Icon and Eye-Gaze based Text Entry Mechanism for Underprivileged Users

A Seminar Report
Submitted in a partial fulfillment of the requirements for the degree of

Master of Technology

in

Information Technology

by

Pawan Patidar
Roll No. 12IT60R11

Under the supervision of
Dr. Debasis Samanta

School of Information Technology
Indian Institute of Technology Kharagpur
Kharagpur-721302, India

September 16, 2013
## Contents

1 Introduction .......................................................... 1  
2 Related Works .......................................................... 1  
   2.1 Text Entry by Direct Gaze Pointing ................................. 1  
   2.2 Text Entry though Eye Switches ................................... 2  
   2.3 Text Entry by Discrete Gaze Gestures ............................. 2  
   2.4 Text Entry by Continuous Pointing Gestures .................... 6  
3 Plan of Work ............................................................ 8  
4 Work Done ............................................................... 9  
   4.1 Design of Interface ................................................ 9  
   4.2 Operation .......................................................... 10  
   4.3 Experiment ......................................................... 12  
   4.4 Results ............................................................ 13  
   4.5 Discussion and Conclusion ....................................... 16  
5 Future Plan of Work .................................................. 16  
6 References ............................................................... 17
Abstract

Text entry by eye gaze is used by people with severe motor disabilities. An eye tracking device follows the users eye movements, and a computer program analyses the gaze behavior. To type by gaze, the user typically points at the characters on an on-screen keyboard by looking at them and selects them by means of dwell time, a prolonged gaze that separates an intentional command from casual viewing. This report provides an extensive review of the research conducted in the area of Eye Gaze based Text Entry. It summarizes results from several experiments that study various aspects of text entry by gaze. Results show that small improvements in the interface design can lead to significant improvements in user performance and satisfaction.

Among various interfaces, pie menus are suggested as powerful tool for eye gaze based text entry among various interfaces developed so far. If pie menus are used with multiple depth layers then multiple saccades are required per selection of item, which is inefficient because it consumes more time. Also dwell time selection method is limited in performance because higher dwell time suffers from inefficiency while lower one from inaccuracy. To overcome problems with multiple depth layers and dwell time, we designed Quickpie, an interface for eye gaze based text entry with only one depth layer of pie menu and selection border as selection method instead of dwell time. We investigated various parameters like number of slices in pie menu, width characters and safe region, enlarged angle of slice and selection methods to achieve better performance. Our experiment results indicates that six number of slices with width of characters area 120 px performs better as compared to other designs.
1 Introduction

The aim of this report is to present a comprehensive study of the gaze-based text entry process and to find ways to make the interaction more efficient and enjoyable. The eye gaze-based text entry, often called as eye typing, is accomplished by direct pointing or looking at the desired letter within interface. To type by gaze, typical computational requirements include an on-screen keyboard and an eye tracking device. Selection of a letter, i.e. eye press, is accomplished by hovering on the letter for a slightly prolonged duration, i.e. dwell time. Many gaze typing systems support eye blink besides eye movement during typing. As the gaze-based text entry mechanism is gaining popularity among alternate text entry mechanisms, it brings a number of design issues that make gaze-based text entry a unique technique with own set of research problem.

Eye gaze based interaction enables user to focus at point of interest on the display by just looking at it. To perform such cursor movements with eye gaze we need a camera which captures images of eye and an eye tracking system is required which takes these images as input and gives coordinates of point where user is looking as output. This way of interaction is particularly very useful for motor handicapped patients because eye movement is fastest motor movement and a person can perform with rotation speed of $700^\circ/s$ [1].

To enter a letter user have to first focus on letter on virtual interface and then take some distinguished action. Two most obvious actions are blinking or dwelling. People blink involuntarily in every several seconds [2], so that it becomes difficult to distinguish between a desired and unintentional blink. One possible way to overcome this problem is to use prolonged blink. In dwelling mechanism, user has to fixate eye gaze on desired area for a specific amount of time called dwell time. Dwell time commonly varies from 200 to 1000 ms and it must be longer than the normal viewing time to prevent false selections. But longer dwell time results in significant degradation in performance, i.e. typing speed because it elapses for each selection of characters. Hence dwell time should be moderate to improve typing speed and avoid false selections according to needs and expertize of the user [6]. Therefore it is observed that blinking and dwelling both suffered from Midas touch problem.

Between two consecutive fixations our gaze suddenly jumps from one point to another through ballistic eye movements. Such movements between fixations are called saccades. A saccade commonly takes 30 to 120 ms having an amplitude range between $1^\circ$ and $40^\circ$ (average $15^\circ$ to $20^\circ$) [1, 4]. Latency period of at least 100 to 200 ms occurs before eye moves to next area of interest and after a saccade eyes will fixate on an object between 200 and 600 ms [1, 4]. Therefore a selection of character can be done faster if it requires only one saccade or more saccades in same direction. Such sequence of saccades is very helpful if user is expert and can predetermine path to select a desired character. User can mark ahead path corresponding to a character without searching for it [5]. Therefore searching time can be significantly minimized.

2 Related Works

2.1 Text Entry by Direct Gaze Pointing

The most common way to use gaze for text entry is direct pointing by looking at the desired letter. A typical setup has an on-screen keyboard with a static layout, an eye tracking device that tracks the users gaze, and a computer that analyzes the users gaze behavior. The Tobii eye tracking device has a camera integrated into the frame of the monitor that shows the on-screen keyboard. To type by gaze, the user focuses on the desired letter by looking at one of the keys on the on-screen keyboard. The system gives feedback on the item in focus, for example, by highlighting the item or by moving a gaze cursor over the key in focus. Once it has the focus, the item can be selected via, for instance,
a separate switch, a blink, a wink, or even a wrinkle or any other (facial) muscle activity. The typed letter appears in the text field, often located above the keyboard. The system may also give feedback on successful selection by speaking out the letter or placing a click sound.

### 2.2 Text Entry though Eye Switches

Some people may have difficulties in fixating because of their physical condition or state of health. They cannot keep their gaze still for the time needed to focus. The user may also be able to move his or her eyes in one direction only. In such cases, other methods for selecting an item by gaze are needed. Voluntary eye blinks or winks can be used as binary switches. For text entry, blinks are usually combined with a scanning technique, with letters organized into a matrix. The system moves the focus automatically by scanning the alphabet matrix line by line. The highlighted line is selected by an eye blink. Then the individual letters on the selected line are scanned through and again the user blinks when the desired letter is highlighted (see Figure 1).

![Figure 1: Sequential row-column scanning.](image)

### 2.3 Text Entry by Discrete Gaze Gestures

Several discrete gaze gestures can also be combined into one operation. This approach is used in VisionKey, where an eye tracker and a keyboard display (Figure 2, right) are attached to eyeglass frames (on the left in Figure 2). VisionKey is attached to eyeglass frames (left). It has a small screen integrated into it (right). Because the key chart is attached in front of the users eye, it is important to make sure that simply staring at a letter does not select it. The VisionKey selection method avoids the Midas touch problem by using a two-level selection method i.e., two consecutive gaze gestures for activating a selection. To select a character, the user must first gaze at the edge of the chart that corresponds to the location of the character in its block. For example, a user wanting to select the letter G first glances at the upper right corner of the chart and then looks at the block where G is located (or simply looks at G). After a predefined dwell time, the selected key is highlighted to confirm a successful selection.

Wobbrock et al. (2008) adapted EdgeWrite (Wobbrock et al., 2003) for gaze control and called it EyeWrite (illustrated in Figure 3). In both, the user enters characters by drawing letter like gestures
Figure 2: VisionKey is attached to eyeglass frames (left). It has a small screen integrated into it (right) in a small dedicated window. The advantage of a separate input window is that there is no danger of the stroke overlapping with other user interface objects, which ensures robust interpretation of the gestures. The window also enables dynamic feedback while the user enters the strokes needed for each gesture. As soon as the gaze enters the active area, an arch is drawn there to illustrate a successful stroke and the target character recognized is shown in the final target area.

Huckauf and Urbina (2007) developed a dwell-time-free text entry system that takes advantage of pie menus, called pEYEs, or pEYEeditor pEYEwrite (see Figure 4). Since bigger sectors are easier to select, letters are grouped into the sectors of the pie. To enter a letter, the user moves the cursor (by gaze) such that it crosses the outer part of the sector that contains the desired letter. A sub-menu
with a separate sector for each of the letters opens immediately without the need for dwelling on it. The target letter is selected by glancing at (or over) the outer part of the sector where the desired letter is located. Again, no dwell time is needed. Since entering a letter requires the selection of two sectors, two strokes are needed for one character.

Figure 4: pEYEwrite (Huckauf & Urbina, 2007) uses pie menus for text entry.

Figure 5: In the figure, the letter A is being selected.

Bee and Andr (2008) adapted Quikwriting (Perlin, 1998) for gaze input (as suggested by Isokoski in 2000 but never implemented by him). In the original Quikwriting, the characters are located in the
active selection areas (sections). The characters are grouped such that the location of each indicates the gesture needed for entering it. Using a hand-operated input device, the user can first search for the target letter and then select it by moving the cursor (e.g., by mouse) to the selection area(s). The character is entered when the pointer is returned to the center. This is a problem with gaze input, since the user cannot visually search for the characters without immediately initiating the selection process. To prevent unintentional selection during visual search, Bee and Andr moved the characters into the inner resting area, near the corresponding section (see Figure 6). The characters were still grouped such that the gesture needed is shown by the position of the character within the group. Furthermore, to support the users selection process, each character from the group was shown in the adjacent sections as soon as the user initiated the selection by moving the cursor to one of the sections (away from the central area). Showing the characters within the sections eliminated the need to look at the central area if the user were to forget which section(s) he or she should select to enter the character. If the user had to look back at the center before finalizing the selection process, an error was likely to result, since a character is entered whenever the pointer is returned to the central area. The location of the text input field where the characters entered appeared.

Also caused problems in the gaze-based implementation: the user could not move the gaze from the central area to the text input area and back without crossing the active selection areas in between and thus unintentionally making selections. In their original implementation, Bee and Andr disabled the writing process when the user looked at the text input field and enabled writing when the gaze returned to the central area. However, after the user trial, they decided it was better to show the written text inside the central area, to minimize the swapping of the gaze between the central area and the text input field (though only a small amount of the previously written text would fit there).

![Figure 6: Quikwriting interface adapted for gaze. The figure illustrates the gaze path for gaze writing g (image courtesy of Dr. Nikolaus Bee, Augsburg University).](image)

The adapted Quikwriting interface was tested against an on-screen keyboard, using a 750 ms dwell time. Participants achieved 5.0 wpm with Quikwriting and 7.8 wpm with the dwell-time-based keyboard. The result is encouraging, though it should be noted that it is based on only a very small
Both Quikwriting and pEYEwrite enable continuous writing without a pause between characters (or words). The segmentation of the gestures is part of the process of moving the pointer (gaze) from one selection area to another. Nevertheless, the writing is still based on distinct gestures, even though some implementations enable entering them without a pause in between. In the following section, we will introduce text entry methods based on continuous pointing gestures. The user does not need to make any distinct gestures to enter characters; only one type of gesture is needed: pointing. However, here the pointing is different from the direct pointing used with the on-screen keyboards. The on-screen keyboard is static (even if implemented as sub-menus), and the user selects the letter by looking at the on-screen key. The text entry methods introduced below implement interfaces that change dynamically, and the user selects (groups of) characters by navigating through a world of characters that is continually changing. Thus, even if one can argue that this is indeed direct pointing, the pointing gesture is not static. It follows the dynamically changing interface, and the direction of the pointing changes smoothly and continuously while writing is taking place.

2.4 Text Entry by Continuous Pointing Gestures

Continuous writing can be especially useful for text entry by gaze because of the nature of human gaze. First, our eyes are always on (if not closed), so it can be compared to a pencil that is never lifted from the paper. Our eyes also move constantly; it is not natural for us to hold our gaze for long on a target. Even if we keep looking at one object, we usually make small saccades within (and around) the object. Dasher is a zooming interface that is operated with continuous pointing gestures. In the beginning, the letters of the alphabet are located in a column on the right side of the screen (see Figure 7). Letters are ordered alphabetically (Figure 7). The user moves the cursor to point at the region that contains the desired letter, by looking at the letter. The area of the selected letter starts to zoom in (grow) and move left, closer to the center of the screen (Figure 8). Simultaneously, the language model of the system predicts the most probable next letters. The areas of those letters start to grow (as compared to other, less probable letters) within the chosen region. This brings the most probable next letters closer to the current cursor position, thus minimizing distance and time to select the next letter(s). The letter is typed when it crosses the horizontal line at the center of the screen. Canceling the last letter is done by looking to the left; the letter then moves back to the right side of the screen.

Stargazer (Hansen et al., 2008) is another system that takes advantage of zooming. However, in Stargazer, the user zooms on the z-axis and panning occurs on the x- and y-axis. At the beginning, all characters are located in the space in a circular form in a familiar (in this case, alphabetical) order around the central area (see Figure 9 and 10). Special characters for backspace, undo, and stop actions are placed in the corners of the display (in some configurations, there is also an option for adjusting the speed, placed in one of the corners see Hansen et al., 2008). The user navigates (flies) in the 3D space of characters by looking at the desired character. The 3D cursor (a spaceship made of three concentric circles) points at the direction of navigation. The display will pan towards the target character and that character will be moved to the center of the screen. The target character will start to grow bigger, indicating that the user is approaching it. Panning and zooming occur simultaneously in part, depending on the thresholds set for the central zoom area. Selection is performed by flying through the target letter. The user is always returned to the initial view after a selection is made.

In an experiment with 48 participants, Hansen et al. (2008) found that novice users learned to write with Stargazer by gaze almost immediately. In the first try, they were able to write their name without losing the orientation. The grand mean of typing speeds was 3.47 wpm, which is fairly slow. However, in another experiment (with seven participants) it was found that after only five minutes of practice, participants achieved 8.16 wpm with a mean error rate of 1.23 with the speed achieved via a
Figure 7: Dasher facilitates text entry via navigation through a zooming world of letters. In the initial state, the letters are ordered alphabetically on the right side of the screen.

Figure 8: As the user looks at the desired letter, its area starts to grow and simultaneously the language prediction system gives more space to the most probable next letters. In the image on the right, the user is in the middle of writing name, with n already selected.

dwell-based on-screen keyboard.

Both Dasher and Stargazer initiate smooth pursuit eye movements while the users gaze follows the target. Both are also mode-free and thus well suited to gaze input.
Figure 9: Writing with Stargazer (Hansen et al., 2008) always starts from the initial point in the center, from which the user navigates towards the desired character.

Figure 10: The view pans and zooms toward the target.

3 Plan of Work

Even when a user wants to fixate on an object by looking steadily, the eyes make small jittery motions. These jittery movements are of less than one degree and can be high frequency tremor or low frequency drift. Moreover high-acuity region of our retina called fovea covers approximately one degree of visual arc that is why we cannot determine precisely where user is pointing by looking steadily.
Considering above difficulties we designed the interface (Quickpie) to fulfil following objectives:

1. Removing dwell time and minimizing searching time and eye movements to maximize typing speed.

2. Preventing false selections (Midas touch problem) to minimize error rate.

4 Work Done

4.1 Design of Interface

Quickpie consists of four regions: pie, characters region, safe region and selection region (see figure 11). A typical design has following properties:

![Figure 11: Effect of number of slices on text entry rate.](image)

**Pie region**

The pie menu at first layer is having radius of 240 px and divided into six slices. Each slice is having initial angle ($\alpha$) equal to 60$^\circ$ colored gray and blue alternatively. Each slice having five characters arranged in alphabetical order (except last slice which contains SPACE and CLEAR with z). When user focus on a slice then it is highlighted and its angle expands 20$^\circ$ in both clockwise and anticlockwise direction so that its enlarged angle ($\beta$) becomes 100$^\circ$. $\beta$ must be much less than 3$\alpha$ otherwise major area of adjacent slices would be overlapped it would be difficult to focus on them.

**Characters region**

This is colored gray (darkness is increasing for each character in clockwise direction) and placed outside the border of pie. This region is having width of 100 px. Characters are arranged alphabetically in clockwise direction and each character is having angle($\gamma$) equal to enlarged angle/number of characters per slice = $\beta/5 = 20^\circ$. When user points on a particular character then its text color changes to red.
Safe region
This is black colored region at outer edge of characters region. This region is having width of 20 px and nothing happens when user focus on this region. This is used to minimize errors due to jittery movements.

Selection region
This is colored red and placed at outer edge of safe region. Width of this region does not important and its angle is equal to enlarged angle ($\beta$). When user enters in this region from safe region than its color turns to green and highlighted character in characters area is entered.

![Diagram of characters](image)

Figure 12: Effect of number of slices on text entry rate.

4.2 Operation
To enter a character, user has to follow following procedure:

1. Focus on the slice containing that character (Corresponding characters, safe and selection areas will appear immediately without any delay. If another slice was focused previously then its corresponding areas will disappear).

2. Focus on area of desired character (Text color of that character will be turned to red).

3. Cross safe region and enter into selection region(after crossing to selection region its color will turn to green, character selected in step 2 will be entered and displayed at current position of cursor so that there is no need to verify entered character by looking at the output text area).

If next character to be entered is in same slice then user do not need first step because desired character is already visible. Hence only second and third steps are required until the next character to be entered is in different slice(see Figure 12).
Temporal threshold is totally removed because no dwell time is required in any of above three steps which is main advantage of this interface. Since $\gamma = \beta/5$, $\beta$ should be taken much higher than $\alpha$ because large items are easy and accurate to focus [7]. Increasing radius of pie is another way to enlarge size of each character but eye movements would be higher in that case which may result in performance degradation. Novice users may takes longer searching time but expert user can search fast and can predetermine path to be followed to enter a sequence of characters. This process of prediction is called mark ahead [5], which is another main advantage of this interface.
Characters area, safe area and selection area are in same direction so that a character can be entered in only one saccade or more saccades in nearly same direction. Therefore it eliminates disadvantage of delays which occurs between saccades in different directions. Moreover characters are alphabetically arranged and not changing position, therefore searching time may be longer for a novice user but it diminishes as user becomes familiar with interface and memorizes positions of characters. Since selection region is outside the characters region, user can dwell for much longer time without producing false selections. This major problem in interfaces with dwelling mechanism is removed. Hence a novice user can take longer searching time without making errors.

Another problem in Quickpie is that due low accuracy of the eye tracking systems and jittery movements, cursor may oscillate several times between characters and selection region which produces erroneous repetitions of selected character although user traversed correct path through gaze only one time. This problem justifies use of safe region (see figure 14). Width of safe region has to be selected according to accuracy of eye tracking. It should be less for more accurate eye tracking systems and ideally it can be zero if system works perfectly.

4.3 Experiment
After getting the optimal solution by using the calculus at user side, this optimal solution is forwarded to every grid and on the basis of that every grid plays a different strategy so that user is able to select the best grid and gets the maximum benefit. The grids play a strategy by selecting the values of different parameters such that it provides the optimal benefit for the user and at the same time grid is also able to maximize its payoff. The optimal solution provided by the grid may not fit into the requirements of the user, however, while playing a strategy grid has to keep in mind that its payoff does not go below a threshold value. For \(i_{th}\) grid the optimization of utility is that:

\[\text{Utility} = \text{Optimal Benefit} \geq \text{Threshold Value}\]

Apparatus
The study took place using Dell-PC with 2.2 GHz Intel Core2Duo processor and 2 GB RAM under windows XP. The interface was displayed on Samsung LCD color display having resolution of 1364768 in normal artificial lightning. Modified Sony PlayStation eye webcam (original lens was replaced by other lens without infrared (IR) filter), open source ITU gaze tracker (developed by IT University of Copenhagen) and an IR lamp consisting of a matrix of 10 IR LED were used. The webcam kept stationary at 40 cm in front of the monitor (see figure 15).

Participants
Eight participants (6 male and 2 female aged between 22 and 28) participated, 2 were expert and 6 were novice to the interface. All of them except one reported normal or corrected-to-normal vision and all were familiar to computer and text composition (see figure 16).

Procedure
Before starting experiment, user needs to calibrate eye using gaze tracker to be able to move cursor with eye gaze. Since recalibration may be required if user changes position of head, participants were asked to remember the phrase to be typed. All were explained clearly how to type using the interface. Participants were instructed to compose text as soon as possible with minimum errors and correct an error using CLEAR key before retype.
4.4 Results

Text entry rate is measured in words per minute (wpm) and error rate in percentage of uncorrected characters to the total characters typed, taking mean of all eight users performance.
Number of slices

To experiment for number of slices, designs with four, five, six and seven slices all with width of characters area 100 px were compared.

**Text entry rate:** 2.53, 2.82, 3.46 and 3.08 wpm are text entry rates of designs with four, five, six and seven number of slices respectively (see figure 17).

![Figure 17: Effect of number of slices on text entry rate.](image1)

**Error rate:** 3.4, 2.9, 3.1 and 3.2 % are error rates of designs with four, five, six and seven number of slices respectively (see figure 18).

![Figure 18: Effect of number of slices on error rate.](image2)

Design with six number of slices has highest text entry rate (3.46 wpm). Although it has slightly more error rate than design with five number of slices, it is selected for further experiments.

Width of character area:

To experiment for width of characters area, designs with width of characters area 80, 100, 120 and 140 px all with six number of slices were compared.

**Text entry rate:** 3.20, 3.73, 4.27 and 3.93 wpm are text entry rates of designs with width of characters area 80, 100, 120, 140 px respectively (see figure 19).

**Error rate:** 3.4, 3.1, 2.7 and 2.8 % are error rates of designs with width of characters area 80, 100, 120, 140 px respectively (see figure 20).
Results are showing that design with width of characters area 120 px has highest text entry rate and lowest error rate. Therefore it is best design among all other designs and selected for further experiments.

**Learnability**

To experiment learnability, design with six number of slices and width of characters area 120 px was taken. One run with this design was held in previous experiment and again experiments with same design were repeated four times.

**Text entry rate:** 3.93, 4.64, 5.39, 5.85 and 6.14 wpm are text entry rates of first, second, third fourth and fifth run respectively (see figure 21).

**Error rate:** 2.8, 2.6, 2.7, 2.9 and 2.8 % are error rates of first, second, third fourth and fifth run respectively (see figure 22).

Text entry rate is increasing faster in early runs but it is not increasing significantly in later runs. Also there is not any pattern in improvement of error rate in subsequent runs.

**Selection method**

The design with width of characters area 120 px and six number of slices was taken again with dwelling time selection method. Dwell time of 400 ms was taken to compare results with selection area method.
Text entry rate of 4.38 wpm and error rate of 3.4

4.5 Discussion and Conclusion
The proposed interface is having no timing threshold as dwell time is removed. It is taking advantage of mark ahead selection that is why performance increases significantly as user gets familiar with the interface. To achieve higher text entry rate, various parameters like number of slices and width of characters region are crucial factors. Also radius of pie region, width of safe region and enlarged angle may affect performance but it is not investigated. Experiment results shows that design with sis number of slices and width of characters area 120 px is best in typing speed and error rate.

5 Future Plan of Work
Currently we are using alphabetical ordering of characters, but clustering and effective positioning of these clusters can be done to improve the performance. Further research can be carried out to minimize effects of jittery movements in this interface and best values of other parameters according to apparatus and type of users.
6 References