Interacting with the Computer Using Gaze Gestures

Heiko Drewes[1] and Albrecht Schmidt[2]

[1] Media Informatics Group, LMU University of Munich,
Amalienstraße 17, 80333 München, Germany
heiko.drewes@ifi.lmu.de
[2] Fraunhofer IAIS and B-IT, University of Bonn,
Schloss Birlinghoven, 53754 Sankt Augustin, Germany
albrecht.schmidt@acm.org

Abstract. This paper investigates novel ways to direct computers by eye gaze. Instead of using fixations and dwell times, this work focuses on eye motion, in particular gaze gestures. Gaze gestures are insensitive to accuracy problems and immune against calibration shift. A user study indicates that users are able to perform complex gaze gestures intentionally and investigates which gestures occur unintentionally during normal interaction with the computer. Further experiments show how gaze gestures can be integrated into working with standard desktop applications and controlling media devices.

Keywords: eye-tracker, gaze gestures.

1 Introduction

Eye-trackers are video-based, and the cost of video cameras has dropped substantially in the past few years. Commercially available eye-trackers work with a resolution of 640 x 480 pixels and this is the resolution of a web cam, which can be bought as a consumer device for a few dollars. Most mobile devices sold today have a built-in camera and there are already the first laptops and desktop computers with built-in camera on the market. It is foreseeable that future monitors will have an integrated camera for no extra cost, as it is the case for integrated speakers today.

The processing power of a standard computer is sufficient to do real-time processing of a video stream. This means that within the near future eye-tracking technology will be available for no extra costs. There are already projects for low-cost or off-the-shelf eye-trackers [1], [2].

There are eye-tracker systems for disabled people to direct the computer and it is imaginable that eye-tracking could become an additional input modality for everybody. But the systems for the disabled are cumbersome to operate and less efficient compared to the classical way of interaction with keyboard and mouse. For this reason researchers from the field of human computer interaction think about new interfaces utilizing the eye-gaze.

1.1 Eye-Tracking

Quantitative research on eye-movements became possible with the invention of the motion camera and the first research dates back to this time. This kind of research was

C. Baranauskas et al. (Eds.): INTERACT 2007, LNCS 4663, Part II, pp. 475 – 488, 2007.

mostly done by psychologists who wanted to understand perception. Most eye-tracker systems were built for analysis of the eye movement and its application in fields like advertisement. The first ideas to use the eye-gaze for interaction with the computer date back to the early 80s and 90s [3] [4] [5]. This was the time when it became possible to process a digital video stream in real-time. For an overview on eye-tracking see [6].

The technological basis of nowadays eye-tracking is easy to understand. An infrared LED causes a reflection spot on the eyeball. As the eye is perfectly round, the reflection spot stays at the same position no matter in which direction the eye is looking. A video camera detects the reflection spot and the center of the pupil. The direction of the eye-gaze can be calculated from the distance of both points by simple linear mapping.

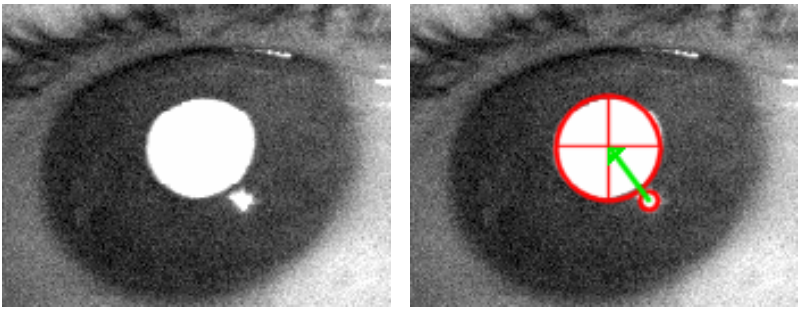

Fig. 1. Video-based eye-tracking uses the reflection of an infrared LED and the center of the pupil to calculate the direction of the eye-gaze. The reflection spot stays in the same position, while the pupil moves.

After a calibration procedure, for example gazing at the four corners of the display, the eye-tracker can deliver screen coordinates to the computer. The method requires that the head stays in the same position. Consequently, such systems need a head fixation or at least a chin rest, but this is no problem for the disabled person who can't move anything except the eyes. A typical commercial system of such an eye-tracker is the ERICA[1] system, which we used for our research.

To achieve freedom of movement in front of the display a head-tracker is necessary. Normally this is done by a second video camera. Such systems are more expensive, but also commercially available, for example the EyeGaze Eyefollower[2] or Tobii 1750 Eye Tracker[3].

1.2 The Problems of Eye-Gaze Based Interaction

Eye-gaze based interaction is now available for more than 20 years [7], but it is solely used in the domain of accessibility. Systems for eye-gaze based interaction typically

[1] http://www.eyeresponse.com
[2] http://www.eyegaze.com
[3] http://www.tobii.com

display a keyboard layout on the display and to enter a character the user has to gaze for a certain time, the dwell time, on the corresponding key. The time which could be saved by the proverbial quick movement of the eyes is eaten up by the dwell time. Reducing the dwell time leads to the Midas-Touch problem – inspecting the display causes unwanted actions [5].

The accuracy problem, which is not only a question of the resolution of the video camera but is intrinsic because of the jitter in the eye-movement, leads to big sizes for the keyboard layout. This causes a space problem on the display.

A general problem is the fact that the eye is mainly an input sensor and not an output actor. The eyes move to see something and not to trigger actions. Using the eyes for both, input and output, may result in conflicts [8]. On the other hand we can communicate with other persons by the direction where we look. As we know that other persons are aware of where we are looking, we keep our eyes under control as a part of our social protocol. The question how much output activity we can put on the movement of the eyes and how much unintentional eye movements interfere with intentional eye movements is not clear yet.

1.3 The Concept of Gestures

Gestures are a well-known concept for computer human interaction. Typical examples are Unistroke [9] and Cirrin [10]. A gesture consists of a sequence of elements, typically strokes, which are performed in a sequential time order. The advantage of gestures is that the number of commands can be increased by increasing the set of gestures. If commands are selected from a list, the increase of commands results in a bigger list und this can cause a space problem. For this reason gestures are used for interaction with small devices. As one problem of eye-gaze interfaces mentioned above is a space problem, gaze gestures are worth examining, especially for interaction with small displays.

1.4 Related Work

Most approaches to utilize the gaze position for computer control follow the concept of gaze as a pointing device and as an alternative for mouse input. There is only little research on different concepts like gestures.

Most work on gestures aims to identify the user's task or attention and use this as context information for smart interfaces. Qvarfordt and Zhai used eye-gaze patterns to build a dialog system [11]. They studied gaze patterns in human-human dialogs and used the results to mediate a human-computer dialog. In contrast to our approach the users did not learn gaze gestures to operate the system. The users were not even aware that they perform gestures.

Isokoski proposed the use of off-screen targets for text input [12]. To enter text the eye-gaze has to visit the off-screen targets in a certain order. The eye movements resulting from this are gaze gestures. The difference to the gaze gestures presented in this paper is that off-screen targets force the gesture to be performed in a fixed location and with a fixed size. The gaze gestures researched in our user study are scalable and can be performed in any location.

2 The Gaze Gesture Algorithm

As there was not much research done on gaze gestures the main research question is whether people are able to perform complex gestures with the eyes. The question which algorithm to use is secondary until the first question is answered.

2.1 Searching for a Gaze Gesture Algorithm

The popular and freely available mouse-gesture plug-in[4] for the Firefox web browser inspired us 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. Horizontal and vertical directions are given by the letters U, D, L, and R for up, down, left and right respectively. Diagonal directions are given by 1, 3, 7, and 9 corresponding to the familiar layout of the number pad on a standard keyboard. A gesture is defined by a particular string consisting of these eight characters.

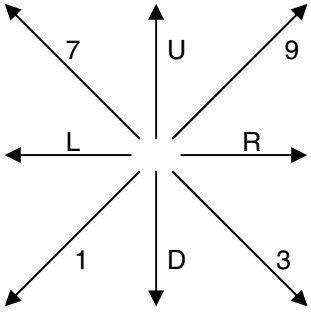

Fig. 2. The names of the eight directions of a stroke

The stroke detection uses a grid of size s. The algorithm maps every point reported by the mouse to a point on the grid by a simple integer division. The origin of the grid is the starting point of the stroke. If the integer division has a result different from zero for at least one of the coordinates, the algorithm calculates the direction. It outputs the corresponding character, but only if it is different from the character before.

The other gesture algorithm which inspired our work is EdgeWrite [13]. This algorithm starts with the four corners of a square and the six connecting lines. A stroke is a move from one corner to another corner and a gesture is a series of connected strokes. It is easy to see that all EdgeWrite gestures can be expressed with the tokens from the mouse gesture algorithm and consequently are a subset of the mouse gestures. This is interesting because the EdgeWrite gestures have the capability for a big complex alphabet.

[4] http://optimoz.mozdev.org/gestures

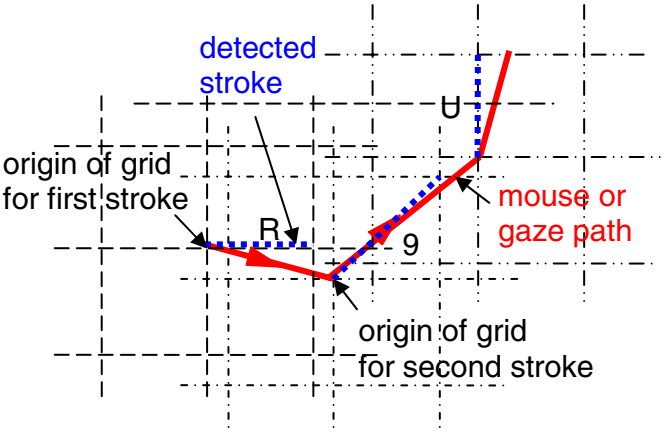

Fig. 3. The figure shows how a mouse or gaze path is translated into the string of characters R9U. The end point of a detected stroke is the origin for the grid to detect the next stroke.

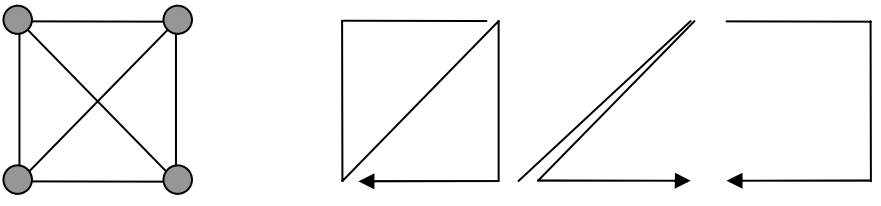

Fig. 4. The four corners and the six connecting lines used for the EdgeWrite gestures and three examples for EdgeWrite gestures (digits 0, 2 and 3)

2.2 Implementing a Gaze Gesture Algorithm

We liked the simplicity of the mouse gesture algorithm, but we disliked the need of a gesture key. So the first modification of the algorithm is the introduction of continuous recognition. Consequently the algorithm must divide between natural eye movements and gestures. The situation is similar for detecting commands with speech recognition.

To better separate the gestures from the natural movement we introduced a time aspect. During the performance of a gesture only short fixations and no long fixation should occur. A long fixation should reset the gesture recognition. We extended the algorithm with timeout detection and introduced a colon as the ninth token. The algorithm generates a colon as output if no other token was generated for time t.

The eye-gaze gesture recognition algorithm is a two-step process. In the first step the algorithm takes the x-y position of the current eye-gaze in pixels and maps this to strokes as described above. In the second step the algorithm recognizes the actual gestures by comparing the string to the given gesture pattern string. If the gesture pattern matches an action can be triggered.

The software is implemented for the Windows platform and the software is written in C++ with Visual Studio.

3 User Study

We conducted a user study with nine participants, six male and three female persons in the age from 23 to 47 years. All persons had a European cultural background and academic education. All of them used computers regularly, but none of them had experience with eye-tracking systems.

3.1 Experimental Setup

For the user study we used the commercial eye-tracker ERICA. The system consists of a camera and a tablet PC mounted together on a stand. The display has a size of 246 mm x 185 mm and a resolution of 1024 x 768 pixels. The distance of the eyes from the screen is 48 cm ± 2 cm. This values result in 0.028° visual angle per pixel or around 36 pixels for 1°. The accuracy of the system is ± 0.5°.

The ERICA system delivers a maximal update rate of 60 Hz or about one position every 17 milliseconds. During the movement of the eye no data are delivered. This results in a gap during a saccade.

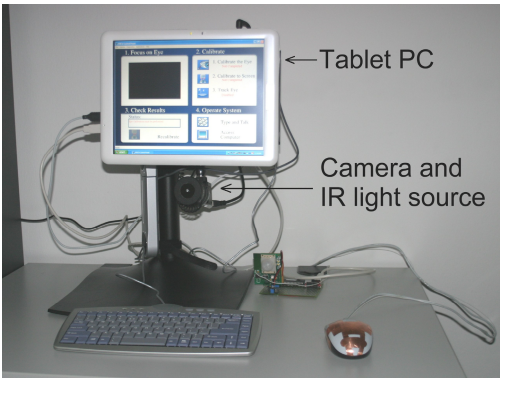

Fig. 5. The ERICA eye-tracker system used for the user study

The software written for the user study did the gesture recognition. The grid size was set to 80 pixels and the timeout parameter t was 1000 milliseconds. The program gave auditory feedback i.e. prompted the recognition of a gesture with a beep. It also had options to display helping lines or blank or structured background. The structured background was a screenshot of a spreadsheet application.

3.2 Design of the User Study

The user study consisted of three different tasks. Prior to the experiment the participant got a brief introduction to the system.

The first task was to close a dialog by using eye-gestures instead of the mouse. The participants were instructed to perform the action by visiting the corners clockwise for YES and counter-clockwise for NO. (Could also be OK and CANCEL). The gaze gesture recognition scanned for the patterns RDLU, DLUR, LURD and URDL for YES and for the patterns LDRU, DRUL, RULD and ULDR for NO. The time needed for the operation was recorded.

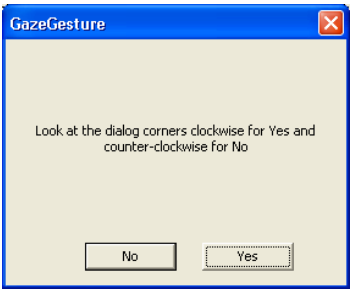

Fig. 6. The first task in the user study was to close a dialog with a gaze gesture

In the second task the users had to do three different gaze gestures of increasing difficulty on three different backgrounds. Again the software logged the gaze activity. To prove that the user is able to do the requested gesture, each gesture had to be repeated three times. This resulted in 27 gestures per candidate. The gestures used were RLRLRL, 3U1U and RD7DR7, see Figure 2 for an illustration. For each performed gesture the required time was recorded.

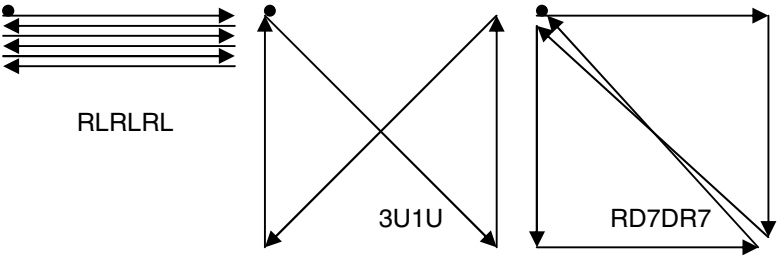

Fig. 7. The three gaze gestures shown had to be performed for the second task in the user study

The first background showed an outlined square with diagonal lines as shown in Figure 8. The helping lines were given to guide the gaze gestures. The second background was a screenshot of a desktop with an open spreadsheet document. This enabled the test users to choose positions for fixations. And the third background was just gray.

The third task was to surf the internet for three minutes. The gesture recognition software logged the resulting gesture string. The reason for this task was to find out which patterns occur during normal work, or at least during surfing.

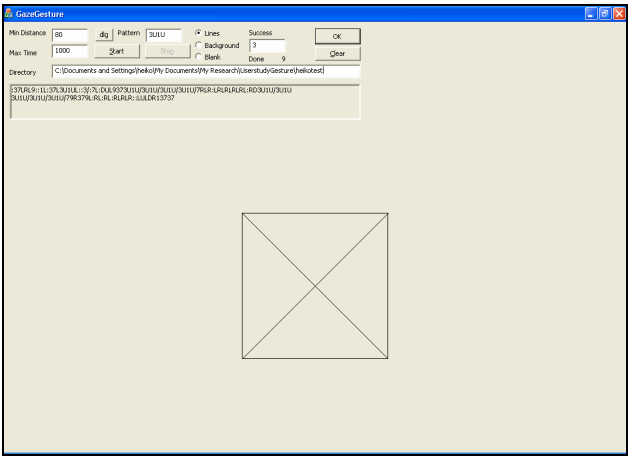

Fig. 8. Screenshot of the program written for the user study. The display modes are blank, structured background and display of helping lines.

3.3 Results of the User Study

All users were instantly able to close the dialogs by gaze with YES and NO using eye-gestures. The average time to perform the gesture was about 1900 milliseconds with a standard deviation of about 600 milliseconds. This time is in the same range than performing the action by mouse including a homing from keyboard to mouse and clicking a button. All participants reported this as an easy task.

Table 1. Average time to perform the gesture for closing the dialog in the first task

Gesture	**Gesture time** average over all subjects (ms)
YES (clockwise all corners)	1905
NO (counter-clockwise all corners)	1818

In the second task, where the participants had to perform the three different gestures, we were surprised that all users were able to perform all gestures on the helpline and text background, most of them with ease. The number of attempts to complete a gesture varied very much. In many cases users were able to perform the gestures instantly whereas for some others it took quite long to complete the task successfully. Table 2 shows the task performance times for the 3U1U gesture.

For the blank background all users could accomplish the gestures RLRLRL and 3U1U. Five of nine users were even able to perform the most difficult task (RD7DR7 on a blank background). In some of these cases it initially took quite long to get the gesture, but after the first success it took not much time to repeat the gesture again.

Table 2. Total time in milliseconds to perform three times the 3U1U gesture

Participant	Helping Lines	Text Background	Blank Background
P1	26808	30795	23915
P2	33528	28400	25407
P3	5899	35611	23513
P4	160370	38506	74567
P5	25106	33177	97240
P6	10355	9353	15022
P7	12789	60708	71633
P8	26849	32477	10926
P9	23724	114074	56722
Mean	**36159**	**42567**	**44327**
Std. Dev.	47452	29874	31216

Overall we learned that with a structured background such as text, tables or web pages, even difficult gaze-gestures can be performed reliably and that neither the background nor the complexity of the gesture has a significant impact on the completion time. The time for the gesture depends only on the number of segments. The average time required for a segment was 557 milliseconds.

Table 3. Average gestures time and standard deviation in milliseconds to perform the three different gestures on three different backgrounds in the second task. The data are from nine participants, except RD7DR7 on blank background, where only 5 participants were able to perform the gesture.

Gesture	Helping Lines	Text Background	Blank Background
RLRLRL	3113 (±627)	3089 (±728)	3288 (±810)
3U1U	2222 (±356)	2311 (±443)	2429 (±307)
RD7DR7	3163 (±490)	3563 (±651)	3569 (±520)

The third task recorded the characters produced by the gaze gesture recognition algorithm while surfing. The total time for the 9 users was 1700 seconds or 28 minutes, resulting in 2737 characters. This results in 1.6 characters per second or about 600 milliseconds for a stroke. This string was searched for the gestures of the first and second task.

Table 5 shows the occurrence of the gestures from the first and second task. Some of the eight gestures to enter YES or NO respectively in a dialog did not occur in the whole string and others at most 3 times. When restricting the recognition to the context of use (e.g. currently a dialog is open) the risk to answer a dialog unintentionally seems to be extremely low.

Table 4. Statistics for detected strokes within half an hour of web surfing

Stroke	Occurrences	Percentage	Stroke	Occurrences	Percentage
:	388	14,1%			
1	136	5,0%	D	178	6,5%
3	136	5,0%	U	229	8,3%
7	138	5,0%	L	685	25,0%
9	115	4,2%	R	732	26,7%

The RLRLRL gesture occurs very often (69 times in the sample of all participants), because this is the natural eye movement during reading and consequently should not be used for commands in general. The 3U1U and RD7DR7 gestures didn't occur during the half hour of surfing. In particular the 3U1U gesture seems to be a good candidate for a gesture that is generally applicable, relatively easy to perform, and very unlikely to appear during normal use.

Table 5. Occurrence of the gestures from task 1 and 2 within half an hour of web surfing

Gesture		Gesture		Gesture	
RDLU	0	DRUL	2	RLRLRL	69
DLUR	2	RULD	3	3U1U	0
LURD	1	ULDR	0	RD7DR7	0
URDL	1	LDRU	1		

4 Experiments with Standard Applications and Media Devices

After the positive results from the first user study, the next step was to look for fields of application for this novel type of interaction. The EdgeWrite gestures provide a full alphabet, but the gaze gestures are not adequate for text input. As seen in the user study a gesture needs 1 to 2 seconds to enter and even the standard dwell time method is faster and typing with the fingers is definitely the more efficient way of text input.

A useful application of gaze gestures is the field of accessibility. Because of the robustness against accuracy problems and immunity against calibration shift a gaze gesture is the perfect way to invoke a recalibration process for the disabled users of eye-tracker systems. It is also imaginable to use the gestures for general macro functions within accessibility systems. For example a gaze gesture could be used to save a document and close the application or to paste content from the clipboard.

One idea was to offer the macro functionality as an extra input modality for everybody. For this reason we implemented a software prototype which is able to recognize a list of gestures and trigger a corresponding command. We used the WM_APPCOMMAND of the Windows operating system to realize an open document, save document and close document function which works with standard

Windows software such as Word. When observing people working with documents and application we noticed that many users put the hands to the mouse to select the save option from the menu and return the hand to the keyboard for further text entry. With the use of gaze gestures it is imaginable to leave the hands on the keyboard - saving the lengthy time for homing and selection – and invoke the save operation with the eyes.

To test the idea we put some colleges, not involved in our research, in front of our system and asked them to type something and save the document with the gaze gesture. They were instantly able to perform the operation asked for. They told us that there is some fascination in the possibility to direct the computer without touching, but for not to grab the mouse they would press the short cut ctrl-s to save their document. The returned to their office and saved the next document by using the mouse. Of course every command invoked by a gaze gesture could also be invoked by a key press. Whether or how many people would use gaze gestures if offered as standard interaction is not clear.

This result motivated us to look for further applications. Gaze gestures could be useful in the case that keyboard and pointing device are out of reach. This situation is typical for controlling media devices, especially media center computers. Such devices normally come along with remote control.

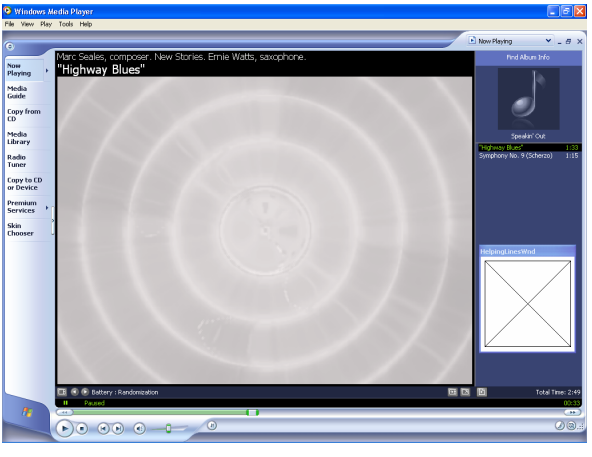

Fig. 9. Screenshot of the media player and a window with helping lines to perform the gestures. It turned out that it is more convenient to use the edges of the main display to enter the gesture and the helping lines are not necessary.

The accuracy of an eye-tracker is given in visual angle. In principle this means, that the spatial accuracy in millimeters or pixels on the screen gets worse with growing distance. But gaze gestures on a big grid are insensitive to accuracy problems and seem to be well suited to the situation.

Thus, we extended our software with additional commands for media control such as play, pause, stop, previous track, next track, media channel up and down and

volume control. To test the system we placed candidates in a distance of one meter away from the display. The one meter distance is the maximum our eye-tracker optics is able to focus and it is longer than the arms of the candidates, so the couldn't reach the keyboard. The observations were encouraging.

The first observation was that people didn't need the helping lines offered. The corners of the display window or the screen provide a natural orientation to perform the gestures. The candidates had no difficulties to perform the gestures.

The next observation was that people experienced it easier to perform big scaled gestures than small scaled gestures. From our recorded data we know that the time needed for a saccade does not increase much if the saccade length gets bigger. A saccade above 5° visual angle lasts about 120 milliseconds (see also [14]). The influence of the scale used for the gesture does not have a big effect on the time to perform the gesture.

It also turned out that the grid size of the gesture algorithm is not critical. The big scaled gestures were reliably detected with the grid size settings for the small scaled gestures. People seem to perform horizontal and vertical eye movements with high precision.

Another observation from this category was the insensitivity of the gesture recognition to the aspect ratio. The gestures do not have to be in square shape. An aspect ratio of 4:3 and 16:9 for the corners also work well. See figure 10 for an illustration.

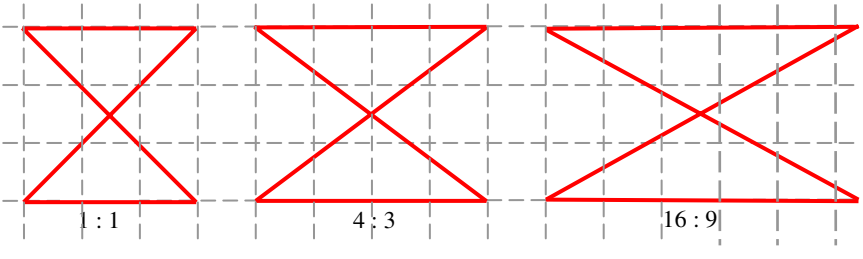

Fig. 10. The gesture algorithm is independent from the aspect ratio

There was already research to use eye-trackers for remote control [15]. Vertegaal et al. used one remote control and eye-trackers on several devices and used the eye-tracker information to find out which device the remote control is meant for. Using the gaze gestures would allow the remote control to be eliminated.

5 Conclusions and Future Work

It seems that the concept of gaze gestures has a potential to be used as an input modality. Gaze gestures solve some of the big problems of eye-gaze interaction. First of all the gaze gestures use only relative eye movements and consequently do not need a calibration of the eye-tracker. Accuracy is not an issue, because the gaze is not used for pointing. As the grid size of the gesture algorithm can be chosen as large as

10° visual angle and the time needed to perform a gesture segment is several hundred milliseconds, the gaze gesture detection does not demand high spatial and temporal resolution from the eye-tracker. This makes it possible to manufacture eye-trackers, which can detect gaze gestures only, with cheap standard devices. Finally the use of gaze gestures does not exhibit the Midas-Touch problem and the users do not feel stressed by not being allowed to look too long at something.

Research on this topic is still in the early beginning and the presented algorithm is not yet fully researched. Further user studies should be done on the question which gestures occur unintentional during normal looking, e.g. watching videos. The parameters of the algorithm, the grid size s and the timeout t, will be a subject for optimization. A bigger grid size up to the dimensions of the display will lead to fewer unintended gestures, because the eye movements normally stay within the display and typical saccade lengths are much smaller than the width or height of the display.

The question whether users will accept gaze gestures as an additional input modality is very interesting. In the field of accessibility the concept of gaze gestures will certainly bring benefit for the user, for example as a substitute of accelerator keys (ctrl-s) or to invoke a recalibration process. In addition to the application as a substitute for remote controls as mentioned above, the gaze gestures can be very useful in fields with high hygienic demands. A surgeon in the operating room could interact with electronic devices using gaze gestures.

It also seems worthwhile to think about alternative gesture algorithms. This will lead to a closer look on the low-level recognition algorithms. The eye-trackers of today are optimized for the detection of fixations and dwell times. Normally the eyes move in saccades, but some people are also able to roll the eyes smoothly.

Our future efforts will focus on gaze gestures on mobile devices, where eye-gaze input is difficult because of the small display size.

Fig. 11. Gaze gestures on small displays

Acknowledgments. The work has been conducted in the context of the research project Embedded Interaction (‘Eingebettete Interaktion’) and was partly funded by the DFG (‘Deutsche Forschungsgemeinschaft’).

References

1. Hansen, D.W., MacKay, D.J.C., Hansen, J.P., Nielsen, M.: Eye tracking off the shelf. In: ETRA 2004, pp. 58–58. ACM Press, New York (2004)
2. Li, D., Babcock, J., Parkhurst, D.: openEyes: a low-cost head-mounted eye-tracking solution. In: ETRA 2006, pp. 95–100. ACM Press, New York (2006)
3. Bolt, R.A: Gaze-orchestrated dynamic windows. In: SIGGRAPH '81, pp. 109–119. ACM Press, New York (1981)
4. Ware, C., Mikaelian, H.H.: An evaluation of an eye tracker as a device for computer input. In: Proceedings of the CHI + GI '87, pp. 183–188. ACM Press, New York (1987)
5. Jacob, R.J.: What you look at is what you get: eye movement-based interaction techniques. In: CHI '90, pp. 11–18. ACM Press, New York (1990)
6. Duchowski, A.T.: Eye Tracking Methodology: Theory and Practice. Springer, New York (2003)
7. Majaranta, P., Räihä, K.: Twenty years of eye typing: systems and design issues. In: ETRA 2002, pp. 15–22. ACM Press, New York (2002)
8. Zhai, S., Morimoto, C., Ihde, S.: Manual and gaze input cascaded (MAGIC) pointing. In: CHI '99, pp. 246–253. ACM Press, New York (1999)
9. Goldberg, D., Richardson, C.: Touch-Typing With a Stylus. In: CHI '93, pp. 80–87. ACM Press, New York (1993)
10. Mankof, J., Abowd, G.D.: Cirrin: a word-level unistroke keyboard for pen input. In: UIST '98, pp. 213–214. ACM Press, New York (1998)
11. Qvarfordt, P., Zhai, S.: Conversing with the User Based on Eye-Gaze Patterns. In: CHI '05, pp. 221–230. ACM Press, New York (2005)
12. Isokoski, P.: Text input methods for eye trackers using off-screen targets. In: ETRA '00, pp. 15–21. ACM Press, New York (2000)
13. Wobbrock, J.O., Myers, B.A., Kembel, J.A.: EdgeWrite: a stylus-based text entry method designed for high accuracy and stability of motion. In: UIST '03, pp. 61–70. ACM Press, New York (2003)
14. Abrams, R.A., Meyer, D.E., Kornblum, S.: Speed and accuracy of saccadic eye movements: Characteristics of impulse variability in the oculomotor system. Journal of Experimental Psychology: Human Perception and Performance 15(3), 529–543 (1989)
15. Vertegaal, R., Mamuji, A., Sohn, C., Cheng, D.: Media eyepliances: using eye tracking for remote control focus selection of appliances. In: CHI '05, pp. 1861–1864. ACM Press, New York (2005)