# GazeEverywhere: Enabling Gaze-only User Interaction on an Unmodified Desktop PC in Everyday Scenarios

**Simon Schenk, Marc Dreiser, Gerhard Rigoll, Michael Dorr**
Technical University Munich
Munich, Germany
{simon.schenk, marc.dreiser, rigoll, michael.dorr}@tum.de

## ABSTRACT

Eye tracking is becoming more and more affordable, and thus gaze has the potential to become a viable input modality for human-computer interaction. We present the GazeEverywhere solution that can replace the mouse with gaze control by adding a transparent layer on top of the system GUI. It comprises three parts: i) the SPOCK interaction method that is based on smooth pursuit eye movements and does not suffer from the Midas touch problem; ii) an online recalibration algorithm that continuously improves gaze-tracking accuracy using the SPOCK target projections as reference points; and iii) an optional hardware setup utilizing head-up display technology to project superimposed dynamic stimuli onto the PC screen where a software modification of the system is not feasible. In validation experiments, we show that GazeEverywhere's throughput according to ISO 9241-9 was improved over dwell time based interaction methods and nearly reached trackpad level. Online recalibration reduced interaction target ('button') size by about 25%. Finally, a case study showed that users were able to browse the internet and successfully run Wikirace using gaze only, without any plug-ins or other modifications.

### Author Keywords

Eye tracking; Gaze-based interaction; Mouse replacement; Smooth pursuit

## INTRODUCTION

Eye tracking has become increasingly popular in the field of human computer interaction, especially in light of the recent introduction of affordable, yet relatively precise eye trackers. Besides the analysis of the user's gaze behaviour for usability measurements, using gaze as input modality has been a field of research for many years. Although gaze has proven to be a fast and reliable input modality for typing [29], and pointing at target positions with gaze is faster than using a mouse [6], using only gaze as an input modality is still not practical. This is because gaze-only interaction poses two major challenges: the Midas touch problem [12], and the "region of uncertainty" problem [15].

The Midas touch problem is named after the mythical king who turned everything he touched into gold, and who eventually starved to death because this also applied to his food. In gaze-based interaction, it describes the problem that the gaze patterns required to decide where to interact may already be misclassified as interaction gestures.

The "region of uncertainty" describes the roughly circular area around the measured gaze position in which the actual gaze position can lie. Its size highly depends on three factors: i) The precision of the used eye tracker; ii) the quality of the tracker calibration; and iii) the angle of foveal vision, i.e. the precision of the human fixation itself. Obviously, only the first two factors can be improved using better hard- and software.

Every system providing gaze-based input has to cope with both the Midas touch and the "region of uncertainty", and most systems do so by developing a unique graphical user interface (GUI) for the software in use. This, however, limits the use of gaze as an input modality to those applications, and hence, a different approach for gaze-only interaction is a modification of the GUI at a system level.

We propose GazeEverywhere, a solution that enables gaze-only mouse replacement on any computer without any system level modification. This is done by following a layered interface approach and adding a transparent layer on top of the computer's GUI. This layer is used to show all visual stimuli needed for the gaze interaction and interacts with the operating system only by sending mouse events. It can either be implemented in software by a transparent, always-on-top application, or using an additional hardware, which is also described in this paper.

In particular, our contribution consists of three parts:

1. A combination of dwell time and the state-of-the-art smooth pursuit-based activation method SPOCK [25] that has been shown to be robust against Midas touches, even in conditions with high mental workload (Figure 1, right).

2. An online offset compensation algorithm that uses the smooth pursuit gestures during interaction to detect any tracker offset on-the-fly and that compensates this offset for a better local tracking precision (Figure 6).


*CHI 2017, May 6-11, 2017, Denver, CO, USA.*

http://dx.doi.org/10.1145/3025453.3025455

3. An optional hardware setup that is able to project superimposed visual information on top of a computer monitor using the same mechanism as in a head-up display and to send mouse events to the controlled computer via a standard USB HID mouse interface (Figure 2).

In a series of three experiments described in the following, we demonstrate that i) SPOCK outperforms a traditional, dwell time-based activation method; ii) online offset compensation improves tracking accuracy; and iii) the interaction technique can be used on off-the-shelf software. All these experiments are done using the optional hardware to demonstrate that gaze interaction is possible even without any software modification.

## RELATED WORK

Since gaze-based interaction has been a research field for many years, different solutions exist of how gaze can successfully be applied as an input modality. In particular, these solutions try either to solve the Midas touch problem, or the "region of uncertainty" problem, or both. This is mostly done in separate applications; however, some effort has been put into enabling complete systems for gaze-based interaction.

### Reducing the Midas Touch Problem

When using user interfaces with static buttons, the most common gaze-based activation method is dwell time: To activate a button, the user has to look at it for a specific duration. This method is easy to learn and ergonomic [12], but it is also very prone to the Midas touch problem. As a consequence, many studies have tried to determine the optimal dwell time. For example, Majaranta et al. proposed an adjustable dwell time for their gaze typing task [17]. They found out that the required dwell time shortens over time and that the optimal dwell time highly depends on the user. After ten repetitions the optimal dwell time ranged between 180 ms and 380 ms. Other researchers used dwell times between 150 ms and 1000 ms [10, 12, 16]. Penkar et al. pointed out that dwell time can be shortened if active areas are distinct from informative areas, e.g. the content of the button is presented below the dwell area [22].

Since no final conclusion can be drawn for an optimal dwell time, a more promising approach is to prevent Midas touches by using a different gesture for target activation [11, 14, 21, 25].

Huckauf and Urbina used a saccade for activation [11]: When glancing at a target, a visual stimulus appears on one side of the target. To activate the target, an anti-saccade, i.e. a saccade towards the other side of the target, has to be performed. Although this prevents Midas touches in theory, anti-saccades are difficult to perform in practice because of attentional capture and thus the user experience may become unpleasant. Lutteroth et al. [14] followed the design guidelines of Penkar et al. and developed a user interface where additional dwell targets are used for activation: When a user gazes close to clickable targets, these targets are highlighted with different colours. A palette of the used colours is presented next to the monitor and by dwelling at one of these colours, the corresponding target is activated. Since the targets do not have to be fixated precisely, this approach also successfully deals with the "region of uncertainty" problem. Schenk et al. utilized smooth pursuit gestures in their SPOCK activation method [25]: When a target is fixated, two moving disks appear above and below the target, and by following one of these disks the user's eye performs a smooth pursuit which leads to the target activation. The symmetric design of moving targets cancels out the attentional capture and hence Midas touches are rare. In contrast to anti-saccades, smooth pursuits are natural eye movements and thus easy to perform.

### Dealing with the "Region of Uncertainty"

The most common solution to circumvent the "region of uncertainty" problem is not to use a selection-activation approach, where a target first has to be selected before it can be activated, but to use a gesture-based approach where different gestures cause different actions directly. These gestures can be either based on saccades [3, 4, 8, 20], or on smooth pursuits [5, 9, 28]. In both cases, gaze episodes, i.e. the gaze position in a certain time frame, are analysed, and the gaze path in those episodes can be compared to predefined patterns. Since only relative eye movements are taken into account, the absolute gaze position is less important and the tracker does not need to be calibrated properly.

Drewes and Schmidt [8] and Møllenbach et al. [20] presented saccade-based interaction methods that did not need any visual stimuli. Where Møllenbach et al. only used single gaze strokes, Drewes and Schmidt connected multiple strokes into one gesture. Bee et al. presented a gaze-based keyboard that was based on multi-stroke gestures [3] and Møllenbach et al. modified this solution and presented it together with other multi-stroke gestures [19]. A different solution for a stroke-based keyboard has been published recently by Best et al. [4].

For smooth pursuit-based interaction, Vidal et al. introduced visual targets that are moving with a constant speed on circular trajectories [28]. When following one of these targets, the user's gaze trajectory is compared to the target trajectories using Pearson's product-moment correlation coefficient [28] and the corresponding target is activated if a matching pair is found. While an initial implementation of this technique enabled gaze-based interaction on a public display [28], Esteves et al. recently used this technique successfully on a smartwatch, demonstrating that even very small circular movements are measurable with an uncalibrated tracker [9].

Although all these gesture-based approaches may be used with low tracker accuracy and big tracker offsets, they require knowledge of the exact positions of the clickable items. Thus, applying them to arbitrary, button-based user interfaces without customizations and deep system integration is difficult.

Skovsgaard et al. investigated a different approach that is applicable to existing, button-based GUIs using a zooming option [27]: When dwelling on a target, the area around the dwell position is magnified and consequently even small targets become clickable. Nevertheless, this approach still requires some form of system integration, i.e. access to the framebuffer, in order to re-render the magnified area.

### Integrating Gaze Control in Existing GUIs

Besides trying to solve the two major challenges in gaze-based interaction on an application level, some effort has been made to incorporate gaze into existing systems and hence make these systems controllable via gaze. Salvucci and Anderson [24] developed a specialized WIMP-category (Windows, Icons, Menus, Pointer) GUI that enables partial gaze control. To cope with the "region of uncertainty", they used a probabilistic model to find the most likely click target around the user's gaze position. At the expense of truly hands-free operation, their system did not suffer from the Midas touch problem because the authors used a "gaze button", i.e. a key on the keyboard, to activate targets.

Lankford [13] integrated gaze input into the Windows operating system. As activation method, dwell time was used and different dwell times performed different actions: A short dwell time enabled dragging and dropping the selected item, a longer dwell time clicked the element, and an even longer dwell time performed a double click. The current state of the dwell time was visualized with different colours and markers at the dwell position. To make even small targets clickable, a "zooming methodology" was implemented that shows a magnification of the fixation's surrounding at the centre of the screen.

A similar approach is used in the commercially available system from Tobii Dynavox called Windows Control[1]. This system has two different operation modes: mouse emulation and direct interaction. In the mouse emulation mode, the mouse pointer is fixed to the gaze position and consequently moved with the eye. In order to click, a dwell time is used and the different click modes can be selected via a button bar. In the direct interaction mode, a two stage approach is used for interaction: first, the action to be performed (click, double click, drag-and-drop) is selected from a button bar. Then, an additional dwell time determines where the action is to be applied.

A more recent solution was presented by Ahn et al. [1] in which they augmented a smartphone with gaze-based interaction. Because a smartphone GUI is designed for finger control, the buttons are rather large and the "region of uncertainty" is less of an issue. In this system, a panel displays a set of possible touch gestures, and gaze is used to select one of these gestures to be triggered. The usability of this approach has been tested successfully in an experiment with physically impaired participants.

All these systems have in common that they are deeply integrated into the operating system. Porting them to a different system therefore requires additional effort or may not even be possible.

## THE GAZEEVERYWHERE SOLUTION

All the solutions presented above that can successfully apply gaze-based interaction on traditional button-based GUIs needed visual feedback such as additional targets to solve the Midas touch problem [18]. This is either done at the application level, i.e. to enable gaze control for a specific application, or at the system level, i.e. to enable gaze control for all applications by integrating it deeply into the system.

Our solution aims to enable a system-wide gaze interaction, but without any deep system integration. We therefore use a layered interface approach, where a transparent GUI layer is placed on top of the system's GUI. This layer is responsible for drawing the visual stimuli and communicates with the operating system only by sending mouse events. A software implementation of this approach could be an application with a transparent always-on-top window. As a consequence, the application does not occlude any other application windows, whereas the visuals drawn on this layer are always visible. If a software solution is not possible (e.g. because the used operating system does not support any eye-tracking devices, or no applications can be installed on the system), an external hardware can be used to implement the transparent layer.

### Software

The main contribution of the GazeEverywhere system is the software component, i.e. the selecting and activating UI elements with gaze only.

#### *Gaze Gestures*

Since the chosen gestures have to be applicable to any button-based GUI, gaze gestures as presented in [20] and [28] cannot be used. Instead we chose a mouse emulation where the mouse interaction is performed in two stages: pointing and clicking. To prevent Midas touches, the gestures used for pointing and for clicking are different. For pointing (target selection) we chose a dwell time, since it can be performed easily and quickly. So far, three possible solutions for the clicking method (target activation) have been put forward: anti-saccades [11], fixations [1], or smooth pursuits [25]. Since anti-saccades are rather unergonomic, we only investigate the latter gestures. Hence, we implemented the two interaction gestures SPOCK and multi-stage dwell time (MSDT) as presented in Figure 1; we evaluate their relative performance in Experiment 1. Both gestures use a dwell time for selection, i.e. dwelling at a position moves the mouse pointer on the desktop PC to that position. At the same time, two targets appear above and below the dwell position. The MSDT targets are static and the SPOCK targets are moving up- and downwards with a constant speed of 5 deg/s. Following (SPOCK) or dwelling at (MSDT) one of these targets triggers a mouse click event on the desktop PC.

#### *Gaze Classification*

Despite great advances in eye-tracking quality, low-cost eye trackers still provide noisy measurements only. We therefore smoothed the incoming gaze data using an exponential decay filter with an alpha value of 0.2:

$$\text{pos}_t = 0.8 \cdot \text{pos}_{t-1} + 0.2 \cdot \text{gaze\_sample}_t. \quad (1)$$

For interaction, the performed gaze gestures have to be classified, i.e. the three possible gestures saccade, fixation, and smooth pursuit have to be detected. To do so, we used a sliding window with a window size of 15 samples ($\hat{=}$ 240 ms) and computed the sample-to-sample speed within this window. In a first step, the speed for each of the 15 samples was computed,

[1] http://www.tobiidynavox.com/windows-control/

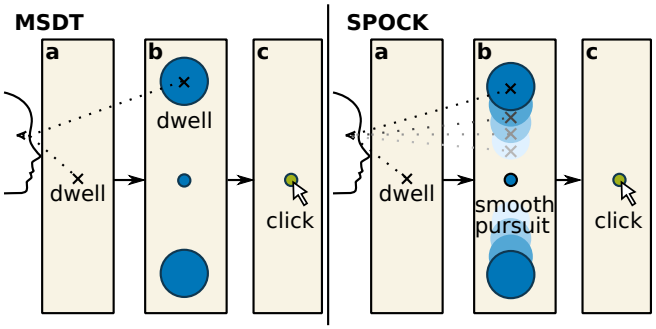


**Figure 1. Left (MSDT): after dwelling at a position (a), two static stimuli appear above and below the fixated point; dwelling at one of these (b) performs a mouse click (c). Right (SPOCK): after dwelling at a position (a), two moving stimuli appear above and below the fixated point; following one of these performing a smooth pursuit (b) performs the mouse click (c).**

and if the maximum velocity of any sample pair was above 80 deg/s the window was discarded as containing a saccade. If no saccades were detected, the mean speed of the entire window was computed and three different cases were distinguished: If the mean speed was below 4 deg/s, the gesture was classified as fixation; otherwise, if it was above 16 deg/s, the gesture was classified as "fast movement", i.e. a small saccade or a fast smooth pursuit. If the mean speed was between 4 deg/s and 16 deg/s, and all the samples were moving in the same vertical direction (the direction of the smooth pursuit stimuli), the gesture was classified as smooth pursuit.

### Hardware Setup

The hardware setup is optional and only needed if a software-only implementation is not applicable. In this case, an additional hardware is used to superimpose all additional visuals on top of the target monitor, and to emulate a USB mouse that is connected to the target PC (Figure 2). Currently, low-cost eye-tracking devices still lack wide-spread operating system support, so that an additional PC may be needed to provide eye tracking for non-Windows computers.

#### *Providing a Transparent Layer*

In order to create a transparent layer in hardware, we use a technique similar to the ones of head-up displays known from cars or fighter jets [23]: The content of a secondary image source is reflected directly into the user's line of sight using a transparent reflection surface. Our secondary image source was a LCD monitor that was mounted and carefully aligned horizontally between the user and the target monitor. Similar to the work in [2], a glass pane was mounted at a 45° angle and served as the reflection surface. A Tobii EyeX tracker was used as input device.

#### *Generating Mouse Events*

To generate mouse events on the target PC without any additional software, the USB HID device class is used: The gaze computer is connected to the target computer via an Arduino Leonardo board. The microcontroller on this Arduino board contains an USB host controller that appears on the target PC as a standard HID mouse/keyboard device. The gaze computer sends messages to the Arduino via Ethernet and the Arduino

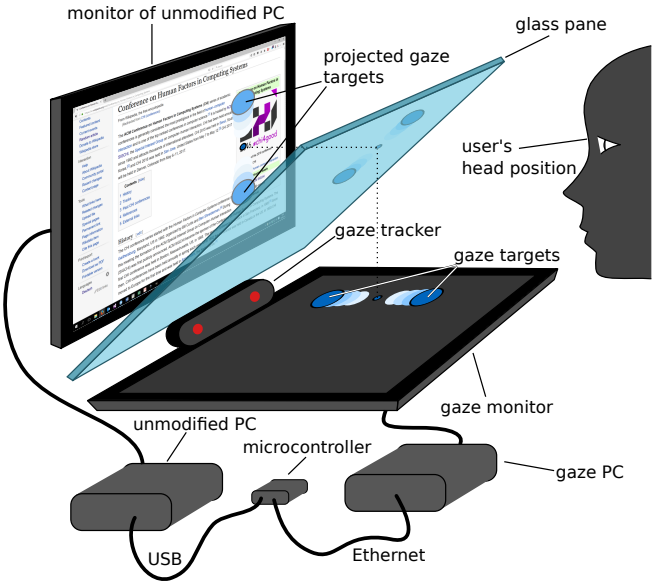


**Figure 2. The optional hardware setup superimposing visual targets for the SPOCK activation method onto a computer monitor: targets displayed on the bottom monitor are reflected into the user's line of sight and superimposed on the PC monitor by the glass pane. Mouse events are generated using a microcontroller and sent via USB.**

translates these into USB messages that are treated as mouse and keyboard events by all HID-supporting operating systems.

## EXPERIMENT 1: ACTIVATION METHOD

We conducted a user study following the ISO 9241-9 standard to address two questions: which of the two presented activation methods performs better, and how well these methods perform compared to existing input devices.

In the standard, pointing devices are tested by activating circular targets. These targets are placed in a circular fashion and have to be activated by successively clicking on them. The order in which they have to be activated is predefined such that the distance between two clicks is maximized. Changing the target size and the target distance changes the difficulty (ID) of the scenario. If a click has been performed outside a target, the corresponding target is counted as an error trial and skipped. The target sizes should be chosen large enough to maintain an error rate of less than 4%. For each trial the mean activation time, i.e. the time between two activations, is measured and a throughput (in bits/s) can be computed. Gaze has been tested before using this standard and showed valid results [30, 31].

In this first experiment the main focus lies only on the activation performance rather than on the pointing performance. Hence, and to assure a fail rate below 4%, we kept the target size constant and varied the target distance to generate a range of difficulties (IDs).

Besides the objective measurements of activation time and throughput we used the NASA TLX and the SUS questionnaire to receive subjective ratings. Furthermore, the gaze coordinates were recorded for a post-hoc analysis of the gaze behaviour.

### Participants

Twelve participants (9 male, 3 female) aged between 21 and 27 years (mean: 24.17, SD: 1.4) were recruited for this experiment. All were undergraduate or graduate students at the local lab and none of the students had any experience with eye tracking before. All participants had normal or corrected-to-normal vision and four participants wore optical correction during the experiments (2 glasses, 2 contact lenses).

### Experimental Setup and Design

The experiment took place in a quiet lab room with darkened windows and additional artificial light. We used the configuration with the optional hardware setup to make sure that our solution works even under these conditions. This heads-up-display setup produced a small parallax error of about 3 mm and sending the mouse events via USB introduced a delay of 18 ms. The two monitors in use were identical iiyama PROLITE 24" LCD monitors with the dimensions of 521 mm × 293 mm (W × H) and a resolution of 1920 × 1080 pixels; users were seated approximately 700 mm in front of the target PC monitor. The transparent reflection surface was a plastic glass pane with a thickness of about 4 mm. A Tobii EyeX tracker with a sample rate of 60 Hz was used to record the subject's gaze. The software running on the gaze computer was written in C# using the WPF framework[2] and ran with a constant 60 frames per second. The software running on the target PC was the *FittsStudy* tool[3] provided by Wobbrock et al. [30].

The pointing gesture for both activation methods was a 300 ms dwell time. The targets of the second dwell time in the MSDT scenario appeared 120 pixels (3.4°) above and below the original dwell position and had a diameter of 80 pixels (2.3°). The targets for the SPOCK scenario had a diameter of 30 pixels (0.9°) and were moving with a constant speed of 5 deg/s up to a maximum eccentricity of 200 pixels (5.7°), after which they were reset to the original dwell position and started moving outwards again. The dwell time for the second dwell in the MSDT scenario was 300 ms, and the minimum pursuit time in the SPOCK scenario was 250 ms.

The study itself was designed in a two-factorial within-subject design with the main factors being the *Activation method* and the *Target distance*. The target size was constant with 100 pixels (2.9°) and three different target distances were used: 900 pixels (25.7°), 560 pixels (16.0°), and 350 pixels (10.0°), resulting in the IDs of 2.20, 2.75, and 3.35, respectively, which lies within the typical range of IDs in the literature [30].

### Procedure

Upon arriving, participants had to fill out an informed consent form. Then the system was introduced and the two different interaction methods were explained to them.

The experiment consisted of two blocks, one for testing the MSDT method, and one for testing SPOCK. In each block, the tracker was first calibrated using a nine-point calibration and then the different target sizes were presented in a random order. Each target distance was tested with 13 targets, where the first two targets were for training and discarded in the evaluation. By repeating each distance, each participant had to select 26 targets per distance with each activation method. At the end of each block, the participants had to fill out the NASA TLX and the SUS questionnaire.

Half of the participants started with SPOCK as activation method followed by MSDT and the other half vice versa, to counterbalance any potential learning effects. The experiment took about 30 minutes in total.

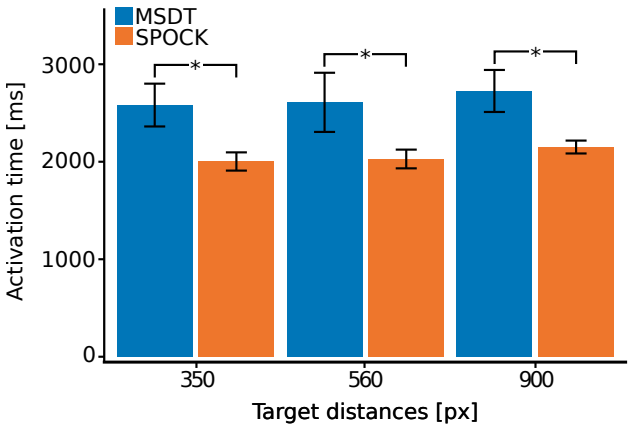


**Figure 3. Activation time for the three different target distances and the two different activation methods: activation time was significantly faster for SPOCK.**

### Results

Three participants had to be discarded from the evaluation due to tracking difficulties such as excessive offsets that led to high error rates above 40%.

Activation time was roughly similar between the different target distances, but clearly differed with activation method (Figure 3). A paired $t$-test showed statistical significance between activation methods for all three target distances ($p = 0.012$ for 350 pixels, $p = 0.030$ for 540 pixels, and $p = 0.01$ for 900 pixels). The mean activation time over all target distances was 2639 ms for MSDT and 2061 ms for SPOCK. This results in a throughput of 1.08 bits/s for MSDT and 1.23 bits/s for SPOCK.

Both SUS and NASA did not show any significant differences and the SUS score averaged at 76 points for SPOCK and 80 points for MSDT (SD: 14 and 12, respectively).

### Discussion

This first experiment showed that the SPOCK activation method was about 580 ms faster than MSDT. To determine whether this time difference was only based on the time spent on the final activation and not on time spent for the initial selection, we analysed the recorded gaze traces. Figures 4 and 5 show one typical trace for each activation method. For MSDT (Figure 4) four individual parts are clearly distinguishable: Acquiring the selection target, dwelling at the selection target,

[2] https://msdn.microsoft.com/de-de/library/ms754130(v=vs.110).aspx

[3] https://depts.washington.edu/aimgroup/proj/fittsstudy/

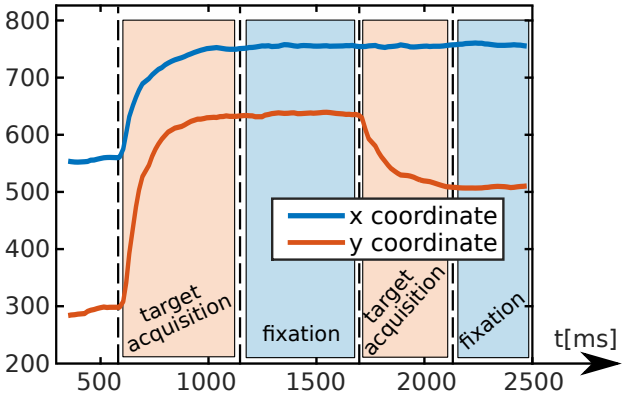


**Figure 4. Gaze plot for the MSDT activation method: The 1 s acquiring the second target for fixation slows down the activation.**

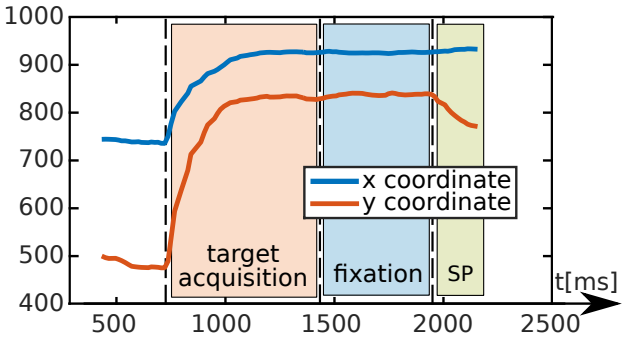


**Figure 5. Gaze plot for the SPOCK activation method: No second target acquisition needed. Detecting the smooth pursuit takes only 250 ms.**

acquiring the second dwell target, and dwelling at the second dwell target. In contrast, the SPOCK method (Figure 5) comprises only three parts: Acquiring the selection target, dwelling at the selection target, and following the smooth pursuit target. The first two parts of both methods represent the pointing gesture, with the remainder representing the activation. The traces show that in fact a similar amount is spent for the selection gesture for both methods; however, acquiring the second dwell target and dwelling at it for the MSDT activation method takes much longer than performing a smooth pursuit for the SPOCK method.

One could argue that a less aggressive smoothing algorithm could potentially lower the target acquisition time, however, Helmert et al. state that smoothing is a required step when using any dwell-based interaction technique [10].

Compared to other input devices, SPOCK (1.23 bits/s) comes close to the performance of a trackpad (1.7 bits/s), but is still outperformed by joystick (2.1 bits/s) and mouse (4.9 bits/s) [7, 30]. Following the results of this first experiment, SPOCK was used as activation method for the following studies.

## INCREASING LOCAL TRACKER ACCURACY

The first experiment showed that a Midas touch free activation method can be implemented with our hardware setup and that this method can achieve a competitive throughput similar to the one of a trackpad. However, to reach an acceptable error rate, a relatively large target size of 100 pixels (as determined in pilot testing) had to be used. In order to make the gaze-based interaction applicable to existing software, this minimum target size had to be further decreased, though, which means to reduce the "region of uncertainty".

However, none of the approaches presented in the related work section could be used, since they all rely on a deep integration in the used GUI. Instead, we aimed to reduce the "region of uncertainty" by increasing the local tracking accuracy. In general, the tracking error of an eye tracker is based on the tracker noise, and the tracker offset. The tracker offset depends on the quality of the used eye model and on the quality of calibration. Typical tracker calibration schemes use 1, 3, 5, or 9 calibration points to optimize the eye model on a global level, but local errors can still be high. However, SPOCK allows to continuously reduce these local errors using pursuit targets as implicit calibration points.

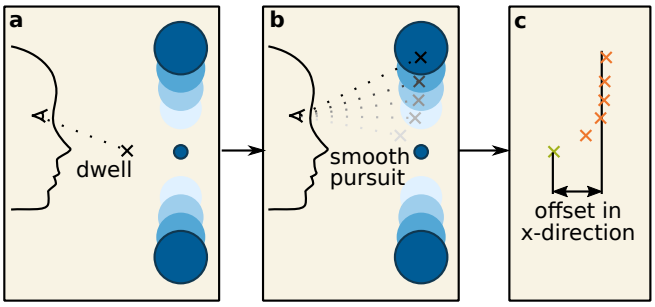


**Figure 6. The fixations and smooth pursuit during the interaction can be used to detect the local tracker offset: If an offset exists, the smooth pursuit targets do not appear at the true dwell position, but with this offset (a); when following one of the targets, this offset has to be overcome (b). By comparing the mean $x$ position of the fixation with the mean $x$ position of the SP, the offset can be computed.**

### Offset Detection

In order to compensate for local tracking offset, it has to be detected first. One way to do so is using the eye movements of the user during the interaction: if an offset exists, the measured fixation point is not the actual fixated point. Hence, the visual targets for activation do not appear directly at the (actual) fixation point, but with an offset. When following one of these targets, this offset is compensated by an eye movement, which can be detected.

Figure 6 shows how this detection works in particular: While dwelling at a position, the $x$ coordinate of the gaze position is stored. When the visual target appears and a smooth pursuit gesture (in $y$ direction) is detected, the average $x$ coordinate of the gaze positions during this smooth pursuit is measured and compared to the stored $x$ coordinate of the dwell. The difference between these two coordinates is the offset of the eye tracker in the $x$ direction.

Since only the offset in one direction can be detected at once, the moving direction of the smooth pursuit targets has to alternate between horizontal and vertical.

### Offset Compensation

To compensate for the tracker offset, the detected offset has to be stored and used afterwards. To do so, the active screen area is divided into a $5 \times 5$ grid (25 cells). Each cell holds the tracker's offset that has been measured for this cell before. All cells are initialized with an offset of zero, and the stored offset is updated afer each activation.

For each new gaze sample, the stored offset of the nine closest cells are taken into account for compensation. The weight of each offset is computed using the inverse distance weight algorithm [26]:

$$\text{pos}_{\text{comp}}(t) = \text{pos}_{\text{raw}}(t) + \sum_{i=1}^{9} d_i^3 \cdot \sum_{i=1}^{9} \left(\frac{1}{d_i^3} \cdot \text{offset}_i(t)\right), \quad (2)$$

where $d_i$ is the distance from the measured gaze coordinate to the midpoint of the $i$th cell.

## EXPERIMENT 2: RECALIBRATION

To test the performance of our offset compensation and its influence on the interaction accuracy, a second user study was conducted in which we used the same *FittsStudy* tool following the ISO 9241-9 standard as in the first experiment. In contrast to the first experiment, we did not want to test which interaction method has higher throughput, but which one produces better error rates. Therefore, we used a constant target distance of 350 pixels (10°) and five log-spaced target sizes: 100 pixels (2.9°), 64 pixels (1.8°), 41 pixels (1.2°), 26 pixels (0.7°), and 17 pixels (0.5°). The question to answer was if and by what amount the offset compensation decreases the error rate at the different target sizes. Moreover, we were interested for which target size an error rate of 50%, which was the subjective threshold for a still usable system, was achieved. Again, the gaze coordinates were logged as well for a detailed analysis.

### Participants

For this experiment, 8 participants (6 male, 2 female) were recruited with an age between 23 and 35 years (mean: 27.37, SD: 3.96). All participants gave written informed consent to take part in the study. They were either researchers or graduate students at the local lab and none had any experience with eye tracking, or gaze-based interaction. Two participants had normal, six corrected-to-normal vision (three wearing contact lenses, three glasses).

### Experimental Setup and Design

The hard- and software setup in this experiment was the same as in the first experiment. As activation methods, we used the same parameters as in the first experiment. For the offset compensating method, the moving direction of the smooth pursuit targets alternated between horizontal and vertical for each individual cell; i.e. if the previous offset had been measured in $x$ direction, targets moving in $x$ direction were presented in order to measure the offset in $y$ direction, and vice versa. Hence, the user did not have a direct influence on the target's moving direction. The movement speed and the maximal eccentricity was the same for both directions and the same as for the non-compensating method (5 deg/s; 5.7°).

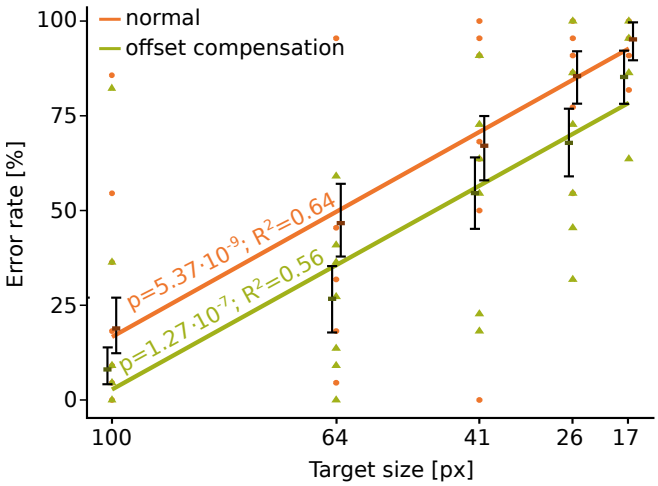


**Figure 7. Error rates with and without recalibration. The marked values show the mean error rate for each method and each target size; the whiskers show bootstrapped 99% confidence intervals.**

The experiment was designed in a two-factorial within-subject design, with *Target size* and *Input type* as main factors.

### Procedure

The experiment itself was divided into two blocks, each one testing one interaction method. At the beginning of each block, the tracker was calibrated with the manufacturer-provided software and validated with a custom routine. Since the calibration quality is crucial for the outcome of this experiment, only calibrations with a maximum validation error of 1.5° were accepted. The validation errors ranged from 0.7° to 1.4° for the non-compensating method, and from 0.5° to 1.5° for the compensating one. In each block, the different target sizes were tested sequentially starting with the largest target and this sequence was repeated once. Each target size was tested with 13 targets and the first two targets of each size were practice targets. This resulted in a total of $2 \times 11 = 22$ measured targets per target size and participant.

To counterbalance any learning effects, half of the participants started with the non-compensating method and the other half with the compensating one.

### Results

One participant had to be discarded from analysis due to a technical defect. For the remaining participants we counted the correct clicks and misclicks per target size (in total $7 \times 2 \times 11 = 154$ targets) and then computed the mean error rates as shown in Figure 7. The error rate for the non-compensating method was about 18% higher on average, and the remarkably similar regression slopes indicate that this difference was independent of target size. Error bars in Figure 7 denote the 99% confidence intervals of a 100,000-fold bootstrapping procedure and show that the error rates are significantly different for each interaction method and each target size.

Furthermore, we used the linear regression to check at which target size the error rate is 50%. For the non-compensating condition, this was the case at 64 pixels (1.8°) and for the

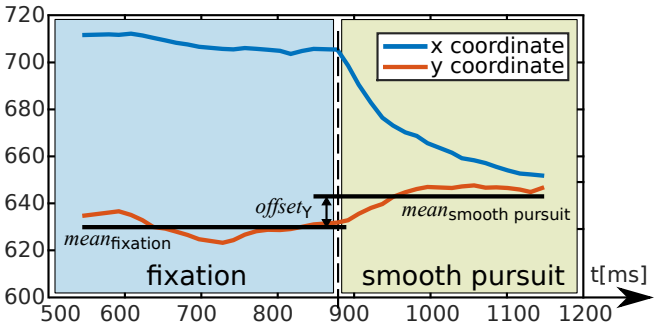


**Figure 8. A gaze trace of a smooth pursuit activation: the tracker offset can be computed by comparing the mean $y$ value during the fixation and during the smooth pursuit in $x$ direction.**

compensating condition, it was 48 pixels (1.4°), which results in a reduction of 25%.

### Discussion

The experiment shows that measuring the local tracker offset is possible and that compensating for it significantly improves the pointing accuracy. It also showed that with the offset calibration in place, targets with a size down to 1.4° become reliably clickable.

To further investigate how well the offset detection actually worked, we analysed the recorded gaze traces, one of which is shown in Figure 8. The smooth pursuit in the $x$ direction starting at about 900 ms is clearly distinguishable from the fixation before. The offset compensating movement in $y$ direction, however, is not easily detectable and by itself might be misclassified as tracker noise or a smooth pursuit. However, when computing the mean $y$ coordinate during fixation period and the mean $y$ coordinate during the smooth pursuit period, the offset in $y$ direction becomes measurable.

## EXPERIMENT 3: REAL-LIFE SCENARIO

Inspired by the results of the first two experiments, our goal was to test the usability of our interaction technique in a practical scenario. We therefore created an experiment where an unmodified version of a web browser has to be controlled by gaze only. In this experiment, the participants had to play Wikirace[4].

In Wikirace, each player starts at the same Wikipedia webpage and the goal is to reach a target Wikipedia page with as few clicks as possible. To navigate to the target page, only clicking on Wikipedia links in the current page is allowed. The benefit of this game is that it naturally limits the input modality to mouse only, without putting artificial restrictions on it and hence reflects the everyday scenario of browsing the web. Furthermore, the game character of the experiment forces the participants to read the presented webpage and to select the link they want to click carefully. Thus, any occurring Midas touches would be recognizable.

### Participants

Seven participants (6 male, 1 female), all either graduate students, or researchers at the local lab with ages between 23 and

[4] https://en.wikipedia.org/wiki/Wikipedia:Wikirace


**Figure 9. An image of the screen during the Wikirace. The two scroll buttons (red for up, green for down) are placed at the bottom of the screen not to cover any crucial content.**

29 years (mean: 26.3, SD: 2.1) took part in this case study. All had normal or corrected-to-normal vision; two already had minor experience with eye tracking, the rest had no experience. Two participants wore glasses during the experiment, and two wore contact lenses.

### Experimental Setup and Design

The hardware setup was the same as in the first two experiments and the game was played using an unmodified version of the Google Chrome web browser. To ensure that the links on the website had a clickable height of at least 47 pixels, the built-in scale factor of the browser was set to 125%. In order to provide full browsing capabilities without the need of a keyboard, some scrolling functionality had to be implemented. To do so, two dwell time buttons were used, which were placed over the task bar. These buttons were part of the transparent layer, and were projected onto the target monitor using the additional hardware setup. Dwelling on the left button triggered a "page down" button press and dwelling on the right triggered "page up" button press. The placement of these buttons was carefully chosen not to cover any crucial information on the target monitor.

Participants were asked to think aloud during the experiment, and a short interview was held afterwards. The questions asked in that interview were what they liked/disliked about the input method, whether they noticed the offset compensation, how easy the technique was to learn, and how demanding it was to use for a longer time. During the experiment, each participant's voice was recorded and analysed post-hoc.

### Procedure

After filling out an informed consent form, the tracker was calibrated using a nine-point calibration. Then the participants had to play three rounds of Wikirace, where the quickest path contained 6, 5, and 4 pages, respectively. The users were asked to think aloud while playing. There was no recalibration between the rounds and completing all three rounds took between 15 and 35 minutes.

### Results and Discussion

The setup did work well in this scenario, and all participants managed to reach the target webpage. The taken paths from

the starting page to the target page varied for all participants and had between 7 and 30 nodes.

Participants liked that our method was Midas touch free. P1, who had previous experience with gaze-based interaction, stated that is was "definitely better than dwell time", and that "there, they had probably many more misclicks". Also, participants enjoyed "being able to rest the eye without clicking" (P4). This shows that the symmetric design to prevent Midas touches does not only work in laboratory experiments, but also in practical scenarios. Although some participants rated the method as exhausting, for some it was "less exhausting than expected" (P4). Participants who did find it exhausting did not like looking at a bright monitor for so long and for them, resting the eye helped.

Most of the participants wished that the pointing would be more precise and they stated that they would prefer using the mouse. However, all agreed that this method was good enough for basic interaction and that if "the hand cannot be used, it is a good alternative" (P6). Four participants noticed that "offset compensation did offer more precision" (P2), and some developed a technique where they specifically clicked next to a link for the system to increase local tracking precision. This shows that for the majority of participants, the offset compensation did bring a noticeable benefit.

None of the participants had any trouble learning the method and they stated that "the method worked surprisingly well" (P3). However, they did not like the scroll buttons and had a hard time dwelling at them, because the low position at the edge of the tracking range often caused a loss of tracking, but "except the scroll buttons, everything worked very well" (P5).

In summary, the interaction method worked very well in this experiment and the overall feedback of the participants was very positive, except for the scroll button implementation, which has to be improved further.

## CONCLUSION

Gaze can be a intuitive and reliable input modality, especially for people with manual motor impairments or in scenarios where hands-free operation is mandatory. With cheap, yet relatively precise eye trackers entering the end-user market, this input technique appeals even more. Although gaze-based user interaction has been a field of research for many years, no solution exists so far that is able to apply gaze interaction to off-the-shelf software. This is partly because additional visual targets are required for advanced interaction methods, and thus the used software has to be modified in order to provide them.

We showed that it is possible to implement a system-wide gaze-only interaction technique which is usable in everyday scenarios and does not need a deep system integration. We achieved this by following a layered interface approach and putting a transparent GUI layer on top of the system's GUI. In a user study following the ISO 9241-9, we could successfully demonstrate that this approach enabled us to implement state-of-the art interaction techniques (MSDT and SPOCK) and that the SPOCK activation method can achieve a throughput close to that of a touchpad (1.23 bits/s and 1.7 bits/s, respectively).

We further introduced a novel method of how to detect the tracker offset using only the interaction gestures and how to compensate for it on-the-fly. A second user study, also following the ISO 9241-9 standard proved that this error compensation method reduces the click error rate by 18% and that the target size required for an error rate of at most 50% can be decreased by 25%.

Finally, we tested our solution in a real-life scenario, where participants had to navigate a web browser. Particularly, they had to play three rounds of Wikirace, where the goal is to navigate to a target Wikipedia page by only clicking on the hyperlinks. The system technically worked in this practical application and the subjective statements of the participants indicate that the solution is usable in such scenarios.

### Limitations and Future Work

There are still some limitations to the system: currently, we are only able to trigger left mouse clicks. Other mouse actions, such as right clicks or mouse wheel events, are not implemented yet. One potential way to implement these actions would be to add a button bar, similar to our scroll buttons, to the transparent layer that lets the user choose between the different mouse events. However, how to implement gaze-based drag-and-drop operations is still an open research question.

The hardware setup in its current state is expensive and bulky. For non-Windows systems, a powerful computer is still required to run the eye-tracking software, but hardware-only eye-tracking modules that do all signal processing on chip are currently under development and thus hardware costs might go down in the future. The size of the hardware setup could be reduced by transparent displays.

## ACKNOWLEDGMENTS

We would like to thank the reviewers for their helpful feedback. Our research was supported by the Elite Network Bavaria.

## REFERENCES

1. Hyunjin Ahn, Jaeseok Yoon, Gulji Chung, Kibum Kim, Jiyeon Ma, Hyunbin Choi, Donguk Jung, and Joongseek Lee. 2015. DOWELL: Dwell-time Based Smartphone Control Solution for People with Upper Limb Disabilities. In *Proc. of the SIGCHI Conf. Ext. Abst. on Human Factors in Computing Systems (CHI EA '15)*. ACM, New York, NY, USA, 887–892. DOI: http://dx.doi.org/10.1145/2702613.2732862
2. Patrick Bader, Niels Henze, Nora Broy, and Katrin Wolf. 2016. The Effect of Focus Cues on Separation of Information Layers. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 509–514. DOI: http://dx.doi.org/10.1145/2858036.2858312
3. Nikolaus Bee and Elisabeth André. 2008. Writing with Your Eye: A Dwell Time Free Writing System Adapted to the Nature of Human Eye Gaze. In *Proc. of the 4th IEEE tutorial and research workshop on Perception and Interactive Technologies for Speech-Based Systems: Perception in Multimodal Dialogue Systems (PIT '08)*. Springer, 111–122.

4. Darrell S. Best and Andrew T. Duchowski. 2016. A rotary dial for gaze-based PIN entry. In *Proc. of the 9th Biennial ACM Symp. on Eye Tracking Research & Applications (ETRA '16)*. ACM, 69–76. DOI: http://dx.doi.org/10.1145/2857491.2857527
5. Dietlind Helene Cymek, Antje Christine Venjakob, Stefan Ruff, Otto Hans-Martin Lutz, Simon Hofmann, and Matthias Roetting. 2014. Entering PIN codes by smooth pursuit eye movements. *Journal of Eye Movement Research* 7, 4 (2014), 1–11.
6. Michael Dorr, Martin Böhme, Thomas Martinetz, and Erhardt Barth. 2007. Gaze beats mouse: A case study. In *Proc. of the 3rd Conf. on Communication by Gaze Interaction (COGAIN '07)*. 16–19.
7. Sarah A. Douglas, Arthur E. Kirkpatrick, and I. Scott MacKenzie. 1999. Testing pointing device performance and user assessment with the ISO 9241, Part 9 standard. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI '99)*. ACM, New York, NY, USA, 215–222. DOI: http://dx.doi.org/10.1145/302979.303042
8. Heiko Drewes and Albrecht Schmidt. 2007. Interacting with the Computer Using Gaze Gestures. In *Proc. of the 11th IFIP Int. Conf. on Human-Computer Interaction*. INTERACT '07, Vol. 4663. Springer, Berlin, 475–488. DOI:http://dx.doi.org/10.1007/978-3-540-74800-7{_}43
9. Augusto Esteves, Eduardo Velloso, Andreas Bulling, and Hans Gellersen. 2015. Orbits: Gaze Interaction for Smart Watches using Smooth Pursuit Eye Movements. In *Proc. of the 28th Annual ACM Symp. on User Interface Software and Technology (UIST '15)*. ACM, New York, NY, USA, 457–466. DOI: http://dx.doi.org/10.1145/2807442.2807499
10. Jens R. Helmert, Sebastian Pannasch, and Boris M. Velichkovsky. 2008. Influences of dwell time and cursor control on the performance in gaze driven typing. *Journal of Eye Movement Research* 2, 4 (2008), 1–8.
11. Anke Huckauf and Mario H. Urbina. 2011. Object Selection in Gaze Controlled Systems: What You Don't Look at is What You Get. *ACM Transactions on Applied Perception* 8, 2 (2011), 13:1–13:14. DOI: http://dx.doi.org/10.1145/1870076.1870081
12. Robert J. K. Jacob. 1990. What you look at is what you get: eye movement-based interaction techniques. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI '90)*. ACM, New York, NY, USA, 11–18. DOI:http://dx.doi.org/10.1145/97243.97246
13. Chris Lankford. 2000. Effective Eye-gaze Input into Windows. In *Proc. of the 1st Biennial Symp. on Eye Tracking Research & Applications (ETRA '00)*. ACM, New York, NY, USA, 23–27. DOI: http://dx.doi.org/10.1145/355017.355021
14. Christof Lutteroth, Moiz Penkar, and Gerald Weber. 2015. Gaze vs. Mouse: A Fast and Accurate Gaze-Only Click Alternative. In *Proc. of the 28th Annual ACM Symp. on User Interface Software and Technology (UIST '15)*. ACM, New York, NY, USA, 385–394. DOI: http://dx.doi.org/10.1145/2807442.2807461
15. I. Scott MacKenzie. 2010. An eye on input. In *Proc. of the 6th Biennial ACM Symp. on Eye Tracking Research & Applications (ETRA '10)*. ACM, 11–12. DOI: http://dx.doi.org/10.1145/1743666.1743668
16. John Magee, Torsten Felzer, and I. Scott MacKenzie. 2015. Camera Mouse + ClickerAID: Dwell vs. Single-Muscle Click Actuation in Mouse-Replacement Interfaces. In *Proc. of the 16th Int. Conf. on Human-Computer Interaction (LNCS)*, Vol. 9170. 74–84.
17. Päivi Majaranta, Ulla-Kaija Ahola, and Oleg Špakov. 2009. Fast Gaze Typing with an Adjustable Dwell Time. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI '09)*. ACM, New York, NY, USA, 357–360. DOI: http://dx.doi.org/10.1145/1518701.1518758
18. Päivi Majaranta, Anne Aula, and Kari-Jouko Räihä. 2004. Effects of Feedback on Eye Typing with a Short Dwell Time. In *Proc. of the 3rd Biennial ACM Symp. on Eye Tracking Research & Applications (ETRA '04)*. ACM, 139–146. DOI:http://dx.doi.org/10.1145/968363.968390
19. Emilie Møllenbach, John Paulin Hansen, and Martin Lillholm. 2013. Eye Movements in Gaze Interaction. *Journal of Eye Movement Research* 6, 1 (2013), 1–15.
20. Emilie Møllenbach, John Paulin Hansen, Martin Lillholm, and Alastair G. Gale. 2009. Single Stroke Gaze Gestures. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI EA '09)*. ACM, New York, NY, USA, 4555–4560. DOI: http://dx.doi.org/10.1145/1520340.1520699
21. Diogo Pedrosa, Maria da Graça Pimentel, and Khai N. Truong. 2015. Filteryedping: A Dwell-Free Eye Typing Technique. In *Proc. of the SIGCHI Conf. Ext. Abst. on Human Factors in Computing Systems (CHI EA '15)*. ACM, New York, NY, USA, 303–306. DOI: http://dx.doi.org/10.1145/2702613.2725458
22. Abdul Moiz Penkar, Christof Lutteroth, and Gerald Weber. 2012. Designing for the Eye: Design Parameters for Dwell in Gaze Interaction. In *Proc. of the 24th Australian Computer-Human Interaction Conf. (OzCHI '12)*. ACM, New York, NY, USA, 479–488. DOI: http://dx.doi.org/10.1145/2414536.2414609
23. D. A. Ray. 1969. Head-Up Display. *The Aeronautical Journal* 73, 703 (1969), 622–624. DOI: http://dx.doi.org/10.1017/S0001924000052295
24. Dario D. Salvucci and John R. Anderson. 2000. Intelligent Gaze-added Interfaces. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI '00)*. ACM, New York, NY, USA, 273–280. DOI: http://dx.doi.org/10.1145/332040.332444

25. Simon Schenk, Philipp Tiefenbacher, Gerhard Rigoll, and Michael Dorr. 2016. SPOCK: A Smooth Pursuit Oculomotor Control Kit. In *Proc. of the SIGCHI Conf. Ext. Abst. on Human Factors in Computing Systems (CHI EA '16)*. ACM, New York, NY, USA, 2681–2687. DOI: http://dx.doi.org/10.1145/2851581.2892291

26. Donald Shepard. 1968. A Two-dimensional Interpolation Function for Irregularly-spaced Data. In *Proc. of the 1968 23rd ACM Nat. Conf. (ACM '68)*. ACM, New York, NY, USA, 517–524. DOI: http://dx.doi.org/10.1145/800186.810616

27. Henrik Skovsgaard, Julio C. Mateo, John M. Flach, and John Paulin Hansen. 2010. Small-target Selection with Gaze Alone. In *Proc. of the 6th Biennial ACM Symp. on Eye Tracking Research & Applications (ETRA '10)*. ACM, 145–148. DOI: http://dx.doi.org/10.1145/1743666.1743702

28. Mélodie Vidal, Andreas Bulling, and Hans Gellersen. 2013. Pursuits: spontaneous interaction with displays based on smooth pursuit eye movement and moving targets. In *Proc. of the 2013 ACM Joint Conf. on Pervasive and Ubiquitous Computing (UbiComp '13)*. ACM, New York, NY, USA, 439–448. DOI: http://dx.doi.org/10.1145/2493432.2493477

29. David J. Ward and David J. C. MacKay. 2002. Fast Hands-free Writing by Gaze Direction. *Nature* 418 (2002), 838. DOI:http://dx.doi.org/10.1038/418838a

30. Jacob O. Wobbrock, Kristen Shinohara, and Alex Jansen. 2011. The effects of task dimensionality, endpoint deviation, throughput calculation, and experiment design on pointing measures and models. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 1639–1648. DOI: http://dx.doi.org/10.1145/1978942.1979181

31. Xuan Zhang and I. Scott MacKenzie. 2007. Evaluating eye tracking with ISO 9241-part 9. In *Proc. of the 12th Int. Conf. of HCI Intelligent Multimodal Interaction Environments, Part III (HCII '07)*. 779–788.