* (pp. 254-261). New York: ACM Press.
11. 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.
12. 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.
13. Zhai, S., Morimoto, C., & Ihde, S. (1999). Manual and gaze input cascaded (MAGIC) pointing. In *Human Factors in Computing Systems: CHI 99 Conference Proceedings* (pp. 246-253). New York: ACM Press.