Eye-S: a Full-Screen Input Modality for Pure Eye-based Communication

Marco Porta[*] Matteo Turina[†]

Dipartimento di Informatica e Sistemistica
Università di Pavia

Abstract

To date, several eye input methods have been developed, which, however, are usually designed for specific purposes (e.g. typing) and require dedicated graphical interfaces. In this paper we present *Eye-S*, a system that allows general input to be provided to the computer through a pure eye-based approach. Thanks to the "eye graffiti" communication style adopted, the technique can be used both for writing and for generating other kinds of commands. In Eye-S, letters and general eye gestures are created through sequences of fixations on nine areas of the screen, which we call *hotspots*. Being usually not visible, such sensitive regions do not interfere with other applications, that can therefore exploit all the available display space.

CR Categories: H.1.2 [Models and Principles]: User/Machine Systems—Human Factors; H.5.2 [Information Interfaces and Presentation]: User Interfaces—Input Devices and Strategies, Interaction Styles

Keywords: gaze interaction, eye writing, eye typing, eye gesture, eye sequence, assistive technology, alternative communication

1 Introduction

Effective pure communication through eyes has been one of the most stimulating challenges since the appearance of reliable and relatively unobtrusive eye trackers. People who are partially or completely paralyzed, and cannot even speak (e.g. quadriplegic persons), are usually able to freely move the eyes, which therefore become their only communication channel.

Eye writing is probably the most intuitive way to convey nontrivial messages to someone: using the natural language to express any concept, also complex thoughts, the "distance" between severely impaired persons and the world they live in can be greatly reduced. The problem of writing through the eyes has been widely considered in the past, and a number of solutions have been proposed.

[*]e-mail: marco.porta@unipv.it
[†]e-mail: turinamatteo@libero.it

ETRA 2008, Savannah, Georgia, March 26–28, 2008.
© 2008 ACM 978-1-59593-982-1/08/0003 $5.00

The simplest approach is surely that based on on-screen keyboards and dwell time: if the user looks at a certain key for more than a predefined time interval, the key is considered pressed and the corresponding letter is typed. Several studies have been carried out connected to this kind of typing and related issues (e.g. [Hansen et al. 2003] and [Majaranta et al. 2003, 2004], focused on the efficacy of dwell time and the importance of feedback). There are also variants where the dwell time is substituted with some kind of switches, such as physical buttons, eye blinking (e.g. [Rasmusson et al. 1999]) or facial muscle activation (e.g. [Surakka et al. 2004]). Moreover, automatic word-completion functionalities are often used to reduce the typing effort (e.g. [Lankford 2000]).

Approaches based on virtual keyboards, however, suffer from a main drawback: fixations may not be exactly centered on the keys, due to both the inaccuracy of the eye tracker (for most commercially-available devices the precision is about 1°) and the fact that users can correctly encode information in the fovea even if they are not looking exactly at the target. Unless keys are very big, this problem may result in a frustrating typing experience, characterized by frequent errors and consequent "undo" operations. Solutions have been proposed to partially solve the problem, such as, for example, fixation tracing [Salvucci 1999]: using hidden Markov models, user actions can be mapped to sequential predictions of cognitive process models, so that unintentional fixations on wrong keys can be discarded. Unfortunately, another relevant shortcoming of virtual keyboards is that they usually occupy large parts of the screen — otherwise keys would be too small to be correctly selected — and this reduces the available space for the display of other content.

While the "traditional" keyboard is the most intuitive answer to the writing problem, several alternative approaches have been proposed, with the aim to overcome its limitations. For instance, Isokoski [2000] describes some interesting solutions based on off-screen targets, placed around the monitor outside the display area, which is therefore completely free for applications. Ward and MacKay [2002] present a method for text entry based on inverse arithmetic coding, where writing is seen as a navigational task. Like in the *Dasher* general data entry interface [Ward et al. 2000], from which the input technique is derived, letters are arranged vertically on the screen, and, as they are looked at, enlarge into big areas that in turn include the most probable successive letters to form a word. Among the several selection and input methods described by Majaranta and Räihä [2002], the technique used in the VisionKey system is quite original, albeit, probably, not very intuitive. Letters are displayed in a grid of cells, and each cell contains four characters. To start a command, the user glances at a corner of the key chart and then glances back at the center. Afterward, proper sequences of glances at one of the four corners, at the center of the grid and at the cell containing the letter to type produce the desired selection. To limit the possible fatigue deriving from dwell time constraints, Ohno and Mukawa [2003] propose the Quick Glance Selection Method (QGSM), where each "button" displayed on the screen (including character keys) is composed of two parts, a command name area and a selection area: to press the button, the user has to look at the selection area. The system by Miniotas et al. [2003] uses a technique, called Symbol Creator, based on assembling characters by means of seven letter "segments", implemented as eye-gaze activated on-screen keys that resemble basic elements of Latin cursive. Fejtová et al. [2006] describe a system called IPad, where the screen is subdivided into "boxes" and the user selects the required letter by gradual division of a selected interval of ordered characters. The choice is made by looking anywhere within two big buttons placed on the left and right upper areas of the screen – the requirement for precise positioning is thus significantly reduced at the expense of a slower writing speed. Such "principle" (grouping several letters under one key) has a long tradition in eye input interfaces, and can be also found in early systems based on electro-oculographic potential (e.g. EagleEyes [Gips and Olivieri 1996]). Wobbrock at al. [2007] describe a system for "eye writing" which uses gestures similar to hand-printed letters, where characters are "drawn" instead of typed. Urbina and Huckauf [2007], at last, present three new input methods characterized by original features. The first method exploits the principle of "pie menus" (when slices containing groups of letters are looked at by the user, new pies are generated which include only one letter per slice); the second technique is based on character buttons arranged around a rectangular area, where the text is actually written; in the third method, finally, letters are arranged on a half-circle in the upper part of the monitor, and are typed by dragging them into a text field.

In this paper we present *Eye-S*, an input modality for pure eye communication which can be used for both writing and providing general commands to the computer. Unlike the just quoted systems, Eye-S does not require the display of any graphical element, therefore not disturbing other applications.

The article is structured as follows: Section 2 discusses the main motivations which have inspired our work; Section 3 explains the Eye-S input modality; Section 4 describes the major features of the system; Section 5 considers possible alternative uses for Eye-S; Section 6 presents experimental results; Section 7, at last, draws some conclusions and provides hints for future work..

2 Motivations Behind Eye-S

As said in the Introduction, Isokoski [2000] considers several approaches to the eye writing problem, all exploiting off-screen targets. Practically, to leave the display area totally free for applications, physical elements (e.g. strips of paper) are placed outside the screen, and used to control the input process. The Minimal Device Independent Text Input Method (MDITIM), in particular, is a technique which uses five tokens for input. Four of this tokens are mapped to the four main directions — north, south, west and east — to give the characters a two dimensional interpretation. The method was initially conceived for pen input: for instance, the 'a' letter is defined by an upward movement of the pen followed by a shift to the lower left position, sequence which can be abbreviated with NSW (North South-West). All characters are defined this way, with proper combinations of strokes (see the examples in Figure 1, where starting points are indicated by circles).

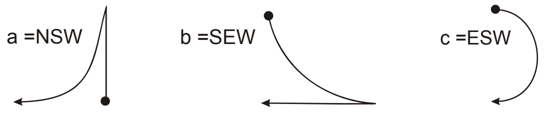


Figure 1: Examples of MDITIM characters

The principle of MDITIM has been transposed to an eye writing application where four external targets, corresponding to the principal directions, are attached to the four sides of a monitor, and a fifth target, placed in the upper left corner of the screen, is used as a modifier (e.g. to input uppercase characters). To "create" a letter, the user has to look at the targets according to the right succession (for example, upper, lower and left target for 'a').

While original, such an approach has unfortunately two drawbacks. Firstly, it requires physical targets to be positioned outside the display area, which might be a problem if the employed eye tracker is not able to precisely follow off-screen gazes. Secondly, the MDITIM coding technique requires the user to learn gaze sequences that rarely resemble the shapes of characters, thus turning out not to be very intuitive.

In the context of action-based gaze input (for example to perform drag and drop operations), Milekic [2003] introduces the concept of "eye graffiti", where gaze gestures are used to form a vocabulary in a way similar to the text input mechanism used in personal organizers. In these devices, natural input is obtained by "drawing" letters, or parts of them, through a pen. As shown in Figure 2, the sketched lines resemble very closely the corresponding characters, and this helps the user to easily learn the input modality.

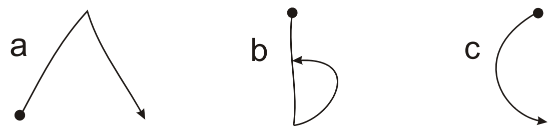


Figure 2: Examples of "letter drawing" in personal organizers

Our work derives basically from the previous considerations. External targets used to free the screen area may be troublesome for certain eye trackers, and an eye-based input method should be as intuitive as possible, to reduce the user's cognitive load. We therefore apply a sort of eye graffiti input approach to on-screen virtual targets, which, while being in well-defined positions, are usually not explicitly displayed.

Another approach based on the graffiti principle, which however could not inspire our work as it is extremely recent, is the one presented by Wobbrock at al. [2007] (quoted in the Introduction). Their interesting system, called EyeWrite, exploits the EdgeWrite unistroke alphabet [Wobbrock et al. 2003], originally created to enable text entry on PDAs and other devices. Essentially, letters are composed by looking at the corners of a square window specifically used for input, according to the EdgeWrite alphabet; when the letter under construction is completely formed, the user looks at the center of the input window to confirm his or her choice. Apart from the hint provided by Milekic [2003] within a general discussion about eye gestures, the paper by Wobbrock at al. [2007] on EyeWrite is almost certainly the first article to describe a letter-like gestural writing application.

However, while probably originating from the same basic idea, *Eye-S*, the system we describe in this paper, differs from Eye-Write in various aspects. On the one hand, Eye-S exploits all the available screen area for input, by means of an invisible yet implicitly identifiable grid of targets uniformly distributed. On the other hand, Eye-S does not require the display of any window or input interface, thus leaving the screen totally free for applications. Moreover, the use of nine targets to compose letters (or create other general commands) makes it possible to choose among many "eye stroke" combinations, thus not constraining eye gestures to a specific and relatively limited alphabet.

3 Eye-S Input Modality

Even if our input method can be used to provide general commands to the computer (as it will be explained in Section 5), it was primarily conceived for writing. Thus, our inspiring stimulus was the possibility for the user to easily "draw" letters through the eyes, without the need for on-screen space-demanding keyboards.

In Eye-S, letters are drawn through sequences of eye fixations on specific parts of the screen, which we call *hotspots*. We use nine hotspots, four positioned at the vertices of the screen, four placed in the middle of each side and one located in the center (Figure 3). Practically, a hotspot is a square area whose size can be varied according to one's preference and/or to the precision of the eye tracker.

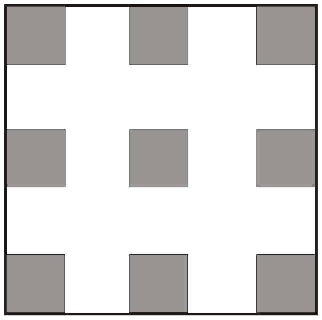


Figure 3: Hotspot positions within the display area

Normally, hotspots are *not* visible. However, their position is quite clear, and almost no effort is required to the user to remember where they are. This means that the screen is totally available for displaying any content, and no area is occupied by specific interfaces for text input.

An *eye sequence* (from which the name *Eye-S* stems) is a succession of fixations on the hotspots, while a *sequence segment* is the linear path between two consecutive fixations; in the following, we will sometimes use the term "eye gesture" with the same meaning as "eye sequence". When the user looks at a hotspot for more than a defined threshold (e.g. 400 milliseconds), a *sequence recognition process* starts. If both the following conditions are satisfied:

1. other hotspots are looked at after the initial one, within configurable time intervals
2. the succession of watched hotspots pertains to a set of predefined sequences stored in a "database"

then a corresponding action is performed. If the system is being used for text input, the action will be the same as typing a key on a keyboard. Eye sequences can be chosen arbitrarily, but in the writing context they will of course resemble the form of letters. For instance, possible eye sequences for the 'a' and 'b' characters are shown in Figure 4.

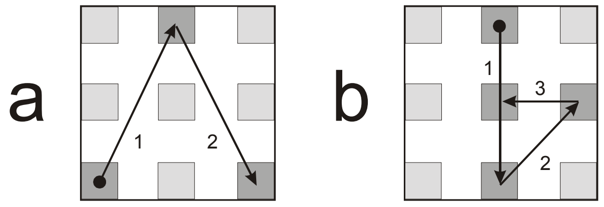


Figure 4: Possible eye sequences for the 'a' and 'b' letters

As it can be seen, the 'a' letter is obtained by watching at first the bottom-left hotspot, then the top-middle hotspot and, at last, the bottom-right hotspot. The sequence for the 'b' letter is instead top-middle, bottom-middle, right-middle and center. Through this 9-points fixation schema, a complete alphabet can be easily built.

Figure 5 shows the most part of the alphabet we have defined and used in our tests. It is important to note, however, that more than one sequence could be associated with the same letter (or sign or general action), as there may be different intuitive ways to describe the letter as a succession of fixations on the hotspots. Moreover, to ease the user's task and simplify the recognition process, in choosing the sequences we have adopted the following criteria:

- no sequence is longer than 3 or shorter than 2 segments (that is, at least 2 and no more than 4 different hotspots are involved in the coding of letters and other writing commands);
- no sequence is a subset of another.

4 System Description

Eye-S is implemented in C# within the .NET Microsoft framework. As an eye tracker, we use the Tobii 1750 [Tobii Technology AB 2003], which integrates all its components (camera, infrared lighting, etc.) into a 17'' monitor.

Allowed sequences and their associated characters or commands are defined in a configuration text file, where each line contains: (1) a sequence code, (2) a possible ASCII code (for characters) or other code referring to a certain action, and (3) a "description" (i.e. a comment). For example, the following is a portion of the configuration file defined for our alphabet:

```
020000103    97    "a"
010043020    98    "b"
.........    ..    ...
102000304    122   "z"
000302000    32    "space"
200000003    8     "backspace"
020013040    63    "?"
.........    ..    ...
```

The sequence code is formed of 9 digits, as many as the number

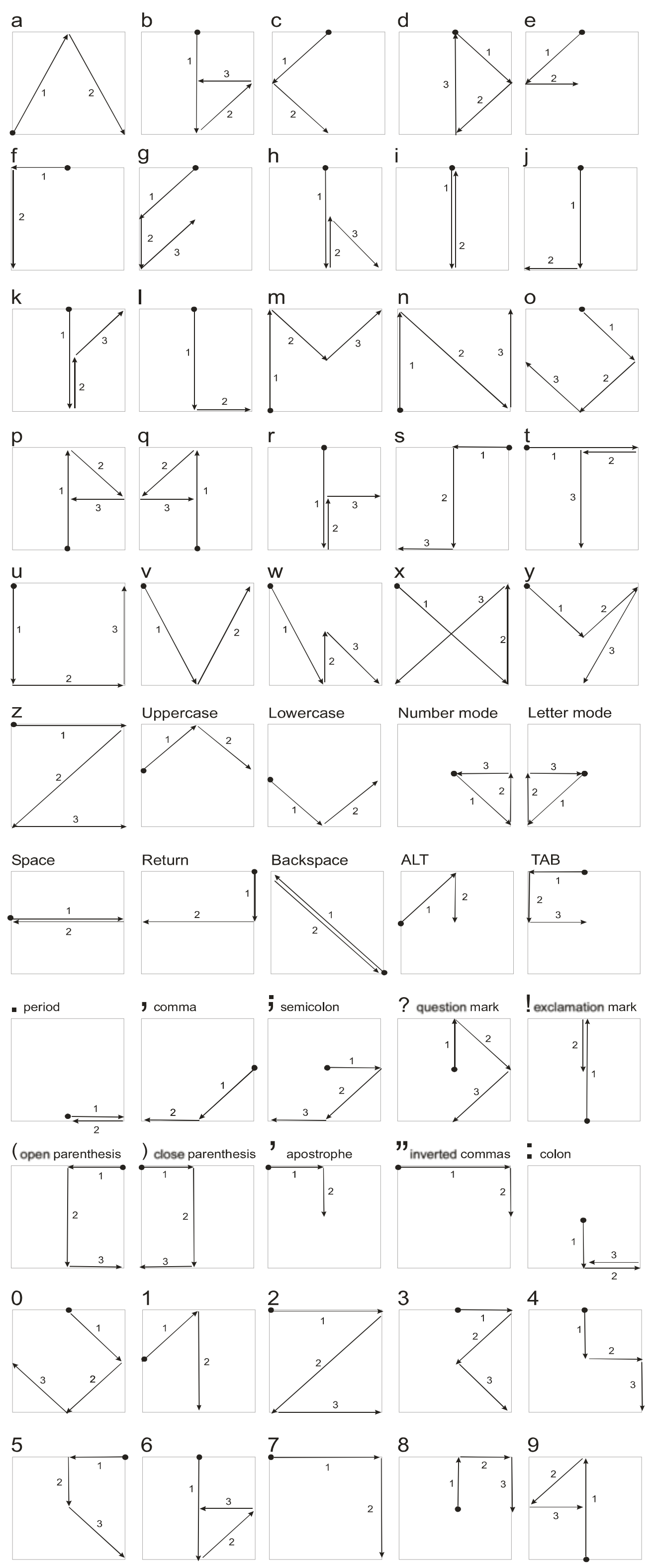
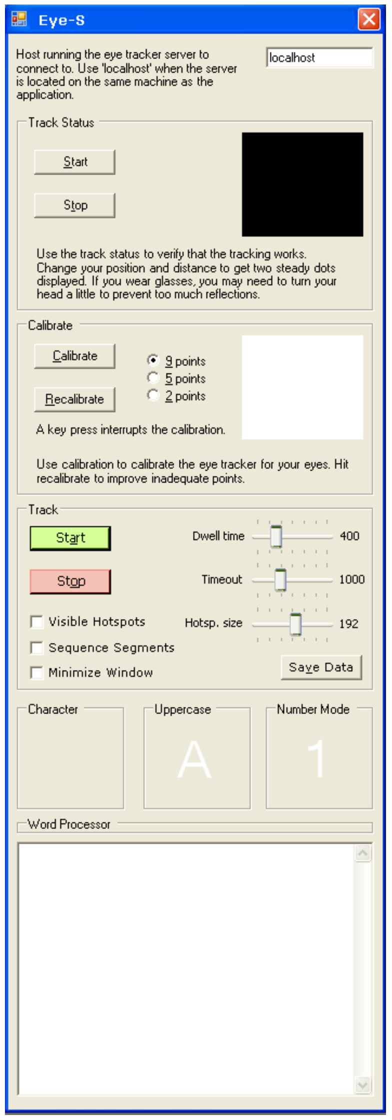


Figure 5: A possible alphabet for Eye-S text input

of hotspots. Hotspots on each of the three rows, from left to right, correspond, respectively, to the digits in positions 1 to 3, 4 to 6 and 7 to 9. The set of values that each digit can assume depends on the maximum length of the sequence, i.e. on the number of segments composing it. If a hotspot is not involved in a certain sequence, then the value of its digit is 0. The first hotspot that must be looked at to start a sequence will have its digit set to 1, the second hotspot will have the digit set to 2, the third set to 3, and so on (when more than one fixation is required on the same hotspot, some number may be missing in the progression, like for 'space' and 'backspace' in the previous example). Since in our alphabet the maximum length of a sequence is three — four hotspots involved — the digits of the sequence code can assume values between 0 and 4 only.

When Eye-S is started in "configuration/training" mode, the interface shown in Figure 6 is displayed.


Eye-S

Host running the eye tracker server to connect to. Use 'localhost' when the server is located on the same machine as the application. localhost

Track Status

Start

Stop

Use the track status to verify that the tracking works. Change your position and distance to get two steady dots displayed. If you wear glasses, you may need to turn your head a little to prevent too much reflections.

Calibrate

Calibrate

Recalibrate

[✓ 9 points] 5 points 2 points

A key press interrupts the calibration.

Use calibration to calibrate the eye tracker for your eyes. Hit recalibrate to improve inadequate points.

Track

Start

Stop

Dwell time 400

Timeout 1000

Hotsp. size 192

☐ Visible Hotspots

☐ Sequence Segments

☐ Minimize Window

Save Data

Character

Uppercase A

Number Mode 1

Word Processor

Figure 6: "Configuration/training mode" interface for Eye-S

In this functioning mode, it is possible to calibrate the eye tracker and set some parameters. For instance, it is possible to choose whether hotspots must be explicitly shown on-screen with a semi-transparent effect, which may be useful at the very beginning of the interaction with the system, to clearly understand where sensitive areas are (Figure 7).

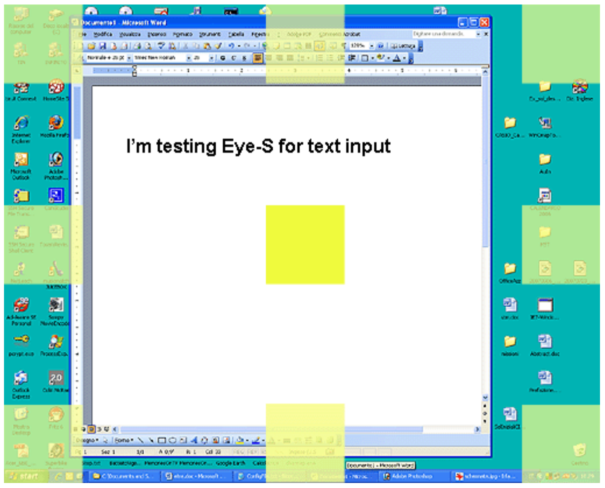


Figure 7: Explicit display of hotspots

Moreover, the size of hotspots and the length of dwell and timeout times can be varied through sliders. Such settings, as well as calibration data, can be stored for subsequent use. After this initial configuration phase, Eye-S can be started and either directly tested for text input through a built-in word processor or minimized for free interaction with all the available screen space.

When Eye-S is launched in "implicit mode", no interface is shown, and the system is ready for use as an input method (according to previously stored data). Both in this case and when Eye-S is started in configuration/training mode with a minimized interface, any application involving text input can be used to write by looking at the hotspots. Eye sequences corresponding to letters, in fact, are translated into keyboard events, and thus the current active window will receive the input. In this regard, it is interesting to note that the focus among open windows can be easily changed by means of a proper sequence, which simulates the ALT+ESC key combination (Figure 8a).

Since it may be useful for the user to be able to turn Eye-S on and off at any time, we use a specific sequence with this precise purpose. Actually, after starting the system in one of the two functioning modes, no sequence is recognized, apart from the "on/off" sequence (which, in our implementation, is the one shown in Figure 8b). As the user performs such eye gesture, Eye-S is really turned on (and the message "Eye-S ON" appears at the center of the screen). When, for any reason, the user wants to turn the recognition procedure off, he or she simply performs the same gesture (the message "Eye-S OFF" will appear).

During system use, it is generally helpful to get a feedback about the sequence composition process. To this purpose, when the user looks at the first hotspot for more than the defined dwell time, a small green square is displayed within the hotspot itself. Such square contains a '1', to indicate that this is the first hotspot of a possible sequence. If the user looks at another hotspot within a timeout, then a yellow square appears, with a '2' written inside it (and the green square is deleted). If the sequence which is being recognized is three segments long, the same happens for the third hotspot (orange square and '3' as a sequence indicator). At last, on the final hotspot of a sequence — whether it is three or four segments long — a red square is displayed which contains the character or "action" recognized, so that the user can immediately understand that the eye gesture has been successfully detected.

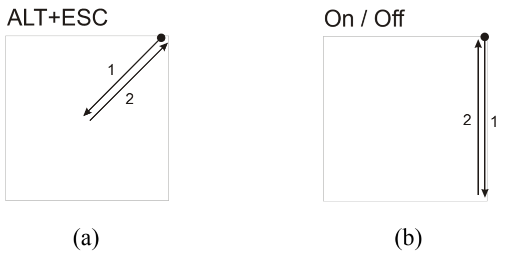


Figure 8: (a) Sequence for ALT+ESC (switch among open windows); (b) Sequence for turning Eye-S ON and OFF

Figure 9 graphically exemplifies this feedback mechanism in the case of the 'p' letter.

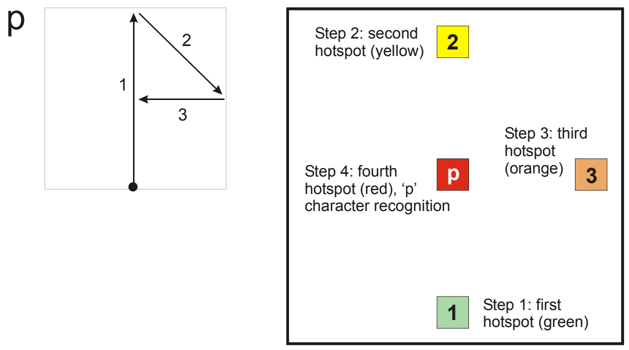

p
1
2
3
Step 2: second hotspot (yellow)
2
Step 3: third hotspot (orange)
3
Step 4: fourth hotspot (red), 'p' character recognition
p
1
Step 1: first hotspot (green)

Figure 9: Example of feedback provided by Eye-S during the composition of letter 'p'

If, at any point of the sequence composition process, the timeout is reached, the recognition progression is reset and the feedback square currently displayed is deleted; the user can then create a new sequence beginning from any of the nine hotspots. As a further feedback, the user can also decide to display straight lines connecting the hotspots, according to the sequence which is being created. While not strictly necessary and potentially disturbing, such additional indication may be useful when learning the alphabet of sequences, which are graphically visualized on the screen as they are composed.

5 Possible Alternative Uses of Eye-S

As emphasized in the previous sections, although we have initially conceived Eye-S as a system for text input, it can be also used to provide the computer with general commands. For instance, there may be sequences associated with all the main applications that the user normally uses, which could be started by simply looking at the hotspots in the correct order. Moreover, in a certain program, predefined sequences might produce specific results.

Even within the writing context we have explicitly considered in this paper, however, the potential of Eye-S for the interaction with general applications is quite high. Most (if not all) Windows interfaces can in fact accept keyboard input through proper key combinations (shortcuts). Thus, providing for sequences corresponding to such combinations makes it possible to easily control applications. Even more generally, still considering the Microsoft Windows environment, the ALT key can be exploited to select menus: when the key is pressed, the upper-left menu is highlighted (e.g. the 'File' menu in applications of the MS Office suite). Each menu has also a shortcut letter that can be used to immediately open it (e.g. 'F' for 'File', 'E' for 'Edit, etc.). Within menus, in turn, each item has a shortcut letter that immediately triggers its action. Eye-S allows the user to easily perform all these operations by means of sequences corresponding to the ALT key and to the shortcut letters. At any moment, the ESC sequence can be used to cancel current selections.

As an example, consider the following possible interaction with Word 2003:

1. the user wants to open an existing file; therefore, he or she performs the ALT sequence (to activate menus), the 'F' sequence (to open the 'File' menu) and the 'O' sequence (to trigger the 'Open' action);
2. in the 'Open' dialog box, the user can now directly write, in the text field, the name of the file to open or, before that, navigate within the file system structure. To do this, the TAB sequence is repeated until the file display area gets the focus; afterward, the arrow keys sequences (UP, DOWN, LEFT and RIGHT) can be used to select a folder or file, and then to open it (through the ENTER sequence);
3. once the document has been opened, the user goes to a specific page by repeating the PAGE DOWN sequence, and then reaches a precise line and character by means of the arrow sequences (the HOME and END sequences can be also exploited to position the cursor at the beginning or at the end of the current line);
4. the character placed before the cursor is deleted by means of the BACKSPACE sequence, and then substituted with another character (of course entered through its sequence);
5. after moving to another section of the document, the user opens the 'Format' menu and chooses 'Paragraph' (ALT, 'O' and 'P' sequences);
6. within the dialog box, the user specifies the properties for the paragraph, moving among input fields through the TAB and SHIFT+TAB sequences, selecting items within drop-down menus by means of the UP and DOWN arrow sequences, entering number values in text fields, and so on; at the end, the 'Ok' button is selected and pressed (ENTER sequence);
7. the user proceeds with other actions…

The just described sample (and simple) scenario is only one of the many that could be identified. In addition, as already stated, MS Office applications have hundreds of more specific shortcuts (e.g. CTRL+S for 'Save', CTRL+F for 'Find', etc.), which could be considered in this well-defined context. If necessary, different configuration files with different sequence associations (and possibly different alphabets) could be even defined, to be used with different programs.

It is also interesting to note that if an application does not provide explicit shortcuts for menus, its interface can be anyway navigated. In fact, once the ALT sequence has been performed, the user can move among the available menus by means of the RIGHT and LEFT arrow sequences; next, after choosing a specific menu, the UP and DOWN arrow sequences allow items within menus to be first selected, and then activated (through the ENTER sequence). Also, the ALT+SPACE and 'n' sequences, executed one after the other, minimize the current window. In general, when no program is being executed and the screen is free, the TAB and SHIFT+TAB sequences allow to move among the desktop, the 'Start' button, and the application bar; within each area, the arrow sequences can then be used to choose and activate specific icons.

Therefore, while Eye-S is configurable for specific purposes, it can be profitably used as a general interaction mechanism as well, even when its actions are simply standard keyboard events.

6 Experiments

Eye-S has been of course informally tested many times during its development. However, once fully implemented, we have carried out more formal experiments aimed at obtaining useful qualitative and quantitative data about its use.

Needless to say, whatever the method adopted, eye input requires a certain training to produce really good results. Even with input techniques which are conceptually close to our usual input modalities, such as those exploiting on-screen virtual keyboards, inexperienced users needs some time to get acquainted with the new form of communication. Using an eye graffiti approach like the one implied by Eye-S requires even further training time, necessary to correctly memorize sequences; and this may prevent very fast input rates to be achieved in a very short time.

Unfortunately, to date we have not had the opportunity to properly train many testers (only two), and hence, although the results obtained from them are quite good, we do not have really relevant statistical data about character input speeds. However, we have also carried out experiments with 8 totally novice users, to investigate how rapid the approach can be in the very initial stages of the interaction and get "first impression" assessments.

The group of 8 volunteer testers, who had never used an eye tracking system before, was composed of 4 males and 4 females, aged between 24 and 38 (28 on average). Before the actual test, each user was explained the purpose of Eye-S and of the experiment, and, after the calibration of the eye tracker, could freely practice with the system for ten minutes. A scheme with sequence associations for each alphabet letter (the one shown in Figure 5) was provided to all participants. System settings, maintained both in this preliminary phase and in the real tests, were the following:

- dwell time (minimum fixation time for a hotspot to be taken as a starting point of a sequence): 400 ms;
- timeout (maximum time elapsed between two fixations on two hotspots during sequence composition, for the sequence not to be discarded): 1000 ms;
- hotspot size (side length): 190 pixels;
- visible hotspots (semitransparent effect): yes;
- visible sequence segments (segments connecting hotspots explicitly drawn during sequence composition): no.

Given the very short training period, we opted for displaying hotspots all the time. After the ten minutes of practice, users were asked to write the sentence: "hello! I am writing with my eyes."

(which could be read on a sheet near the Tobii monitor). We measured the following data:

- time taken to accomplish the task (expressed in seconds);
- number of wrong characters (*w1*, wrong sequences created, for example because not correctly remembered);
- number of characters "not finished" (*w2*), i.e. whose sequences were started but then interrupted for any reason (typically because timeouts were reached).

Table 1 shows the results obtained.

Table 1: Results of the experiments with totally novice users

	T1	*T2*	*T3*	*T4*	*T5*	*T6*	*T7*	*T8*	*Avg*
time	222	238	164	122	221	195	209	137	188.5
w1	3	4	2	2	1	0	3	1	2
w2	4	8	2	1	6	5	4	1	3.9

Although the times taken to write the sentence are rather high for all the testers, all of them concluded the task successfully. In point of fact, considering that they could practice only for very few minutes and did not have any previous experience with eye tracking systems, we think that the results are comfortable (on average, we have noted that about one third of the time was spent looking at the scheme of sequences; once correctly learnt, the total time is greatly reduced). As a qualitative judgment, all the users appreciated very much the originality of the input approach.

Besides the eight inexperienced users, as stated, also two expert testers were involved in more accurate experiments, aimed at assessing the real potential of Eye-S as a technique for text input. "Expert" means that they had been using the system for about two hours and a half before the test (in previous days). Even though the results achieved cannot be statistically relevant, we are very confident that they closely reflect more accurate data we will obtain from future tests carried out with more testers.

To compare these results with those provided by inexperienced users, also in this case the testers had to write the sentence "hello! I am writing with my eyes.". The only differences in system settings for these experiments were the dwell time of 340 ms (instead of 400) and the fact that hotspots were not visible. Table 2 shows the outcomes of the trials.

Table 2: Results of the tests with experienced users

	T1	*T2*	*Avg*
time	62	67	64.5
w1	0	0	0
w2	0	1	0.5

As shown in Figure 10, the performance in this case is quite different from that of novice testers.

In a separate and longer experiment, the two experienced users had also to write the text of a news item taken from a newspaper, which was dictated to them. The text was composed of 1547 characters (including spaces and punctuation). On average, the testers were able to write 34 characters per minute, i.e. 6.8 WPM if we consider an average length of 5 characters per word. Although this speed is lower than those cited in the literature for on-screen key-

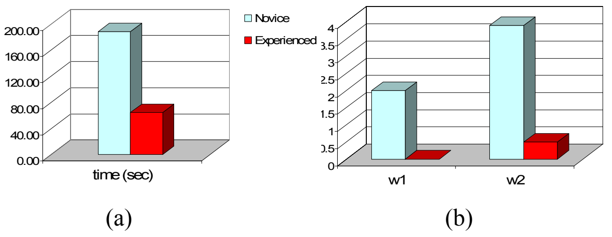


Figure 10: Novice vs. experienced users — (a) Average times to accomplish the task; (b) Average values for *w1* and *w2*

boards (e.g. 9.89 WPM reported by Majaranta et al. [2004]) and for some other input approaches (e.g. 7.99 WPM stated by Wobbrock et al. [2007] for EyeWrite), we think that Eye-S should be considered in its totality, with both pros and cons. In particular, the advantage of leaving the screen completely free for applications, thus not interfering in any way with them, is relevant to us. Moreover, while for now we have tested Eye-S as a text input method only, the system can be used as a general input approach for almost any kind of command.

7 Conclusions and Future Work

In this paper we have presented Eye-S, a system for providing eye input to the computer.

The main feature characterizing Eye-S is surely the fact that it does not require any graphical interface, thus leaving all the available display area free for applications. On-screen virtual keyboards and other kinds of eye-based input techniques, in fact, are usually space-demanding, and may interfere with the use of programs. We think that this advantage can compensate for a little slower writing rate when compared to other text input systems, which use visible graphical elements as targets.

In the context of text input, however, we will need to carry out more accurate tests, with more experienced users. In addition, while the adopted sequences for letters, punctuation, etc., derive from our personal preferences (we judge them good approximations of character shapes), more accurate choices might improve the writing speed. To this purpose, we will also consider word completion functionalities.

Although the main application scenario we have considered in this paper is that of text input, certainly one of the most useful and studied, it is important to stress again that the approach can be exploited to specify general commands as well. The meanings associated with sequences can in fact be always chosen according to the needs of the particular context. Moreover, as explained in Section 5, even when used as a text input tool, Eye-S can allow easy interaction with many of Windows' functionalities, which can be a great advantage.

Acknowledgements

This work has been supported by funds from the Italian FIRB project "Software and Communication Platforms for High-Performance Collaborative Grid" (grant RBIN043TKY).

References

FEJTOVÁ, M., NOVÁK, P., FEJT, J., and ŠTĚPÁNKOVÁ, O. 2006. When can eyes make up for hands? In *Proceedings of the 2nd Conference on Communication by Gaze Interaction (COGAIN 2006)*, Turin, Italy, September 4-5, 46-49.

GIPS, J., and OLIVIERI, P. 1996. EagleEyes: An Eye Control System for Persons with Disabilities. In *Proceedings of the Eleventh International Conference on Technology and Persons with Disabilities*, Los Angeles, California, USA, March.

HANSEN, J. P., JOHANSEN, A. S., HANSEN, D. W., ITOH, K., and MASHINO, S. 2003. Command Without a Click: Dwell Time Typing by Mouse and Gaze Selections. In *Proceedings of INTERACT 2003*, Zürich, Switzerland, September 1-5.

ISOKOSKI, P. 2000. Text Input Methods for Eye Trackers Using Off-Screen Targets. In *Proceedings of ETRA 2000*, Palm Beach Gardens, FL, USA, 15-21.

LANKFORD, C. 2000. Effective Eye-Gaze Input into Windows. In *Proceedings of ETRA 2000*, Palm Beach Gardens, FL, USA, November 6-8, 23-27.

MAJARANTA, P., and RÄIHÄ, K. 2002. Twenty Years of Eye Typing: Systems and Design Issues. In *Proceedings of ETRA 2002*, New Orleans, Lousiana, USA, 15-22.

MAJARANTA, P., MACKENZIE, I. S., and RÄIHÄ, K. 2003. Using motion to guide the focus of gaze during eye typing. In *Proceedings of 12th European Conference on Eye Movements (ECEM12)*, Dundee, Scotland, August.

MAJARANTA, P., AULA, A., and RÄIHÄ, K. 2004. Effects of Feedback on Eye Typing with a Short Dwell Time. In *Proceedings of ETRA 2004*, San Antonio, Texas, USA, March 22-24, 139-146.

MILEKIC, S. 2003. The More You Look the More You Get: Intention-Based Interface Using Gaze Tracking. In *Proceedings of the 7th Annual Museum and the Web Conference*, Charlotte, North Carolina, USA, March 19-22.

MINIOTAS, D., SPAKOV, O., and EVREINOV, G. 2003. Symbol Creator: An Alternative Eye-based Text Entry Technique with Low Demand for Screen Space. In *Proceedings of INTERACT '03*, M. Rauterberg et al. (Eds.), IOS Press, IFIP, 137-143.

OHNO, T., and MUKAWA, N. 2003. Gaze-Based Interaction for Anyone, Anytime. In *Proceedings of HCI International 2003*, Crete, Greece, June 22-27, Vol. 4, 1452-1456.

RASMUSSON, D., CHAPPELL, R., and TREGO, M. 1999. Quick Glance: Eye Tracking Access to the Windows95 Operating Environment. In *Proceedings of the 14th International Conference on Technology and Persons with Disabilities*, Los Angeles, CA, USA, March 15-20.

SALVUCCI, D. 1999. Inferring Intent in Eye-Based Interfaces: Tracing Eye Movements with Process Models. In *Proceedings of CHI '99 (Conference on Human Factors in Computing Systems)*, New York, ACM Press, 254-261.

SURAKKA, V., ILLI, M., and ISOKOSKI, P. 2004. Gazing and frowning as a new human-computer interaction technique. *ACM Transactions on Applied Perception*, Vol. 1, Issue 1 (July), 40-56.

TOBII Technology AB 2003. Tobii 1750 Eye-tracker (Release B), November '03.

URBINA, M. H., and HUCKAUF, A. 2007. Dwell Time Free Eye Typing Approaches. In *Proceedings of the 3rd Conference on Communication by Gaze Interaction (COGAIN 2007)*, Leicester, UK, September 3-4, 65-70.

WARD, D. J., BLACKWELL, A. F., and MACKEY, D. J. C. 2000. Dasher - a Data Entry Interface Using Continuous Gestures and Language Models. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology (UIST 2000)*, San Diego, CA, USA, November 5-8.

WARD, D. J., and MACKAY, J. C. 2002. Fast Hands-free Writing by Gaze Direction. *Nature 418*, August 22, 838.

WOBBROCK, J. O., MYERS, B. A., and KEMBEL, J. A. 2003. Edge-Write: A Stylus-Based Text Entry Method Designed for High Accuracy and Stability of Motion. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology (UIST '03)*, Vancouver, BC, Canada, November 2-5, 61-70.

WOBBROCK, J. O., RUBINSTEIN, J., SAWYER, M., and DUCHOWSKI, A. T. 2007. Not Typing but Writing: Eye-based Text Entry Using Letter-like Gestures. In *Proceedings of the 3rd Conference on Communication by Gaze Interaction (COGAIN 2007)*, Leicester, UK, September 3-4, 61-64.