Providing motor impaired users with access to standard Graphical User Interface (GUI) software via eye-based interaction

Howell Owen Istance[1], Christian Spinner[2] and Peter Alan Howarth[3]

[1,2]Department of Computer Science, De Montfort University,
The Gateway, Leicester, LE1 9BH England, U.K.

[3]Department of Human Sciences, Loughborough University
Loughborough, Leicestershire, LE11 3TU England, UK

[1]*hoi@dmu.ac.uk,* [3]*p.a.howarth@lboro.ac.uk*

ABSTRACT

We have designed an on-screen keyboard, operated by eye-gaze, for use by motor-impaired users. It enables interaction with unmodified standard Graphic User Interface (GUI) software written for able-bodied users, and it is not solely designed around the need to enter text. The keyboard will adapt automatically to the application context by, for example, loading a specific set of keys designed for use with particular menus whenever a menu is displayed in the target application. Results of initial evaluation trials are presented and the implications for improvements in design are discussed.

Keywords: eye-control, visual keyboard, physically-challenged, disability, handicapped

1. INTRODUCTION

A practical way to enable people with various forms of motor impairment to work efficiently in office environments is to provide the means whereby they can use the same software tools at the workplace as their able-bodied colleagues. This requires building interaction devices which allow control over the range of software available and which ideally are usable by people who have different motor impairment.

Eye-based interaction is attractive in this context for a number of reasons. First, it offers the prospect of reducing learning time by providing a 'natural' means of pointing at a displayed object on-screen. Second, moving the eye is fast and positioning a pointer on a required object can be done quickly. Third and perhaps most important, users with severe degrees of motor impairment generally retain good ocular motor control, and so devices based on eye-movement may be used by a larger range of users with motor impairments than devices relying on other muscle groups.

These advantages have, however, to be offset against the known problems of eye-based interaction. These include the level of accuracy with which eye position can be measured, the degree of fine control that the user can exercise over eye movement and the need to be able to disengage eye-control whenever the user wishes to look at the screen without issuing commands. To overcome these problems, eye-based control may be combined with other input modalities, such as speech, so that eye-gaze is used for pointer positioning only and other modes of input are used to make selection actions (equivalent to those normally made by the mouse button). However, at this stage we have restricted ourselves to the use of the eyes alone for both cursor control and selection, rather than investigating combinations of different modalities. Previous work (Istance and Howarth, 1994) investigated the use of eye-gaze for emulation of a mouse to interact directly with standard Graphic User Interface (GUI) applications, and it was concluded then that an indirect 'soft' control device offered a means of overcoming many of the problems associated with the direct eye-based interaction approach. Software-generated virtual keyboards that are displayed on a display screen (which are synonymously termed either an "on-screen keyboard" or a "visual keyboard") offer a solution to these problems, and then because there is no need to invoke additional input modalities, one does not have to rely upon the user having any form of motor control other than over their eyes alone.

The idea of a visual keyboard displayed on a screen and operated by some external device is certainly not new. Indeed, visual keyboards have been developed in the past for use with a variety of interaction devices, such as joysticks and mice, as well as eye-gaze control. Some of these have emulated normal keyboards, but in doing so have been restricted almost entirely to text entry. These have not allowed control over the variety of objects such as those

 ISBN 0 7049 1140 X

contained in menus and dialogue boxes found in modern direct manipulation GUI software. There are many applications, such as Web browsers, which have very limited requirements for text entry but instead require interaction with displayed documents, such as scrolling or clicking on links, as well as interaction with menu items. Other visual keyboards have been developed to control specific software applications only, for example communication and control systems, and can not be used as general purpose input devices.

2. FUNCTIONS AND DESIGNS OF EXISTING VIRTUAL KEYBOARDS

2.1 Single Application Keyboards

EyeScan (Eulenberg, 1985), BlinkWriter (Murphy et al., 1993), ERICA (Hutchinson et al., 1989; White et al., 1993) and EyeTracker (Friedman et al., 1981) are examples of applications which support eye-gaze controlled text entry.

Early text entry devices for physically-challenged users arranged letters according to their frequency of occurrence in text and allowed indirect selection by stepping through a matrix (e.g. MAVIS system in Schofield, 1981; HandiWriter in Ten Kate et al., 1980). This technique has the advantage of accommodating a large set of characters and other symbols within a single template. Demasco and McCoy (1992) proposed a virtual keyboard model which, besides scanning single characters, also supports scanning of words. This technique is unnecessarily restrictive for eye-based pointing and is better suited to devices reliant on more limited forms of motor control. Nevertheless, laying out character keys on the basis of frequency of use is an important design consideration. As selecting larger targets with an eyetracker is easier than selecting smaller targets, frequently used keys can be made larger within the keypad (static sizing) or they can be expanded temporarily when the pointer moves within the key (dynamic sizing), as well as being positioned appropriately.

Operating a visual keyboard by eye is comparable to the one-finger approach of a novice user who also delays each keystroke after the position of a desired key has been recognised. It is a highly sequential task as text must be entered character-by-character with forced delays between the 'eyestrokes'. Consequently, text entry becomes tedious and keystroke savings through a word prediction system not only have the potential to improve speed but also to reduce errors and user fatigue. Koester and Levine (1994) examined user performance in text entry tasks with word prediction by considering the trade-off between the number of keystrokes and the additional cognitive and perceptual loads imposed by having to select from the presented word choices. They concluded that the cognitive cost of presenting a set of word choices, and explicitly selecting one, largely eliminated the performance advantage of keystroke savings. However, their results depended very much on the degree of disability and the chosen method of controlling the keyboard. For instance, there may be a different trade-off when using a mouthstick for typing on conventional keyboard than in the case of a visual keyboard, where the input and output channel is the same. Indeed, the absence of the need to switch between external keyboard and screen suggest that there will be a lower cognitive load when a visual keyboard is in use.

2.2 General Purpose Keyboard Emulators

EyeTyper, developed by Friedman et al. (1984) was an eye-gaze controlled keyboard which could be used in place of the standard keyboard of an IBM-PC. Hence, it was transparent to the host computer's software but its architecture was based on a single keypad template which did not allow more than a very simplistic keyboard emulation.

More recently, the visual keyboard developed by Bishop and Myers (1993) at the University of Iowa supports control by an eyetracker device. The arrangement of character keys is advantageous and groups frequently-used characters towards the centre of the screen. In our experience, a discrepancy between visible cursor position and actual eye position is more likely in the screen corners, and the keys and button sizes of their visual keyboard are arranged to take this into account. However, the support for editing text is quite limited. Key combinations to highlight text, such as *<Ctrl+Shift+Right>* are not possible, and whereas characters are available for text input, they are not available for command selection (e.g. *<Alt+F>*). Thus it is not possible to make use of the rich functionality of a word processor like MS-Word, thereby loosing the advantage of services such as spell-checking.

3. DESIGNING THE VISUAL KEYBOARD

3.1 Customising the Keyboard

Given that the application for which the keyboard cannot be specified in advance, it is clearly important to provide end-user customisation, which can be provided through:

- providing suitable configurable defaults for the keyboard (e.g. dwell time settings)

- providing an off-line keypad editor to enable new keypads to be created which are suited to different applications
- providing the option to create keys which invoke macros. For example, a keypad intended for use with Netscape might contain a key to display a particular page, and this would invoke a macro with the keystrokes necessary to display the URL prompt and enter the characters defining the data.
- self-adaptation of visual keyboard to the current application context (e.g. automatically loading a menu keypad)

3.2 Summary of Design Requirements

In summary, there are a number of requirements that a visual keyboard controlled by eye needs to satisfy:

- it should support both keyboard and mouse emulation
- it should support effective interaction with GUI components such as the scrollable lists, text fields and buttons found with dialogue boxes and not be solely designed around the need for text entry
- it should provide mechanisms to compensate for the inaccuracy inherent in eye-based control
- it should enable the user to customise the device to suit individual preferences concerning tasks within specific applications, but should allow the user to switch between different applications without the need for device reconfiguration.

4. OPERATION OF THE KEYBOARD

4.1 System Architecture

The eyetracker provides raw data on the positions of the left and right eye to the host machine (Istance and Howarth 1994). This data is processed completely separately from the visual keyboard. The software responsible for this uses the current gaze position to move the mouse pointer. The visual keyboard runs as an application, and is sensitive to the position of the mouse pointer. If the mouse pointer remains within the area of the key for a specified time (dwell time), a Windows event corresponding to the key is generated. In this way, it is possible to set different dwell times for different types of key. The visual keyboard is thus completely separate from the eyetracker system providing the data, and although we have concentrated on eye-control, the keyboard could perfectly well be used with other types of input device, such as a joystick or mouse.

4.2 Overall Design of the Visual Keyboard

The keyboard is comprised of three sections as shown in Figure 1. The left section contains the general keyboard system commands menu, the centre section contains the variable keypad, and the right section contains the keypad selection menu.

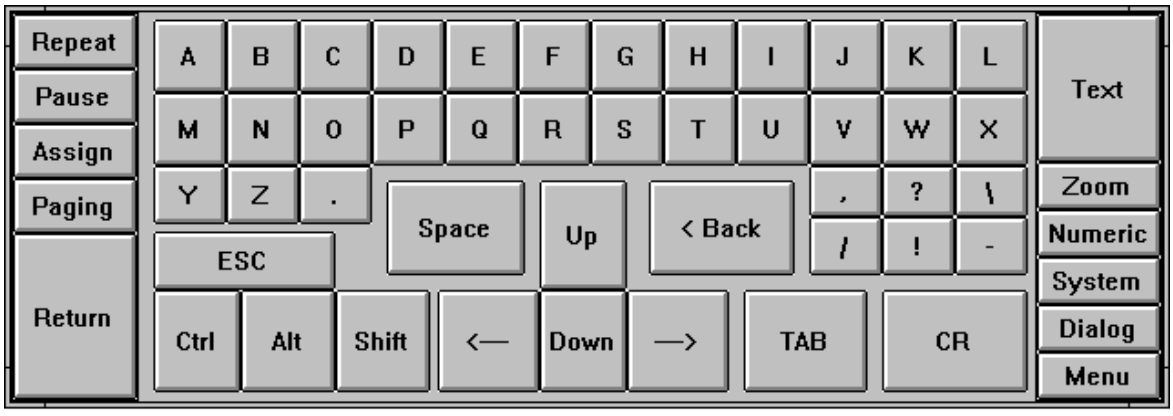

Figure 1. *Visual keyboard with text keypad loaded*

In the keypad selection menu and the system command menu, each key expands when the pointer first moves into it and reverts to its original size when the pointer moves over another key. When the cursor moves off either menu, the last key expanded remains enlarged. This makes dwelling with the eye within the key area easier without the penalty of taking up window space for keys which are not being used (the same principle as a pull-down menu). Additional

keypads may be created in the keypad editor and added to the menu of standard keypads or may replace them in the menu.

4.3 Keypad Design

4.3.1 Text Keypad. The current text keypad (shown in Figure 1) incorporates two important design decisions. First, it is based on an alphabetical arrangement of characters (although this could be replaced by a Dvorak or QWERTY arrangement. Quill and Biers (1993) recommend a QWERTY arrangement, but this is rejected for this prototype because the physically-challenged user is not likely to be familiar with it. Second, the key arrangement gives prominence to the <cursor control> keys due to their relative importance not only during text entry (see also Gould et al. (1985)) but also during text editing.

4.3.2 Dialogue Keypad. The dialogue keypad shown in Figure 2 contains the keys necessary to control dialogue boxes. There are fewer keys here than in the text keypad and therefore the individual keys are larger. The main keys are the <cursor control> keys, the <next item> and the <previous item> keys (corresponding to <tab> and <shift-tab> respectively) and the <escape> key.

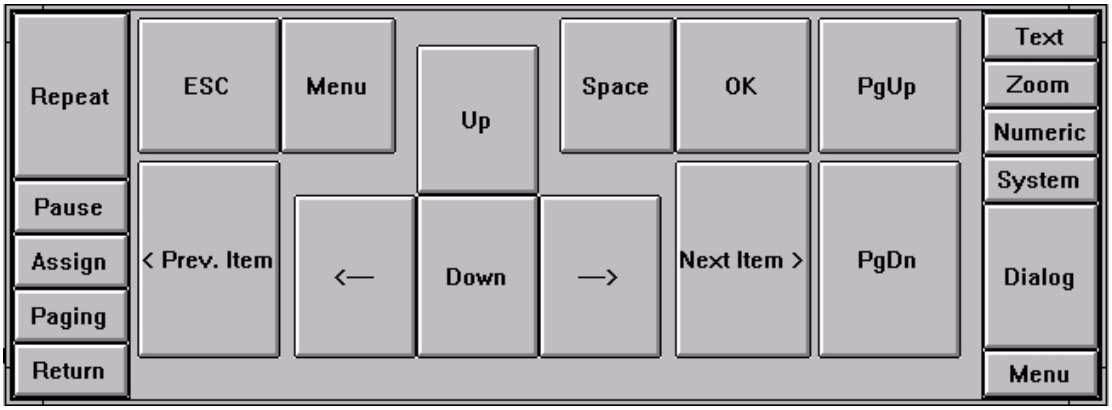

Figure 2*: Dialogue Keypad*

4.3.3 Menu Keypad. The *menu keypad* (not shown) is another context-sensitive keypad for interaction with the menu system. The main keys here are the <cursor control> keys, the <menu> key (corresponding to the <alt> key), the <escape> key and a key which contains the text of the currently highlighted menu item. Selecting this key selects the menu item (and is equivalent to pressing the <carriage return> key).

4.3.4 Zoompad. Many users may have difficulty in precisely positioning the cursor, and so a zoompad has been incorporated (Figure 3). The zoompad shown here emulates mouse commands. It incorporates the equivalent mouse command selection by means of radio buttons. The user looks at a region of the client window and after a brief dwell interval has expired, the region is copied and enlarged into the keypad. The user effects a click action within the zoom area and the event is sent to the corresponding part of the client window.

4.4 Overriding Automatic Keypad Selection

The visual keyboard automatically loads and displays keypads appropriate to the current task context as, for example, if the user accesses a menu when a dialogue box is displayed in the client application. However, this self-adaptation mechanism is not always desirable. For instance, if the first control object in the dialogue box is a text element, the user may prefer the text keypad for the initial activation. In such cases, the user can make explicit assignments using the *<Assign>* button in the left hand part of the keyboard system command menu. Subsequently, the assigned keypad will be loaded automatically when the client application context is the same.

Proc. 1st Euro. Conf. Disability, Virtual Reality & Assoc. Tech., Maidenhead, UK, 1996

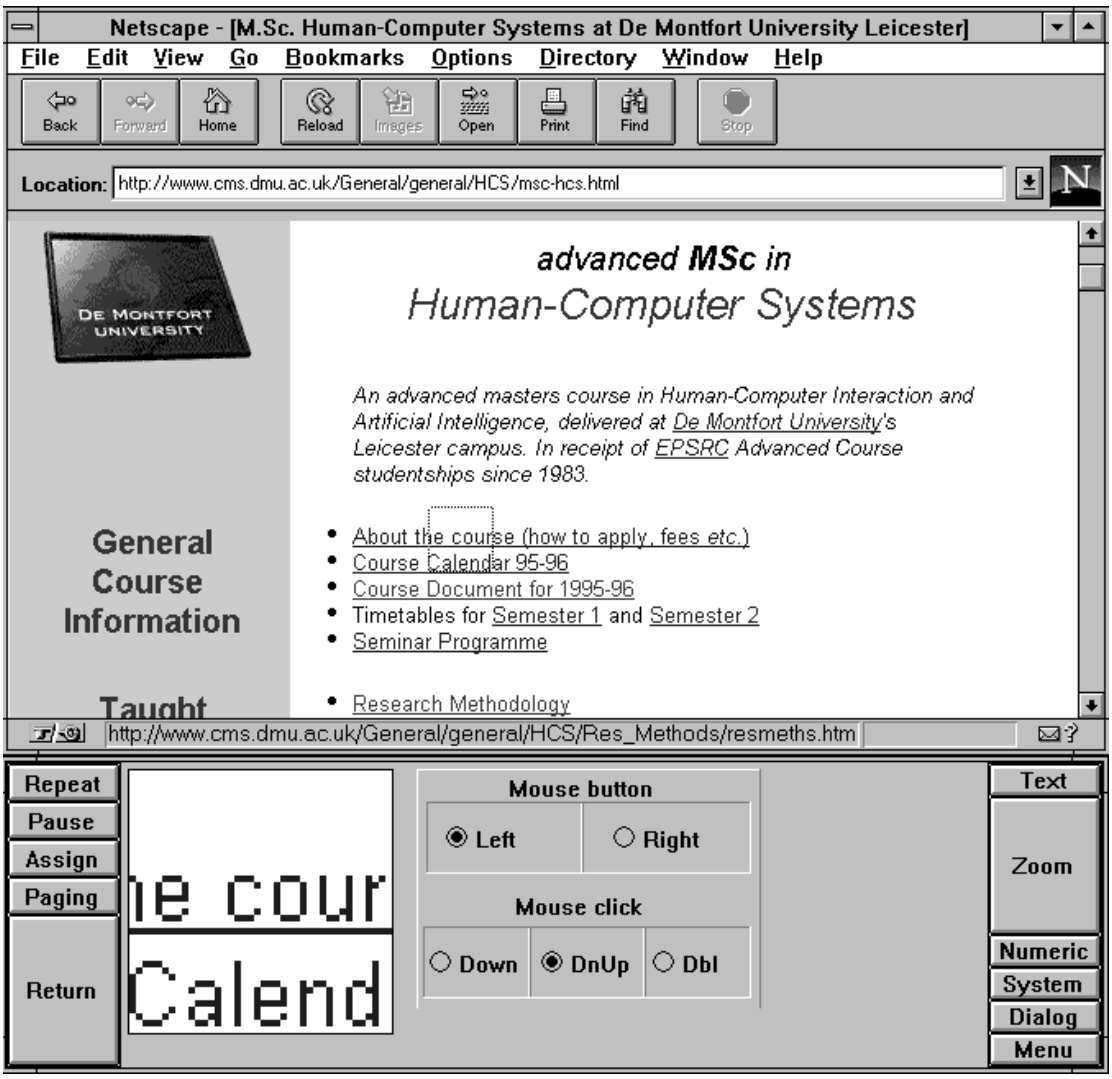

Figure 3: *Zoompad used together with Netscape*

***4.5** Object Dwell Time*

Dwell times for different types of keys can be set individually and these can be altered by the user to reflect personal preference. The values in table 1 show the different settings used as initial values in the evaluation trials (described in section 5). The user is warned about expiration of the dwell time by a change in the cursor. The mouse pointer is represented by a circular cursor within the keyboard and this changes to show a black spot in its centre just prior to the end of the dwell interval. If the user does not wish to select the key, they may look away at this point and selection is then inhibited.

Table 1: *Dwell time settings*

Dwell object	**Dwell time**
<*Cursor control*> keys	1000 ms
<*Enter*> key (menu keypad only)	2500 ms
System command menu	1000 ms
Keypad selection menu	500 ms
Target selection (*zoompad*)	2000 ms
Any other key	1500 ms

4.6 Window Arrangement

The window arrangement of the client application and visual keyboard is supported by three different approaches. In combination, these attempt to compensate for the fact that the on-screen keyboard has to take up a finite part of the available screen area, thereby reducing the area available for the application.

- *Paging:* In the case that the window of a client applications is partially overlapped by the visual keyboard or even requires the whole screen, the <paging> command moves the client application window so that either its top half or bottom half is displayed in the area of the screen above the keyboard window.
- *Heading:* Windows and in particular dialogue boxes which appear initially in the centre of the screen will be automatically moved to the top of the screen if there is overlap. In the case that the remaining screen space is too small, the <paging> command will be selected.
- *Bounding:* Re-sizing the visual keyboard will automatically re-size the client application window.

5. EVALUATION TRIALS

This section reports results from initial evaluation trials with a modern word processor (MS-Word 6.0). More evaluation work remains to be done, both with this application and with other types of application. The trials do, however, give a good indication of the success, or otherwise, of some of the design ideas that have been included in the keyboard. The input device was the eye-control system described previously (Istance and Howarth 1994).

5.1 Selection of Tasks

A set of five tasks, which constituted an integrated exercise, was completed by each subject:

1. Run application and open text document
2. Enter a few lines of text (an address)
3. Save the file
4. Edit the existing text
5. Require help about a specific problem

The tasks, and keystroke level actions required to execute them, are transferable to most word processors, and one objective of the evaluation trials was to judge whether the visual keyboard was effective enough to perform the specified tasks. In addition, in order to evaluate the efficiency with which the tasks were performed the effort required for the user to correct errors, or to edit text, served as a useful performance indicator.

5.2 Results and Discussion

All five (able-bodied) subjects were able to complete the tasks which were given to them. Moreover, all were able to recover from errors and consequently this usability objective of an effective visual keyboard was met in this context. The mean time spent on the completion of all tasks was 19 minutes per subject whereas the total time, including recalibration and repetition of tasks due to eyetracker problems, required on average 38 minutes. The task completion time of the whole task sequence did not vary across subjects by more than three minutes either side of the mean.

5.2.1 Text Entry and Feedback. On average, subjects were able to enter their address in about seven minutes (task 2) which corresponds to a text entry rate of only one word per minute. This is very inefficient and it was observed that most errors occurred during this task. Subjects frequently unintentionally entered a character twice and consequently the time spent on correcting those errors had a major impact on text entry efficiency. The reasons for this lay partly in the techniques used by subjects to get feedback, partly due to a lack of training, and partly due to a delay in updating cursor position.

Two subjects looked for feedback in the text document after each character was entered whereas the other subjects entered a sequence of characters before checking. When the subject's task completion times were compared, the first approach was found to be more efficient. This was because errors were recognised earlier and consequently the subjects required less cursor control keystrokes to return to the site of the error. However, one might expect reductions in key location and selection times in the second approach as the user's attention remained on the keyboard for longer periods of time. There is clearly room for improvement in the rate of text entry, however even the present rate may be acceptable if text entry is limited to a few letters, such as when entering a file name or a help item for which to search.

5.2.2 Interacting with Dialogue Boxes. During tasks 1 and 5, (see section 5.1) navigation problems occurred when interacting with dialogue boxes. In some cases, subjects mistook the control object with the input focus and so manipulated interface elements by keystrokes which were actually designated for a different object. The visual

Proc. 1st Euro. Conf. Disability, Virtual Reality & Assoc. Tech., Maidenhead, UK, 1996

feedback showing the current input focus was not always clear. These problems are caused by a lack of feedback when moving the input focus from one control to another. When an ordinary keyboard is used, it is possible to press a key and observe changes in the visual appearance of a control simultaneously. Using the eyes to 'press' a virtual key prevents the user from observing the effects of the command as it is taking place. However, during these tasks, unintentional key presses occurred far less frequently than with the text entry task, as one might expect given that the dialogue keypad had fewer, and therefore larger, keys.

5.2.3 Combinations of Keystrokes. The fourth task incorporated the use of shortcut keys for stepping from one word to the next in order to select and highlight text. Most subjects had no difficulties in combining two or three keys and hence were able to move the caret with shortcut keys rather than by a series of cursor control keystrokes.

5.2.4 Dwell as a Means of Activating Commands. Problems associated with interference between mouse events generated directly by the eyetracker software and similar events generated by the visual keyboard were apparent. These arose on occasions when subjects were reading a document or browsing a dialogue box. While subjects were aware that there is some form of response when dwelling too long within the visual keyboard area, the possibility of an event happening inside the client window was not apparent. Whilst it would be possible to disable all event generation in the client window by the eyetracking software, there is a case for letting the user interact directly with the client application (by the eyetracking software sending events directly to it) as well as by using the visual keyboard. The advantage lies in greater flexibility and not always having to use the keyboard on-screen. The disadvantage lies in the possibility of generating unwanted events in the client window.

The feedback provided by the change in the cursor, which was intended to warn that dwell time was about to expire, was generally misinterpreted by subjects who thought that the change signified that the key had been pressed.

5.2.5 Feedback on Modifier Key States All subjects were observed to have difficulties in remembering the current state of the modifier keys, for example, forgetting that the <*Shift*> key was locked, and this issue needs addressing. If another keypad was selected after locking the modifier key, then feedback indicating that a subsequent keystroke would lead to a key combination was lost. Errors were also made even though the keypad remained visible, but the effect of the locked key was not obvious. For example, moving the carat with the cursor control keys to correct a typing error with the <Shift> key still locked resulted in the text becoming highlighted rather than the carat simply being moved.

5.2.6 Mechanisms for Changing Keypads. All subjects were able to operate the menu-based keypad selection mechanism, although frequently this required more than one attempt. This is acceptable because the keypad selection is based on a short dwell time (500 ms). Once the desired keypad button had been "acquired" subjects could easily keep the pointer within that button because of its large size. The use of expanding, 'fish-eye' buttons has been shown to be particularly successful for this type of eye-based interaction. Furthermore, the self-adaptation by the visual keyboard by loading the appropriate keypad automatically was also successful in reducing user input.

5.4.7 User Preferences for Interaction Styles with the Visual Keyboard. In general, the most preferred means of command selection was the use of the zoompad to select the smart icons of the toolbar, followed by menu item selection. The least preferred option was using shortcut keys and accelerators. Preferences appeared to depend on the expertise with MS-Word. For instance, one subject reported frequently using shortcut keys for text editing and preferred to use the keypad supporting use of the shortcut keys. A follow-up trial where all tasks, excepting the text entry task, were performed using the zoompad showed considerable increases in speed.

6. CONCLUSIONS

This work has demonstrated how a visual keyboard controlled by eye can be used to interact with standard software produced for the able-bodied user and thus allow the physically-challenged user to benefit from the wealth of software produced for modern GUI environments. This includes being able to access the Internet using existing browsers without the need for any modification either to the browser or to the keyboard.

Furthermore, many GUI applications require precise pointing ability which is often lacking in eye-controlled systems (Istance and Howarth, 1994) and the visual keyboard is capable of assisting with these tasks. Adequate solutions for non-keyboard sensitive GUI objects have been considered problematic in the past (Shein et al., 1991, 1992). The zoompad has been shown to provide an effective solution here.

Further work is required to improve the text entry rate achievable with this keyboard. Part of the problem here lies with the eyetracking system and its associated data processing software, rather than with the keyboard itself. A major problem was the need to correct unintentional keystrokes rather than the time required to locate and press an individual key. The next phase in the development will look closely at the causes of this and will examine means of improving text input rates. In addition, means of editing and interacting with existing text using the visual keyboard will be studied

more closely. At the workplace, the visual keyboard is perhaps more likely to used for editing existing documents than for original document creation.

Future work with the visual keyboard will focus on the issue of feedback and examine ways of overcoming the visual separation between the keyboard and the region of client window from which the user receives feedback about the effects of commands. Additionally, it is intended to examine how direct eye-based interaction with the client application could be integrated with the use of the visual keyboard.

The major conclusion of this project is that the visual keyboard can be considered as a valuable low-cost enhancement to the eyetracker with the capability to compensate for its limitations as a pointing device. It has demonstrated that effective interaction with standard modern software applications using eye-based interaction techniques is entirely possible. This will greatly enhance the possibilities for physically-challenged users to work with the same software products and on a more equitable basis with their able-bodied colleagues.

7. REFERENCES

Bishop, J.B. and Myers G.A. (1993) Development of an effective computer interface for persons with mobility impairment. *Proceedings of the Annual Conference on Engineering in Medicine and Biology,* **15 (3)** 1266-1267.

Demasco, P.W. and McKoy, K.F. (1992) Generating Text From Compressed Input: An Intelligent Interface for People with Severe Motor Impairments. C*ommunications of the ACM*, **35 (5)** 68-77.

Eulenberg, J.B., King, M.T. and Patterson, H. (1985) Eye-Controlled Communication. *Proceedings of Speech Technology '85 - Voice Input and Output Applications,* 210-211.

Friedman, M.B., Kiliany, G. and Dzmura, M. (1985) An Eye Gaze Controlled Keyboard. *Proceedings of the 2nd International Conference on Rehabilitation Engineering*, 446-447.

Friedman, M.B., KILIANY, G., Dzmura, M. and Anderson, D. (1981) The Eyetracker Communication System. *1st National Search for Applications to Aid the Handicapped,* John Hopkins University, 183-185.

Gould, J.D., Lewis, C. and Barnes, V. (1985) Effects of Cursor Speed on Text-Editing. *Proceedings of the CHI'85 Conference on Human Factors in Computing Systems*, 7-10.

Hutchinson, T.E., White, K.P., Martin, W.N., Reichert, K.C. and Frey, L.A. (1989) Human-Computer Interaction Using Eye-Gaze Input. *IEEE Transactions on Systems, Man, and Cybernetics,* **19 (6)** 1527-1534.

Istance, H.O. and Howarth, P.A. (1994) Keeping an Eye on your Interface: The Potential for Eye-Based Control of Graphical User Interfaces (GUI's). In Cackton, G., Draper, S.W. and Weir, G.R.S. (eds), *Proceedings of HCI'94: People and Computers IX,* 195-209.

Koester, H.H. and Levine, S.P. (1994) Modelling the Speed of Text Entry with a Word Prediction Interface. *IEEE Transactions on Rehabilitation Engineering,* **2 (3)** 177-187.

Murphy, R.A. and Basili, A. (1993) Developing the user-system interface for a communications system for ALS patients and others with severe neurological impairments. *Designing for Diversity; Proceedings of the Human Factors and Ergonomics Society,* **2,** 854-858.

Quill, L.L. and Biers, D.W. (1993) On-screen keyboards: which arrangements should be used? *Designing for Diversity; Proceedings of the Human Factors and Ergonomics Society,* **2,** 1142-1146.

Schofield, J.M. (1981) *Microcomputer-based Aids for the Disabled.* Samet, P.A. (ed.), London: Heyden.

Shein, F., Hamann, G., Brownlow, N., Treviranus, J., Milner, M. and Parnes, P. (1991) WIVIK: A Visual Keyboard for Windows 3.0. *Proceedings of the 14th Annual Conference of the Rehabilitation Engineering Society of North America (RESNA),* 160-162.

Shein, F., Treviranus, J., Hamann, G., Galvin, R., Parnes, P. and Milner, M. (1992) New directions in visual keyboards for graphical user interfaces. In Murphy, H.J.(ed.), *Proceedings of the 7th Annual Conference on Technology and Persons with Disabilities,* 465-469.

Ten Kate, J.H., Frietman, E.E.E., Stoel, F.J.M.L. and Willems, W. (1980) Eye-Controlled Communication Aids. *Medical Progress through Technology,* Springer-Verlag, **8,** 1-21.

White, K.P., Hutchinson, T.E. and Carley, J.M. (1993) Spatially Dynamic Calibration of an Eye-Tracking System. *IEEE Transactions on Systems, Man and Cybernetics,* **23 (4)** 1162-1168.