www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 2 Issue 5 May, 2013 Page No. 1439-1448

Evaluation of Eye Gaze Interaction

Miss A W. Chorey
Computer Science of Engineering Sipna College of Engineering Amravati, India

Abstract—***Eye gaze interaction added simple and convenient way to communicate with computer. With a growing number of computer devices around us, and the increasing time we spend for interacting with such devices, we are strongly interested in finding new interaction methods which ease the use of computers or increase interaction efficiency. Eye tracking seems to be a promising technology to achieve this goal. In this paper, I present two experiments that evaluate an interaction technique. The results show that algorithm, which makes use of knowledge about how the eyes behave, preserves the natural quickness of the eye. Eye gaze interaction is a reasonable addition to computer interaction and is convenient in situations where it is important to use the hands for other tasks. It is particularly beneficial for the larger screen workspaces and virtual environments of the future, and it will become increasingly practical as eye tracker technology matures.***

Keywords
Eye movements, eye tracking, user interfaces, interaction, and techniques.

I. Introduction

With the invention of the computer in the middle of the last century there was also the need of an interface for users. In the beginning experts used teletype to interface with the computer. Due to the tremendous progress in computer technology in the last decades, the capabilities of computers increased enormously and working with a computer became a normal activity for nearly everybody. With all the possibilities a computer can offer, humans and their interaction with computers are now a limiting factor. This gave rise to a lot of research in the field of HCI (human computer interaction) aiming to make interaction easier, more intuitive, and more efficient. Interaction with computers is not limited to keyboards and printers anymore. Different kinds of pointing devices, touch-sensitive surfaces, high-resolution displays, microphones, and speakers are normal devices for computer interaction nowadays. There are new modalities for computer interaction like speech interaction, input by gestures or by tangible objects with sensors. A further input modality is eye gaze which nowadays finds its application in accessibility systems. Such systems typically use eye gaze as the sole input, but outside the field of accessibility eye gaze can be combined with any other input modality. Therefore, eye gaze could serve as an interaction method beyond the field of accessibility. The aim of this work is to find new forms of interactions utilizing eye gaze and suitable for standard users.

An eye-gaze interface seems to be a promising candidate for a new interface technique, which may be more convenient than the ones we use. Traditionally, disabled people who cannot move anything except their eyes use eye gaze interaction. These systems are designed to direct the computer solely by the eyes. Such systems work well and are a great help for people who need them, but for others they are cumbersome and less efficient than keyboard and mouse. This contradicts the fact that looking is an easy task and that eye movements are fast. An eye-gaze interface might offer several potential benefits like ease of use, interaction speed-up etc.

The details of the experiment give insight into how our eye gaze interaction technique works and why it is effective. It is not surprising that the technique is somewhat faster than the mouse. Our research tells us the eye can move faster than the hand. The test of our approach is how our entire interaction technique and algorithm preserves this speed advantage of the eye in an actual object selection task. We studied the physiology of the eye and used that information to extract useful information about the user's higher-level intentions from noisy, jittery eye movement data. Even though our algorithm is based on an understanding of how eyes move, it was unclear that our eye gaze interaction technique would preserve the quickness of the eye because the eye tracking hardware introduces additional latencies. Performance of any interaction technique is the product of both its software and hardware.

II. Overview and Related Work

There is an immense knowledge on eye tracking and related fields such as the anatomy and physiology of the eye, the movement of the eyes and visual perception. The knowledge spreads over many disciplines of science including biology, medicine, psychology, and neurology. This overview starts with a definition of eye tracking followed by a short history of eye tracking and the fields of application for eye trackers. The next two sections explain the eye-tracking technology and present available eye-tracker systems.

A. Definition of Eye Tracking

The term *eye tracking* as it is used here means the estimation of direction of the user's gaze. In most cases the estimation of the gaze direction means the identification of the object upon which the gaze falls. In the case of a standard computer device, the coordinates on the screen identify the object of gaze. Interpretation of gaze direction is more complex for eye tracking in 3D virtual worlds and becomes difficult when interacting with the real world.

Eye trackers differ in the degrees of freedom which they can track. Simple eye trackers report only the direction of the gaze relatively to the head (EOG and systems rigidly mounted on the head) or for a fixed position of the eyeball (systems which require a head fixation). Systems that are more sophisticated allow free head movements in front of a stationary system. Such systems do some kind of (implicit) *head* tracking. In addition, wearable eye trackers for use in 3D virtual worlds have to report the direction of the gaze in space and not only relatively to the head. This work refers to such systems with the term eye tracker and does not explicitly name such systems as eye-and-head-tracker. (In the same way, this work refers to the eye and not to the eye and equilibrium organ although this would be more precise in some cases.)

Most video-based eye trackers deliver not only the direction of gaze but also the size of the pupil. The widening and narrowing of the pupil is an emotional response to the perceived scene and for this reason is quite interesting for research. However, as the main function of the pupil is the regulation of the amount of light entering the eye, such research requires stable light conditions. This work does not consider eye tracking in the sense of pupil size detection.

B. Methods of Eye Tracking

The most direct method is the fixation of a sensor to the eye. The fixation of small levers to the eyeball belongs to this category, but is not recommended because of high risk of injuries. A safer way of applying sensors to the eyes is using contact lenses. An integrated mirror in the contact lens allows measuring reflected light. Alternatively, an integrated coil in the contact lens allows detecting the coil's orientation in a magnetic field (see Figure 3). The thin wire connecting the coil with the measuring device is not comfortable for the subject. The big advantage of such a method is the high accuracy and the nearly unlimited resolution in time. For this reason, medical and psychological research uses this method.

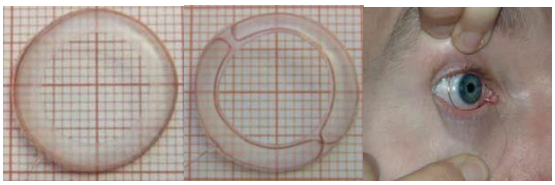

Fig.1 Search

Another method is electrooculography (EOG) where sensors attached at the skin around the eyes measure an electric field (see Figure 4). Originally, it was believed that the sensors measure the electric potential of the eye muscles. It turned out that it is the electric field of the eye which is an electric dipole. The method is sensitive to electro-magnetic interferences but works well as the technology is advanced and exists already for a long time. The big advantage of the method is its ability to detect of eye movements even when the eye is closed, e.g. while sleeping.

Fig.2 Sensors attached at the skin around the eyes

Both methods explained so far are obtrusive and are not suited well for interaction by gaze. The third and preferred method for eye-gaze interaction is video. The central part of this method is a video camera connected to a computer for real-time image processing. The image processing takes the pictures delivered from the camera and detects the eye and the pupil to calculate the gaze's direction. The following sections present several solutions to achieve this. The big advantage of video-based eye tracking is the unobtrusiveness. Consequently, it is the method of choice for building eye-gaze interfaces for human-computer interaction.

III. Experiment-1 Gaze Gestures

Gestures are a well-known concept for computer human interaction. The idea behind gestures is the fact that we are used to employ movements of the body, mostly with the hands and the head, to communicate or to support communication. While such intuitive gestures are vague and culture dependent we are also able to perform well-defined and elaborated gestures. One example is handwriting; other examples are the sign language for deaf mutes or the semaphore alphabet.

Gestures for computer interaction are not very intuitive, as it requires learning a set of gestures and their semantics. For this reason the use of gestures for computer interaction has been seen as something for specialists or for very specific purposes. However, with the introduction of the iPhone and similar products, which have a touch sensitive surface as the only input modality, the use of gestures performed with the fingers became popular. In addition, the interaction with tabletop computers, which will appear on the mass market soon, will strongly rely on gesture input.

The research presented here focuses on gestures performed with the eyes. We use eye movements for our human-human communication which could be called eye gestures. Typical examples are to wink, to blink with the eye, or to roll the eyes. Such eye movements can include movements of the eyelid and eyebrows and may be seen as part of a facial expression. This work restricts itself to the eye movements reported from a

commercial eye tracker which represents the direction of the gaze. For this reason the gestures performed with the eye are called gaze gestures and not eye gestures. This work restricts itself to the eye movements reported from a commercial eye tracker which represents the direction of the gaze. For this reason the gestures performed with the eye are called gaze gestures and not eye gestures.

A. Concept of Gaze Gestures

1) The Firefox/Opera Mouse Gestures

The popular and freely available mouse gesture plug-in for the Firefox web browser gave the inspiration to implement a similar gaze gesture algorithm. The mouse gesture plug-in traces the mouse movements when a gesture key, normally the right mouse key, is pressed and translates the movements into a string of characters or tokens representing strokes in eight directions as depicted in fig.3.

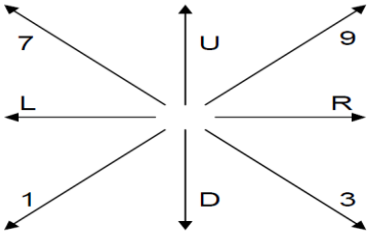

Fig. 3 The naming of the eight directions used for the mouse gestures.

The mouse gesture recognition algorithm of the Firefox browser receives the coordinates of the mouse device. Whenever the mouse pointer moved more than a threshold distance away from the start position the algorithm outputs a character for the direction of the movement. The current mouse pointer position becomes the new start position and the detection of a new stroke starts. The algorithm uses eight directions; U, R, D, and L for up, right, down and left respectively and 1, 3, 7, and 9 for the diagonal direction according to the standard layout of the number pad on the keyboard. The notation of the direction follows the notation introduced by the mouse gesture algorithm.

The definition of a gesture is a string consisting of the eight characters for the eight directions. As the mouse gesture recognition does not produce a new character if the direction did not change, the string must not contain pairs of the same character.

2) The EdgeWrite Gestures

Beside the mouse gestures the EdgeWrite were most inspiring for this work. The EdgeWrite gestures use the order in which four points, the corners of a square, are reached (see Figure 4). It is easy to see that all EdgeWrite gestures can be expressed with the direction characters of the mouse gestures. Consequently, the EdgeWrite gestures are a subset of the mouse gestures. Nevertheless, they have the capability to define a large alphabet as shown by Wobbrock et al. who assigned at least one gesture to all letters and digits in the Latin alphabet. As a display has four corners to look at, the Edgewrite gestures are easy to adapt to the gaze.

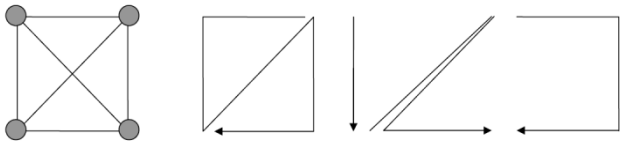

Figure 4: The four corners and the six connecting lines used for the EdgeWrite gestures and examples for EdgeWrite gestures (digits 0, 1, 2 and 3).

B. The Gaze Gesture Recognition Algorithm

A gesture consists of a sequence of continuous elements, typically strokes, which are performed in a sequential time order. To detect a gesture it is necessary to know when the gesture starts and when it ends. In the case of pen input the gestures starts at the moment the pen touches a surface and ends when lifting the pen. For the mouse gestures the situation is different, as it does not help to lift the mouse pointer. For this reason the mouse gestures use a gesture key, the right mouse button in the standard configuration, to indicate the input of a gesture. This makes sure that the algorithm does not detect gestures during normal mouse input and it allows defining an alphabet where one gesture pattern is a part of another gesture pattern.

The situation for gaze gestures is similar to the mouse gestures because we cannot lift the gaze like a pen. However, gaze gestures should avoid the need of a gesture key. With the need of a gesture key the gaze gestures lose the properties of being a remote control and a hygienic (touch free) interface which is their main benefit.

The absence of a gesture key requires continuous gesture recognition and creates the problem that natural eye movements could produce a valid gaze gesture without intention. The problem of separating gaze gestures from natural eye movements is very similar to the challenge of recognizing spoken commands for computer input where a normal conversation should not trigger commands.

The mouse gesture detection algorithm works in a quite simple way. The algorithm receives x-y positions and calculates the difference to the start position. As long as an integer division with grid size *s* produces zero for both components the algorithm continues to receive further x-y positions. As soon as the integer division has a result different from zero for at least one component, the current position becomes the new start position and the algorithm produces a character for the direction of the stroke as output, but only if this character is different from the last output. See Figure 5 for an illustration. The algorithm detects a gesture if the output of the last characters matches a given gesture string.

To achieve a good separation from natural eye movements the demand is a performance of the gesture without break. This means that a time aspect has to be introduced into the detection

algorithm. Whenever the algorithm did not detect a stroke for time *t* it outputs a colon as the ninth possible character. Gazing longer than time *t* at the same position resets the gesture recognition. The output of a colon also sets a new start position. This becomes important when working with big grid sizes i.e. grid sizes bigger than half of the display size.

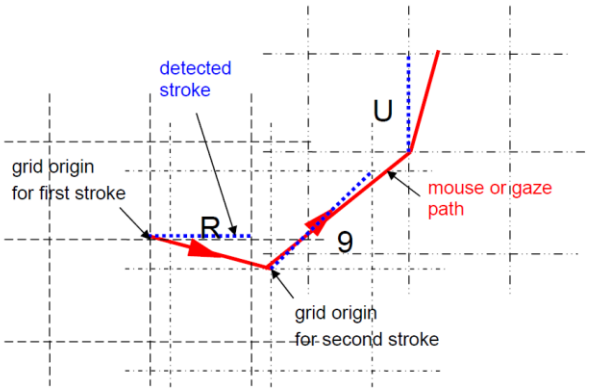

Figure 5: The figure shows how a mouse or gaze path translates into the string R9U. The end of a detected stroke is the origin for the grid to detect the next stroke.

C. User Study and Experiment with Gaze Gestures

To test the applicability of gaze gesture user study and experiment was conducted. The goal of the user study used gaze gesture for secure pin entry at ATM cash machines.

1) PIN-Entry User Study

Personal identification numbers (PINs) are one of the most common ways of electronic authentication these days and they are used in a wide variety of applications. PINs have to stay a personal secret as revealing the PIN can result in a personal loss. In the case of drawing cash from an ATM, which mostly happens in public space, there is a high danger that somebody observes the entry of the PIN. To avoid such shoulder surfing a less observable PIN entry method is desirable. Assuming that observing and interpreting eye movements is much more difficult than observing the finger movements on a number pad leads to the idea to use eye gaze for a more secure PIN entry. The study presented here used the same gaze-based input techniques, but additionally researched the use of gaze gestures as a further input technique.

The PIN-entry user study used three different gaze-based techniques for PIN entry. The first and second method used gaze pointing to enter the PIN on a number pad displayed on the screen (see Figure 6). The first method used a dwell time of 800 milliseconds and the second method used a button, which had to be pressed when looking at the correct number on the number pad display. The second method was introduced as hardware key or gaze key, but called look & shoot method in the context of the user study as this name is self-explaining and got high acceptance by the participants. The prototype displays an ATM number pad of an overall size of 730 × 450 pixels. Each button has a size of 180 × 90 pixels which is about 5° by 3° visual angle and hence clearly above the typical eye tracker accuracy of ± 0.5°. To retain the security benefit there was no feedback except asterisks to indicate successful entry of a digit.

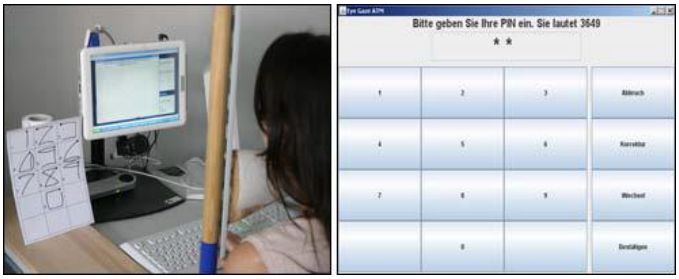

Fig. 6 The user study setting (left) and the layout of the number pad for PIN entry

The third method used gaze gestures to enter the digits. As there is no need for remote entry of PINs it is not an obstacle to use a gesture key. With a gesture key it is not necessary to separate natural eye movements from gestures and this allows more freedom in the design of a gesture alphabet and consequently allows a more intuitive alphabet. Without a gesture key the digit 9 and 5 would not be distinguishable.

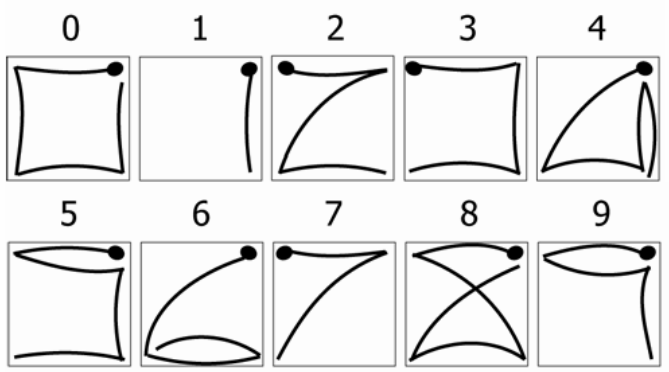

Fig. 7 The digit gestures used for the prototype

In the user study, 21 volunteers completed the three different PIN entry tasks and afterwards answered a questionnaire. Seven of the participants were female and all participants were aged between 22 and 37 years. Five of them had already used an eye tracker before, but not on a regular basis. Figure 9 shows the completion time and error rate for the three different entry methods. The evaluation of the data using analysis of variance (ANOVA) showed no significant advantage regarding execution times for the look & shoots or dwell time method. Using the look & shoot method a four digit PIN entry took the subjects 12 seconds in average whereas a PIN entered using dwell time took 13 seconds. The error probability also showed no significant difference. Using dwell time, 15 of the entered 63 PINs were faulty (23.8%), using look & shoot 13 entered PINs contained errors (20.6%).

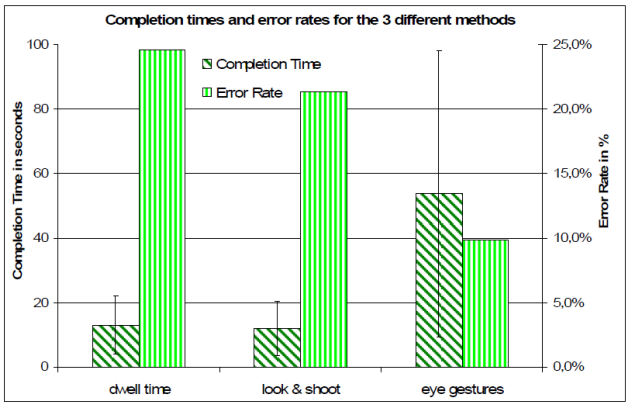

Fig.8 Completion times and error rates for three different methods of eye gaze interaction

The results of the user study show that eye gaze interaction is a suitable method for PIN.Entering PIN numbers with the gaze gesture method took much longer than using the 'classic' methods (an average of 54 seconds per PIN entry was measured) but was also much more robust against errors than the methods described above. Only six of the entered PINs using gestures were erroneous (9.5%). Using a binomial test shows a significant enhancement of the error rate ($p < 0.008$).

The gaze gesture method is less intuitive than the classic methods as some subjects initially had problems to produce recognizable gestures. Furthermore, the gesture alphabet was unknown to all participants. This explains the big difference in time for completing the PIN entry task. The participants spent much time for looking at the sheet with the gestures for the single digits. As already shown in the previous user studies, a stroke within a gaze gesture needs about 400 to 500 milliseconds (a little bit more than 100 milliseconds for the saccade and around 300 milliseconds for the fixation) and entering a digit with four strokes takes about 2 seconds. A four-digit PIN with one second break between the inputs of the digits will last around 10 seconds. Indeed, there were participants in the study who entered the PIN correctly within 14 seconds. It needs a further study to find out whether all users can achieve this time once they are trained for gaze gesture input.

In addition to the absence of a calibration process, the big advantage of the gaze gesture method is its robustness against input errors. Due to the abandonment of feedback for enhanced security, each wrong gaze leads to an incorrect PIN entry when using the dwell time or look & shoot method. This leads to high error rates for these methods. When using the gestures, a wrong gaze leads most probably to an unrecognizable gesture and not to an entry of a wrong digit. For gaze gestures the errors one level below the digit entry, i.e. at the gesture recognition level.

The main reason why a gesture performed by a user is not recognized by the system is a lack of exactness in the hand-eye coordination. As a button has to be pressed and held while performing the gesture, often an additional stroke was detected directly before or after the proper gesture. The reaction time for the finger, typically 300 ms is long compared to the time of a saccade, typically 100 ms. These unintended upstrokes or tails could be filtered out by the algorithm and improve the recognition rate.

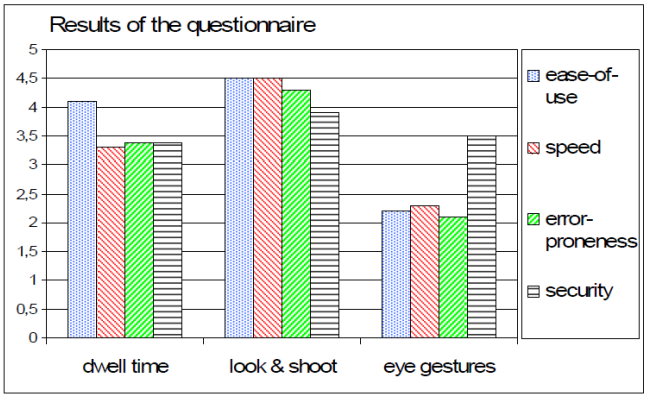

Fig. 9 Results of the questionnaire on a scale from 1 to 5

Figure 9 presents the results of the questionnaire. The users prefer the look & shoot method – it is easier and quicker but also more error-prone than the gaze gestures.

IV. Experiment-2:Reading Detection

Reading is a very important activity when using a computer. Today, many researchers in the field of human computer interaction are interested in reading. One possibility to detect whether a text was read is to use the reading detection mentioned above, collect the x- and y-coordinates, and find out whether these coordinates completely cover the text area. This will result in a rather complex algorithm and as the error rate for reading detection in the mentioned research is still above 10 percent; such an algorithm will not produce reliable results. One algorithm presented here provides feedback to the user and therefore achieves a perfect detection with the help of the user. The other algorithm gives numerical values on how intense a text was read. It is worth mentioning that these algorithms work fine with standard sized fonts, i.e. with font heights which are much smaller than the accuracy limit of the eye.

A. Analysis of the Gaze Path while Reading

Recording the gaze path in initial experiments resulted in patterns as shown in Figure 10.

Fig 10 Gaze path reading a text

It is easy to see that the gaze moves in saccades with brief fixations. Revisits can occur especially in difficult texts. A preliminary evaluation of the data gave saccade lengths of around 30 pixels within a line and backward saccades which are slightly smaller than the length of the line. The geometry of the system results in the 1° visual angle being 36 pixels wide. Thus, the angle of the forward saccades is about the same as the angle of the fovea or the angle of accuracy. It is interesting to note that in consequence the first fixation points in a line are about half a degree behind the beginning of the line and the last fixation is the same distance away from the end of the line. In the vertical direction the low accuracy causes the problem that it is not possible to detect reliably which line the gaze is reading if the line height is within a normal range.

B. An Algorithm for Reading Detection

The reading detection algorithm uses the fact that the gaze visits the text in sequential order. The basic parameters of the algorithm are the width *w* and the height *h* of a rectangle around text points and the distance *d* of text points. The height *h* should be a little bit bigger than the accuracy so that the eye tracker will catch the gaze position when reading the line. The width *w* must not be smaller than the distance *d* of text points to cover the line completely and should be bigger than a typical forward saccade so that the saccade does not jump over it. The algorithm starts with a text point which is *d*/2 from the beginning of the first line. As soon as the reported gaze position is within a rectangular area around the current text point, the text point is marked as read. Then the algorithm chooses the next text point on the same line, *d* ahead of the previous one. In case of a line wrap, the next point is at *d*/2 from the beginning of the next line. A line wrap happens when the next text point in the line is less than *d*/2 away from the end of line (or behind the end of line). This rule in the algorithm is crucial, as people tend to skip the end of the line even when they read carefully. Once the gaze visited all text points the text was read completely. See Figure 12 for an illustration.

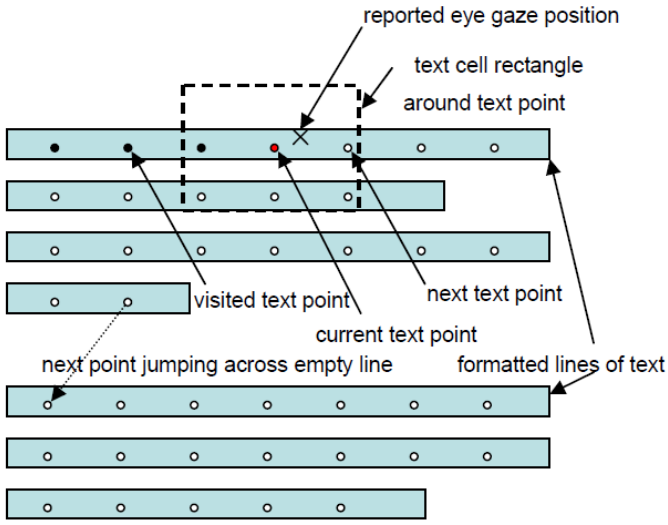

Fig. 12 Illustration of the reading detection algorithm

By design, this algorithm allows the reader to take a break, perhaps to have a look at the system clock, before continuing to read. It is also robust against calibration shift because the dimensions of the rectangle can be relatively large (compared to a calibration shift) without decreasing the reliability. A problem of this algorithm is the case that some words or lines are left out by the reader. In this case the algorithm 'hangs' that means the algorithm waits for the left out word to be read and ignores all further reading. For this reason the system should provide feedback on the portion of text being read for instance by a change of the background color. As mentioned already it is important not to have text points close to the end of the line, because such ends of the line would be places where the algorithm hangs. In Figure 13 it is visible that some lines exceed the coverage by the text cells.

The interesting aspect of the algorithm is that it works with standard font sizes, i.e. with font heights below the accuracy limit. This property is a result of the demand that the height of the text cell has to be bigger than the accuracy of the eye. In consequence the algorithm marks a text cell as read even if the gaze is a little bit above or below the line. The ambiguity between the current text line and the upper or lower text line dissolves by the concept of a current text point which implies also a current line.

A problem arises with very short texts which can be read with one glance. To detect reading in a line of text there should be at least two or three fixations. Together with the fact that the first and last fixation points are about 0.5 degrees from the beginning and end of the line and that the fixations steps are around one degree, a line should have a minimum length of 3° visual angle. This is about 100 pixel on the system used.

As traditional reading always works in sequential order independent of the script used, the algorithm can be applied to right-to-left scripts (Arab) and to vertical scripts (traditional Chinese) too (see Figure 13).

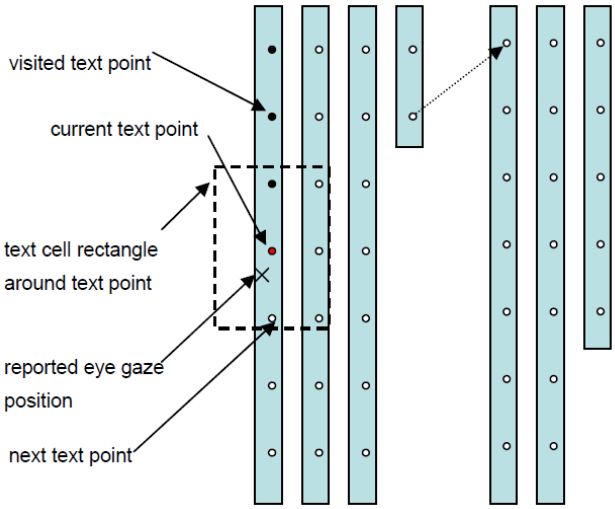

Fig 13 Illustration of the reading detection algorithm for vertical scripts

It is worth to emphasize that the algorithm only ensures that the gaze visited all text points, but does not tell whether the reader understood the text. It is possible that the eye read the text mechanically while the mind is absent. It is also possible that a user reads a text in a language she or he does not understand if the system demands it.

C. User Study for Reading Detection

For the conduction of a user study special software was written including the implementation of the reading detection algorithm. The width *w* of the rectangle was set to 100 pixels, the height *h* to 50 pixels, and the distance *d* to 100 pixels. The geometry of the setup and the conversion of pixels in degrees. With the short lines presented in the study there were only two text points in a line.

A group of nine people, six male, three female, aged from 23 to 47, took part in the user study. Every subject got three tasks. The first task presented a dialog box with a poem of Edgar Allen Poe (23 lines, 150 words). In the first task the dialog presented feedback for the text already read (see Figure 14). The dialog box disappeared when text was totally read. The task was to read the text to make the dialog disappear. The second task was another dialog box with a poem in the native language of the user (13 lines, 69 words), this time without providing feedback. The third task was a repetition of the second task but this time with the demand to read the poem loudly.

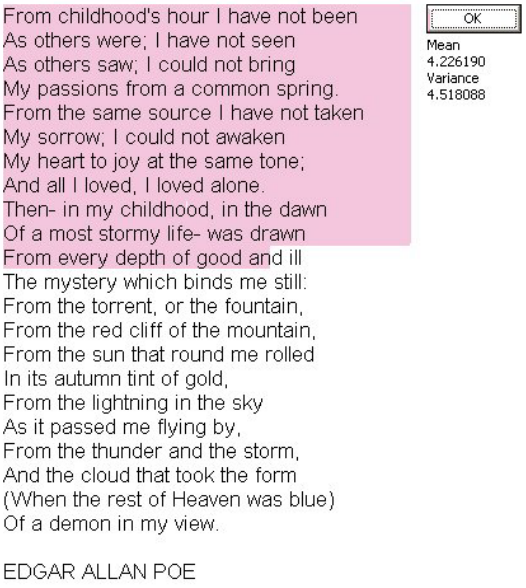

Fig. 14 Feedback for compulsory reading

In the first task all test users were able to let the dialog disappear (100%). Some of them complained that the feedback was faster than their reading and they had the feeling of being chased. Consequently, further research should test feedback with delay. There are two possibilities to achieve this. One possibility is a delay based on time. The other possibility is not to provide feedback for the text cell currently read, but for the text cell read before (and special treatment for the very last text cell).

In the second task, the dialog detected correctly complete readings in six cases (67%) and failed in the other three cases (33%). In the third task, the dialog correctly detected complete readings in five cases (56%) and failed in the other four cases (44%).

The results of the second task were much better than expected while the third task was worse than expected. Without visual feedback the expectation was that users do not read carefully and the algorithm will not complete. Because the task given was to read the text, the users did this carefully. Perhaps this would not be the case if another task had been given for which it is necessary to read something. The third task, reading loudly, had the intention to force the users to read the text slowly and carefully. However, the chin rest for head fixation was an obstacle for speaking and the users moved their heads, which had a negative effect because the eye tracker calibration assumes a fixed head position.

A detection rate of 67% is sub-optimal and consequently compulsory reading should provide feedback, where the success rate is 100%.

D. Values for the Quality of Reading

Detecting whether a text was read carefully or just skimmed needs another approach. A heat map is the typical way to visualize how good a text was read but it needs a human mind to interpret. A programmatical approach needs a method to reduce the data to a few values.

The proposed algorithm starts with formatting the text to find text points and surrounding rectangles. Every fixation inside the rectangle increases a counter for this area. It is also possible to accumulate the fixation time. This mechanism is a kind of a low-resolution heat map and illustrated in Figure 80. The data allows calculating two values – a mean, which tells the number of fixations per text cell, and the variance or standard deviation, which tells how uniformly the fixations are distributed over the text cells.

If there are n text cells and $f_1, f_2, \ldots, f_n$ are fixations for each cell the intensity of reading I can be defined by the mean and a normalizing factor c:

$$I = \frac{c}{n}\sum_{i=0}^{n} f_i$$

Using the mean makes I independent of the length of a text. The factor c adjusts the value of I, so that it will be 1.0 if a text was read carefully once, and depends on the size of the text cell used in the algorithm. If the vertical extent of the text cell is the height of a text line, the horizontal extent is the distance of the text points, and distance of the text points is an average length of a forward saccade, then c is 1. If the text cells are bigger, several fixations can fall into the text cell, and if cells overlap, one fixation is counted multiple times because it is inside several text cells. The factor c will correct this to get the value 1 for a complete read.

To decide whether a text was read carefully I is not enough because reading half of the text twice results in the same value as reading the whole text once. Consequently, a second parameter D is necessary to know whether the gaze positions are distributed equally over the text cells. The variance provides such a parameter.

$$D = \frac{1}{n}\sum_{i=0}^{n} (cf_i - I)^2$$

If the text cells were all visited equally D will be close to zero, otherwise D will be higher. A perfect read will result in $I = 1$ and $D = 0$.

It is possible to visualize the quality of reading by coloring the background of the text cells according to the number of fixations counted for each text cell. Text cells with a high number of fixations get a dark background color while text cells with a low number of fixations get a light one. The result is a so-called heat map but with a resolution of a text cell and not with a pixel. Figure 15 shows examples for such visualizations.

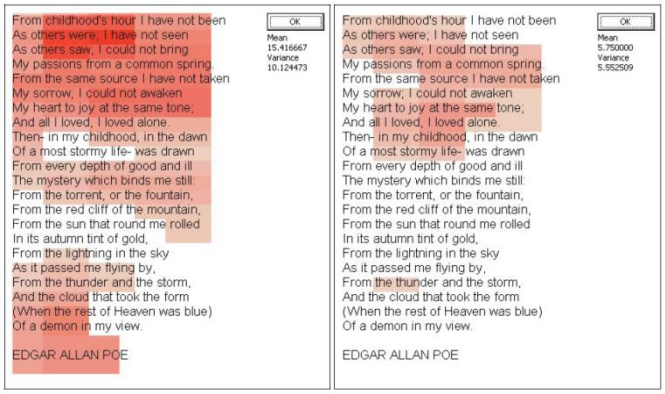

Fig 15 Examples for the quality of reading when skimming a text
Lessons learned:

V. Summary of the Results

The user studies showed that people are able to perform gaze gestures. Helping lines are not helpful and should be replaced by helping points which displays naturally provide with the four corners. Gaze gesture detection works over some distance and can serve as a remote control. Gaze gestures can easily be separated from natural eye movements if they are performed on a big scale. Performing gaze gestures on a big scale is easy on large displays where the corners provide helping points.The other solution for small displays is the use of gaze gestures only within a context. If the gesture detection is only active when the device expects an answer, an accidentally performed gaze gesture will not disturb general interaction.

One stroke of a gaze gesture needs about 500 milliseconds. With at least four strokes for the gesture, it is obvious that gaze gestures are not well suited for text entry as they are slower than other gaze-based methods.
Reading detection is an example of eye-gaze context information which is not trivial, but also not too vague. In contrast to the reading detection algorithms published so far, which detect reading as activity, the algorithm presented here detects whether a particular piece of text was read. The remarkable property of the reading detection is that it works quite well even with line heights below the eye tracker's accuracy. That means that the reading detection works with the standard font sizes used in existing GUIs. It is in the human nature to skip some words when reading a lengthy text quickly. To deal with this human behavior the second algorithm provides numerical values for how good a text was read. For the same reason, a system should provide feedback on what text was read already when demanding to read the text carefully and complete. The user studies showed that a direct feedback disturbs the reading process. The users reported that they felt chased by the immediate feedback. Therefore, future research should try feedback appearing with a delay.

VI. Conclusion

Eye gaze interaction is a useful source of additional input and should be considered when designing advanced interfaces in the future. Moving the eyes is natural, requires little conscious effort, and frees the hands for other tasks. People easily gaze at the world while performing other tasks so eye gaze combined with other input techniques requires little additional effort. An important side benefit is that eye position implicitly indicates the area of the user's attention. We argue for using natural eye movements and demonstrate interaction techniques based on an understanding of the physiology of the eye. Our algorithm extracts useful information about the user's higher-level intentions from noisy, jittery eye movement data. Our approach is successful because it preserves the advantages of the natural quickness of the eye. We presented two experiments that demonstrate that using a person's natural eye gaze as a source of computer input is feasible.

The intention behind Section 3 was to try an alternative approach to direct the computer. In summary, the user studies show that the gaze gestures presented here work quite well. The gaze gestures are a special purpose gaze-only interface and not as intuitive as eye gaze pointing. The big advantage of gestures is that they do not require a calibration procedure, allowing instant use by different people. Whether this would be accepted by the masses as an interface – for example as remote TV control – is still an interesting question.

The approach in section IV is to use unintentional eye movements to provide context information. It leaves the concept of directing the computer intentionally with commands given by the eye. The computer analyses the movements of the eye to give smart assistance to the user. Chapter VI had the following outcome:

Eye gaze provides valuable context information. For the aim to provide human-computer interfaces, which work similar to human-human interaction, the information from the eye gaze is essential. Context information in general is still a vague concept in a range from trivial applications to social intelligence. The simplest form of context information from the eyes is attention.

VII. References

1. [Kumar, Paepcke, Winograd 2007] Kumar, M., Paepcke, A., and Winograd, T. EyePoint: Practical Pointing and Selection Using Gaze and Keyboard. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems CHI '07. ACM Press (2007), 421 – 430.
2. [Kumar, Garfinkel, Boneh, Winograd 2007] Kumar, M., Garfinkel, T., Boneh, D., and Winograd, T. Reducing Shoulder-surfing by Using Gaze-based Password Entry. In Proceedings of the 3rd Symposium on Usable Privacy and Security SOUPS '07, vol. 229. ACM Press (2007), 13 – 19.
3. [Jacob, Karn 2003] Jacob, R. J. K., and Karn K. S. Eye Tracking in Human-Computer Interaction and Usability Research: Ready to Deliver the Promises (Section Commentary). In The Mind's Eyes: Cognitive and Applied Aspects of Eye Movements, ed. by J. Hyona, R. Radach, and H. Deubel, Oxford, Elsevier Science, 2003
4. [Jönsson 2005] Jönsson, E. If Looks Could Kill – An Evaluation of Eye Tracking in Computer Games. Master Thesis (2005), Royal Institute of Technology, Stockholm, Sweden
5. [Majaranta, Aula, Räihä 2004] Majaranta, P., Aula, A., and Räihä, K. Effects of Feedback on Eye Typing with a Short Dwell Time. In *Proceedings of the 2004 Symposium on Eye Tracking Research & Applications*. ETRA '04. ACM Press (2004), 139 – 146.
6. [Majaranta, MacKenzie, Aula, Räihä 2003] Majaranta, P., MacKenzie, I. S., Aula, A., and Räihä, K. Auditory and visual feedback during eye typing. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*. CHI '03. ACM Press (2003), 766 – 767.
7. [Mankof, Abowd 1998] Mankof, J., and Abowd, G. D. Cirrin: a word-level unistroke keyboard for pen input. In *Proceedings of the 11th Annual ACM Symposium on User interface Software and Technology* UIST '98, ACM Press (1998), 213 – 214.
8. [McGuffin, Balakrishnan 2002] McGuffin, M. and Balakrishnan, R. Acquisition of Expanding Targets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* CHI '02. ACM Press (2002), 57 – 64.
9. [Thomae, Plagwitz, Husar, Henning 2002] Thomae, N., Plagwitz, K.-U., Husar, P., and Henning, G. Hough-Transformation zur Bildverarbeitung bei der Blickrichtungsbestimmung. In Proceedings 36. Jahrestagung der Deutschen Gesellschaft für Biomedizinische Technik. Biomedizinische Technik, Band 47, Ergänzungsband 1, Teil 2, (2002), 636 – 638.
10. [US 6578962] Calibration-free eye gaze tracking. US Patent Issued on June 17, 2003 [Vertegaal 2008] Vertegaal, R. A Fitts Law comparison of eye tracking and manual input in the selection of visual targets. Conference on Multimodal interfaces IMCI '08. ACM (2008.) 241 – 248.
11. [Vertegaal, Dickie, Sohn, Flickner 2002] Vertegaal, R., Dickie, C., Sohn, C., and Flickner, M. Designing Attentive Cell Phone Using Wearable EyeContact Sensors. In *CHI '02 Extended Abstracts on Human Factors in Computing Systems*. CHI '02. ACM Press (2002), 646 – 647.
12. [Vertegaal, Dickie, Sohn, Flickner 2002] Vertegaal, R., Dickie, C., Sohn, C., and Flickner, M. Designing Attentive Cell Phone Using Wearable EyeContact Sensors. In CHI '02 Extended Abstracts on Human Factors in Computing Systems. CHI '02. ACM Press (2002), 646 – 647.
13. [Vertegaal, Mamuji, Sohn, Cheng 2005] Vertegaal, R., Mamuji, A., Sohn, C., and Cheng, D. Media EyePliances: Using Eye Tracking For Remote Control Focus Selection of Appliances. In CHI '05 Extended Abstracts on Human Factors in Computing Systems. CHI '05. ACM Press (2005), 1861 – 1864.

14. [Vertegaal, Slagter, van der Veer, Nijholt 2001] Vertegaal, R., Slagter, R., van der Veer, G., and Nijholt, A. Eye Gaze Patterns in Conversations: There is More to Conversational Agents than Meets the Eyes. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '01. ACM Press (2001), 301 – 308.

15. [Wade, Tatler, Heller 2003] Wade N. J., Tatler B. W., and Heller D. Dodge-ing the issue: Dodge, Javal, Hering, and the measurement of saccades in eye-movement research. Perception (2003) 32: 793 – 804. (as referenced in [Schneider@])

16. [Wagner, Bartl, Günthner, Schneider, Brandt, Ulbrich 2006] Wagner, P., Bartl, K., Günthner, W., Schneider, E., Brandt, T., and Ulbrich, H. A Pivotable Head Mounted Camera System that is Aligned by Three-Dimensional Eye Movements. In Proceedings of the 2006 Symposium on Eye Tracking Research & Applications. ETRA '06. ACM Press (2006), 117 – 124.