WHAT IS POOR MAN'S EYE TRACKING GOOD FOR?

Carsten Ullrich
DFKI GmbH
Stuhlsatzenhausweg 3
D-66123 Saarbrücken, Germany
+49 681 302 5370
cullrich@dfki.de

Dieter Wallach
Digital Media Department
University of Applied Sciences
Kaiserslautern, Germany
wallach@ergosign.de

Erica Melis
DFKI GmbH
Stuhlsatzenhausweg 3
D-66123 Saarbrücken, Germany
+49 681 302 4629
melis@dfki.de

ABSTRACT In this paper, we describe a new methodology for tracking a user's areas of interest on a computer screen. In contrast to cost- and analysis-intense eye-tracking studies, our approach operates on a software-only basis that requires no additional hardware. In an exploratory study, we compared our software-based tracker with hardware eye trackers with respect to the types of recorded data. Interestingly, our software-based «poor man's» eye tracker allows the collection of data that would pose several problems for hardware eye tracker. In this paper, we describe a study in which we compared whether both approaches yield similar data. A first evaluation of our results clearly indicates high correlations between the software eye tracker and the hardware system.

Keywords
Usability Testing, Eye Tracking, Empirical Methods, Toolkits

1. INTRODUCTION

In his article "What is Eye Tracking Good for?" Schroeder [5] enumerated the "Can" and "Can't" of hardware eye trackers. At the German Research Center of Artificial Intelligence in Saarbrücken we have developed a software-based (poor man's) eye tracker that overcomes most of the "Can't"s of hardware eye trackers that Schroeder discussed, while still offering almost the same advantages. DFKeye was initially developed for the use in the eLearning environment ACTIVEMATH [4], however we recently started to explore its scope and applicability as a substitute for hardware eye trackers with arbitrary interfaces.

Although we know of two similar approaches to DFKeye, their application presuppose a somewhat limited and artificial experimental setup since they either need participants to perform significant and cognitive demanding extra effort [3] or they rely on the user's compliance to specific experimental instructions [6].

2. DFKEYE: A POOR MAN'S EYE TRACKER

Hardware eye trackers use infrared light reflections on the user's cornea to track the fixations of participants. In this paper, we argue that trailing a user's area of interest can be closely approximated by tracking the user's mouse movement in a proper setup [1], [2].

When using our DFKeye, the screen is divided into several regions. Text areas, Figures, information containers or arbitrary other user interface regions located within these regions are hidden from the user by blurring their content. To make the respective region's content visible, the user focuses it by moving the mouse pointer into this area. As Byrne et al. [1] discuss, there is evidence that users tend to move the mouse pointers to the regions where they are gazing at. Whenever a user enters or leave a certain region, DFKEye records the respective start and end times. From a user's point of view, the visibility of a region is removed after a parameterizable time slot and its content is hidden again. To regain focus, the user needs to move the mouse again.

3. WHAT IS POOR MAN'S EYE TRACKING GOOD FOR?

There are a several reliably functioning hardware eye-trackers on the market — so why can there be a need for a poor man's eye tracker such as DFKeye? Besides the disadvantages of the enormous costs for buying and maintaining a hardware eye tracker, we also found restrictions when applying it in research. On a general level, Schroeder [5] enumerates the types of data that can be gathered using hardware eye trackers. What about a software-based system like DFKEye? According to Schroeder, hardware-based eye-trackers:

> *Can tell whether users are even looking at the screen*. Since DFKEye is not actually tracking gazes, but focuses on the user's behavior, it cannot tell whether a user is just passively sitting in front of the screen, or focusing any areas on it. However, as soon as a user wants to use information on the screen, her mouse movements will indicate her areas of interest. In this sense, although DFKEye cannot tell whether the user is *not* looking at the

screen, we can certainly tell *when* she is and *which* information she is requesting.

Tell whether users are reading or scanning. Even though we cannot distinguish between reading and scanning on the grain size of words, we can gather evidence for reading vs. scanning at the region level by inspecting the focus times of the different regions. Short, frequently varying foci might indicate whether a user is scanning for some item, whereas long, stable foci suggest that the user is reading. Since the regions on the screen are only blurred, but not blackened, the user is still able to see the overall layout of screen and has thus enough visual cues for finding relevant areas on the screen.

Compare user scan patterns. This analysis requires the comparison of single user data, which both the hardware- and the software-based eye trackers yield.

Learn the relative intensity of a user's attention to various parts of an interface. Information about the distribution of a user's attention to different screen areas, directly arises from the division of the screen into distinct regions. For each region DFKEye registers when and how long the user looked at it.

The comparison above indicates that our behavior-based approach provides a functionality that allows to address research questions that previously required expensive hardware eye-trackers. It is not our intention to propose DFKEye as a substitute for hardware-based system, however, we found several questions that are not answered by the use of hardware eye trackers. DFKEye can, for example:

Let you know whether users actually see something on the screen. A hardware eye tracker cannot distinguish between users studying the region they are looking at and users who simply stare at them, not perceiving the respective content. However, since DFKEye blurrs the content of a region after a certain period of time, a. participant who wants to continue studying the region, needs to move the mouse in order to make the region's content visible again.

Prove that users didn't see something. Peripheral vision cannot be tracked by a hardware eye tracker. In a setting that uses DFKEye, however, regions that are not focused cannot be perceived by a user since they are blurred and their content therefore not accessible.

Test everybody. Hardware eye trackers are difficult to calibrate, to keep calibrated, and are sometimes not usable with some participants at all, whereas DFKEye requires users only being able to move a mouse. In our opinion, this amounts to be the major advantage of DFKEye: Even when testing completely cooperative participants, the calibration procedure often takes almost as long as the experiment the eye tracker is actually used for. Depending on the type of hardware eye tracker (head-mounted vs. remote), participants need to be instructed not to move their heads too much. This requirement implies a rather uncomfortable situation for the participants and restricts the duration of an experiment that makes use of eye tracking. In contrast, DFKEye does neither need a calibration procedure, nor specific instruction regarding head movement, and can thus even be applied remotely via the Web. With only a few exceptions, most eye tracking studies tested only a very small number of participants, or relied on single case studies — due to its effortless applicability DFKEye promises to conduct studies with a broad number of participants.

DFKeye can be easily adapted to a broad range of experimental requirements: Three different variants of hiding the elements within a region have been implemented:

cover (covers the respective completely),

fade (changes the font colour to approximately match the background colour),

and *zoom* (changes the font size to miniscule).

DFKEye allows the adjustment of the delay between entering a region with the mouse pointer and unhiding it, as well as setting the time after which the region is hidden again when the mouse pointer is not moved.

DFKeye is realized as a Javascript application that uses onMouseOver/on-MouseOut event handlers on HTML DIV elements to catch the mouse movements. Time information is send to a servlet (java server application) immediately after a region is blurred. All data is stored in an XML format and can thus be easily transformed to the required input format easily using XSL-stylesheets. DFKEye is implemented in strict conformance to the W3C standards, and is usable in all W3C-compliant browsers (Mozilla, Netscape6) and in IE5. Currently, we are preparing a DFKEye-kit that can be easily integrated into existing web-based systems.

3.1 Comparing Both Methodologies

As argued in the last paragraph, there are obviously several advantages associated with the use of DFKEye in comparison to a hardware-based tracker. However, the central question is whether DFKEye and a real eye tracker in fact yield similar data patterns when applied in a study. To evaluate this question, we conducted a study targeted to analyze the relative strength of visual cues, using a hardware eye tracker and DFKEye. Although data analysis is still underway, our initial results are encouraging: To allow for a direct comparison of the data gathered, we contrasted the data on the level of screen regions which were defined for the hardware-based tracker (SMI) and for DFKEye. The correlation coefficient of the data gathered with the two systems fall within the range between .73 and .94 and thus

indicate that the hardware-based eye tracker and DFKEye obviously yielded similar results in this study.

In our opinion, DFKEye offers an easy-to-use, inexpensive and reliable alternative to hardware eye tracking. Since DFKEye can easily be integrated into existing web-based applications such as eLearning frameworks [4], it allows the use of evidence for a user's regions of interest in domains where hardware-based eye trackers would not be applicable.

4. ACKNOWLEDGEMENTS

We thank Michael Dietrich for his work for the implementation of various versions of DFKEye and Thorsten Moritz for conducting the evaluation study.

5. REFERENCES

[1] Byrne, Michael D., Anderson, John R., Douglass, Scott & Matessa, Michael: *Eye Tracking the Visual Search of Click-Down Menus*. Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit, p. 402-409, 1999.

[2] Chen, M. C., Anderson, J. R., and Sohn, M. H.: *What can a mouse cursor tell us more? Correlation of eye/mouse movements on web browsing.* Proceedings of Computer Human Interaction (CHI) 2001, p. 280-281, 2001.

[3] Egner, Itti and Scheier. Comparing attention models with different types of behavior data, *Investigative Ophthalmology and Visual Science* (Proc. ARVO 2000), Vol. 41, No. 4, p. S39

[4] Melis, E., Büdenbender, J.,Andres, E.,Frischauf, A., Goguadze, G., Libbrecht, P., Pollet, M., and Ullrich, C. ACTIVEMATH: A generic and adaptive web-based learning environment. *Artificial Intelligence and Education,* 12(4), 2001.

[5] Schroeder, Will: *What is eye-tracking good for?* http://www.uie.com/eyetrack2.htm

[6] Wilhelm, T., Yom, W., and Berger, S. Site-Covering eine innovative Methode zur Erfassung der Informationsaufnahme und des Entscheidungsverhaltens auf Webseiten, *Planung & Analyse*, April 2002