# Look and Lean: Accurate Head-Assisted Eye Pointing

Oleg Špakov, Poika Isokoski, Päivi Majaranta
TAUCHI, School of Information Sciences,
University of Tampere
{firstname.lastname}@uta.fi

## Abstract

Compared to the mouse, eye pointing is inaccurate. As a consequence, small objects are difficult to point by gaze alone. We suggest using a combination of eye pointing and subtle head movements to achieve accurate hands-free pointing in a conventional desktop computing environment. For tracking the head movements, we exploited information of the eye position in the eye tracker's camera view. We conducted a series of three experiments to study the potential caveats and benefits of using head movements to adjust gaze cursor position. Results showed that head-assisted eye pointing significantly improves the pointing accuracy without a negative impact on the pointing time. In some cases participants were able to point almost 3 times closer to the target's center, compared to the eye pointing alone (7 vs. 19 pixels). We conclude that head assisted eye pointing is a comfortable and potentially very efficient alternative for other assisting methods in the eye pointing, such as zooming.

**CR Categories:** H.5.2 [Information Interfaces and Presentation]: User Interfaces – Evaluation/methodology; Input devices and strategies

**Keywords:** eye tracking; head movements; pointing; gaze input

## 1 Introduction

Pointing by gaze has both advantages and restrictions when compared to the traditional pointing methods using computer mouse and touchpad. One clear advantage of using gaze for pointing is the speed of eye movements: various studies have shown eye pointing to be about twice as fast as mouse pointing (e.g., [Ware and Mikaelian 1987]). However, due to limitations in current technology and challenges inherent in eye biology (size of fovea, micro saccades) [Hansen and Ji 2010; Monden et al. 2005], the accuracy and precision of eye pointing have not been competitive.

The main forms of inaccuracy in the gaze position signal from eye trackers are calibration error and noise. Noise can be dealt with by averaging filters. Finite-impulse response filters are quite common and easy to implement; they are also popular in the field of eye tracking. For example, Kumar et al. [2008] suggested using a filter where the resulting point is calculated as the weighted average of N latest gaze points. The stability of the cursor can also be improved by counteracting eye jitter using techniques such as force field, speed reduction, warping to target [Zhang et al. 2008]. Calibration error is harder to deal with since it includes drift, which is the portion of the error that changes dynamically over time. The reasons for drift include changes in lighting, eye geometry, and the relative position of the camera and the eye.

The insufficient accuracy of eye tracking does not allow direct substitution of mouse by gaze in interfaces developed for mouse-based interaction. Hence, a number of dedicated gaze-friendly applications and task specific tools have been developed with specialized adaptations for eye pointing [Skovsgaard et al. 2012]. Increasing the size of the interaction elements is a common solution to overcome the low accuracy of eye-tracking data (e.g., [Hansen et al. 2001]) but it raises other issues that hinder fluent interaction: less space is left for other informative elements to display, only a small number of interaction elements can be visible at the same time, the need for scrolling will increase, etc. Another popular method of adaptation is to use zooming [Lankford 2000] or other gaze-contingent techniques that dynamically increase the size of the interface elements under focus [Ashmore et al. 2005; Miniotas et al. 2004; Špakov and Miniotas 2005]. However, zooming requires time and can dramatically slow down the interaction [Skovsgaard et al. 2010].

Few attempts have been made to improve gaze-based interaction by introducing other modalities as means for correcting or fine-tuning the approximate cursor position given by gaze. MAGIC pointing by Zhai et al. [1999] exploited the eyes' ability to move fast to enhance manual mouse pointing. Gaze was used to first quickly place the cursor near the target, and the mouse was then used to select the target. There was no true mixture of input signals, though, as a signal from only one input modality affected the cursor at a time. MAGIC pointing is a noteworthy solution for using gaze as an additional input modality. More recent examples of the work in this area demonstrate the usefulness of gaze pointing combined with manual selection in both desktop environments [Fono and Vertegaal 2005; Kumar et al. 2007; Biedert et al. 2012; Rozado 2013] as well as pervasive and mobile setups [Stellmach and Dachselt 2012, Turner et al. 2013]. However, these are not hands-free, which is our present focus.

Miniotas et al. [2006] combined eye pointing with spoken commands for target selection. The interactive elements nearest to the gaze cursor were marked by a colored frame. Users could select the desired target by speaking out its color if gaze alone was not accurate enough to select. Although the interaction remained hands-free, making a selection took more than 2.5 seconds and resulted in rather high error rates (>15% errors using 30x30 pixel targets).

Because of the limited accuracy in eye-tracking data, head movements were proposed in several studies as a more accurate alternative input channel (see, for example, [Stellmach and Dachselt, 2013]). In particular, Bates and Istance [2003] compared head and eye pointing. They found that eye mouse showed superior pointing speed but head mouse was superior in terms of pointing accuracy. This leads directly to our design where the eyes are used for fast movements and small head movements are used to improve accuracy when selecting small targets. In fact, our proposed method of combined gaze and head pointing may benefit people who use head pointing as their main method of input. Eyes enable increased pointing speed while head pointing ensures accurate target selection. As a bonus, the strain on the neck should reduce as gaze overtakes part of the effort required for moving the cursor. Similarly, people with repetitive strain injury may benefit from this alternative to conventional mouse.

One of the benefits of our solution is that it does not require any additional hardware to track the head movements. Modern video-based eye tracking systems estimate a number of eye tracking parameters and report the gaze point as coordinates on a plane usually matched with the display surface. In order to compute the intersection of the gaze vector and the display the trackers need among other things an estimate for the position of the eye(s) being tracked. Some trackers make the eye position information available along with the gaze point data. For example, systems developed by the Tobii Technologies produce data for both eyes. Each sample consists of its validity code, distance from the camera to the eye, pupil size, gaze point on the screen, and eye position in the camera view. The last parameter gives some insight into the user's movements as *the eye position in the camera view* reflects changes in the user's head position.

To our knowledge, the only attempts to exploit this parameter are the recent studies [Mardanbegi et al. 2012; Špakov and Majaranta 2012] where the eye position in the camera view was used to implement simple head gestures that could enhance gaze interaction. For example, a simple nod could be used to select the target pointed by gaze. Our attempt is the first to exploit it for improved accuracy.

We hypothesize that using very small head movements may significantly improve the accuracy of eye pointing without introducing other modalities or channels of input. Below, we discuss how head movements could be employed for correcting the gaze cursor position, and introduce our implementation. We then describe the experiments that compare eye pointing with and without such correction, followed by the results and discussion on the observed effects. We finish with concluding remarks and suggestions for future work.

## 2 Gaze Cursor Location Correction by Head Movements

### 2.1 Head Movements

To be precise, changes in the eye position in the camera view ($EP_{CV}$) do not directly reflect head movements: they have six degrees of freedom and cannot be represented by a simple 2D point. The actual type of movement is not important for our aim. Any movement that makes it possible to adjust the gaze cursor in 2D, which is convenient for the users and does not worsen the quality of eye tracking, is suitable for our purposes.

Non-rotational head movements (i.e., the body is also involved in these movements) in horizontal and vertical directions are best suited for this purpose, because the angle of the head in relation to the camera does not change, only its location in the camera view. Rotational head movements of small amplitude in same direction will have approximately the same impact on the $EP_{CV}$, but users may find them easier to produce than non-rotational movements. Especially, vertical non-rotational head movements could be quite inconvenient as they require body flexion. On the other hand, rotational head movements may influence the quality of the eye-tracking data more than the non-rotational movements. Therefore, they should be used with care; for example, maintaining the same angular head orientation in space is an obligatory condition with some eye-tracking systems.

Leaning forward or backward may also be reflected on the $EP_{CV}$ as vertical movements, since the camera is located aside from the screen center. However, according to our experience, such leaning can noticeably degrade the quality of gaze data. Finally, head tilting (roll) is not a common movement and we thus do not consider it to be a suitable candidate as a convenient mean to correct the location of a gaze cursor.

Prior to the experiments, we ran a pilot test to explore the potential of using changes in $EP_{CV}$ to correct the gaze cursor location. The pilot test showed that eye movements measured by our remote eye tracker had very little impact on the $EP_{CV}$: saccades across the screen caused much smaller change in the $EP_{CV}$ than tiny head movements. In contrast to [Mardanbegi et al. 2012] where authors used a head-mounted eye tracker, we concluded that no algorithms to filter out eye movements from $EP_{CV}$ are needed. Also, we found that horizontal and vertical head movements of small amplitude did not notably influence the measured point of gaze. Head rotations had some influence to it but, more importantly, horizontal head rotations (yaw) often caused eye losses by the eye tracking system for users wearing eye glasses. Therefore, we recognized the left-right head (and body) movements and the up-down head rotations (pitch) as the best choices for correcting the gaze cursor's X and Y coordinates, correspondingly. Below, we explain in details the procedure for using the $EP_{CV}$ to adjust the gaze cursor position.

### 2.2 Gaze Cursor Location Correction

Correcting the gaze cursor location using $EP_{CV}$ requires setting up a "reference point" from which an offset will be measured. An obvious solution is to memorize the $EP_{CV}$ coordinates at the moment the pointing by gaze starts (i.e., when the first valid eye data sample arrives from the eye tracking system). This solution is sufficient in the controlled experiment, but otherwise users should have the option to change it at any time, as occasional shifts in their neutral location are highly expectable. A possible solution is to use a dedicated command (keyboard key, gaze gesture, virtual button, etc.) to fix a new reference point.

There was no way to estimate the quality of eye position data itself. However, we learned that the 60 Hz eye tracker used in this study reports highly precise $EP_{CV}$ points, as they remained quite stable if the head was still (very minor fluctuation, that, we believe, was caused by the tiny head movements that are inevitable if the head is not fixed physically). If this data is not very precise, an average over several points should be calculated.

The gaze cursor should be relatively stable to make the corrections by head movements feasible. If the eye tracking system has no internal smoothing filter, it must be implemented in the application with gaze input. There are a number of filters found in literature that were reported as sufficiently effective in smoothing gaze cursor. Based on the recent comparison of gaze data smoothing filters [Špakov 2012], we suggest that a slightly modified two-state FIR filter with the triangular kernel function described by Kumar et al. [2008] might suit this purpose. The modification consists of an increase in the saccade duration threshold to 50 ms, with minimum length of 50 pixels; i.e., three outlying points are allowed instead of one in the original work. In addition, the time window was set to 500 ms; thus, the number of gaze points involved in smoothing is about 30.

During eye pointing, an offset from the current $EP_{CV}$ point in relation to the reference was used to determine the amount of shift, applied to the gaze cursor. Several conversion functions can be used to transform this offset into the shift: linear, sigmoid, tangent, etc. The conversion used in this study was linear. Other functions might perform better under different conditions. For example, a tangent function accelerates the correction and may be more suitable if the calibration accuracy is very poor.

The exact values of the transformation function parameters depend on the units of $EP_{CV}$ points, distance from eyes to an eye tracker, and the desired sensitivity of the correction algorithm. The eye tracker used in this study reported normalized values of the $EP_{CV}$ points (between 0 and 1). We set the coefficient of the transform function equal to 500: this resulted in about 15 pixels shift of the gaze cursor for about 1 cm of linear head movement in the plane orthogonal to the vector from the camera to eyes, when users were sitting at about 60 cm distance from the screen.

The pseudo-code of the algorithm for controlling a cursor location by gaze and head movements ($GCA_{HM}$: Gaze Correction Algorithm by Head Movements) is shown in Algorithm 1. It consists of two parts: one for smoothing gaze points (function *Filter*), and another for correcting the current gaze point according to head movements (function *GetCursorLocation*).

The function *GetCursorLocation* receives two arguments, a gaze point and an eye point ($EP_{CV}$). It calls the function *Filter* passing the new gaze point and the set of previous gaze points, and receives from it the initial cursor location as a “smoothed” gaze point. This location is then corrected according to the procedure described above and then used as the output of the *GetCursor-Location* function. The algorithm has three parameters influencing the smoothing (time window, saccade threshold, and saccade minimum duration) and one parameter influencing the correction (linear transformation coefficient).

We conducted three experiments to evaluate the usefulness and efficiency of this gaze cursor correction method.

## 3 Experiment 1: Comparison of Pointing Accuracy With and Without the $GCA_{HM}$ Algorithm

In the first experiment, we compared eye pointing with and without head correction (gaze+head vs. gaze only).

### 3.1 Participants, Equipment and Procedure

Nine volunteers consisting of the staff of the local university, 4 males and 5 females between 29 and 61 years (M = 39) took part in testing the $GCA_{HM}$ algorithm. All had experience in using gaze for pointing tasks, two were wearing glasses. Tobii T60 eye-tracking device with a 17” display (resolution of 1280x1024 pixels) was used in the study. Experimental software with stimuli presentation and implemented $GCA_{HM}$ algorithm was devel-

**Algorithm 1** *Pseudo-code of the algorithm ($GC_{AHM}$) for estimating cursor location from gaze and eye ($EP_{CV}$) points.*

```
define const TimeWindow =500
define const Threshold = 50
define const SaccadeMinDuration = 50
define const LinearTransformCoef =500

define buffer as array of time-stamped points
define refEyePoint as point
define altBuffer as array of time-stamped points
define fixation as point

function GetCursorLocation (gazePoint, eyePoint)

  if refEyePoint is not assigned
    refEyePoint ← eyePoint

  if buffer is empty
    append gazePoint to buffer
    return gazePoint

  remove points older than TimeWindow from buffer
  define cursorLocation as point
  cursorLocation ← Filter (buffer, gazePoint)

  define offset as vector from refEyePoint to eyePoint
  adjust cursorLocation by (LinearTransformCoef * offset)
  return cursorLocation

function Filter (gazePoints, newGazePoint)

  if fixation is empty
    append newGazePoint to gazePoints
  else
    if distance from newGazePoint to fixation < Threshold
      clear altBuffer
      append newGazePoint to gazePoints
    else
      append newGazePoint to altBuffer
      if time interval of altBuffer > SaccadeMinDuration
        clear gazePoints
        gazePoints ← altBuffer
        clear altBuffer

  define x, y and w as integers
  for index from length of gazePoints to 1
    add (index * gazePoints[index]. x) to x
    add (index * gazePoints[index]. y) to y
    add index to w

  fixation ← { x / w, y / w }

  return fixation

for each sample received from an eye tracker
  define gazePoint, eyePoint and cursorPoint as point
  gazePoint ← { sample.gazeX, sample.gazeY }
  eyePoint ← { sample.eyeX, sample.eyeY }
  cursorPoint ← GetCursorLocation (gazePoint, eyePoint)
  move cursor to cursorPoint
```

oped in C# (MS Visual Studio with .NET 2.0) to run on MS Windows XP operating system.

The screen was divided into 20 equal cells (5x4), and a target was shown five times in each cell in random order. The target was a dark circle with 10 pixel diameter shown in the center of a light-blue square (45x45 pixels). Each target was visible for 2.5 seconds. There was a 1 second pause before the next target appeared. The gaze cursor was visible as a dark semi-transparent circle with a 10 pixel diameter. The task was to place the cursor on the target as accurately as possible, and then signal the task completion by pressing a key.

Prior to the test a supervisor explained the purpose of the study and calibrated the eye tracker for each participant. Then the participants shortly practiced using head movements for gaze cursor location correction. The gaze only condition was omitted from the practice phase since all participants were experienced in eye pointing.

During the experiment, each participant completed two blocks of trials: with and without the $GCA_{HM}$ algorithm. For the block with the corrections by head movements enabled (condition "ON") the supervisor instructed the participants to look at the central circle of the target and try to use head movements, if needed, to move the gaze cursor as close to it as possible (ideally, to overlap it). The trial was completed by pressing a space key on a conventional keyboard. The participants were asked to spend no more than 2.5 seconds completing each trial: we tried to model quick interaction and long pointing duration could cause extra frustration and tiredness to the participants. After the time elapsed, the target disappeared even if the participant had not hit the space bar.

For the block without the correction by head movements (condition "OFF") the supervisor instructed the participants to look at the central circle of the target and press a space key as soon as they noticed that the gaze cursor was over the target. Or as soon as they decided they could not point more accurately.

The order of the blocks was counterbalanced between the participants. The total number of trials completed by each participant was 100 per block. The experimental software logged the target location, trial result (key pressed or not), trial duration, and the gaze cursor offset from the target center at the moment of keypress. The experiment took 15-20 minutes in total.

## 3.2 Results and Discussion

The first 30 trials of each block were not included in the analysis to reduce the effects of learning and correction strategy development. For valid trials the selection time (ST) and the cursor distance from the target (D) were computed. A trial was recognized as valid if 1) a selection was recorded in 2.5 seconds after the target appearance, and 2) D was less than 70 pixels. The filter by D was introduced to remove a few outlier trials where the gaze point suddenly re-located during the key press.

The percentage of trials removed from the analysis because of these filters was 9.8% (9.5% for ST, 0.3% for D) in the "ON" condition and 2% (1.8% for ST, 0.2% for D) in the "OFF" condition. A paired-samples t-test showed that the difference in the percentage of invalid trials was statistically significant ($p = 0.049$). We did not find any differences in the percentage of invalid trials between participants with and without eye-glasses.

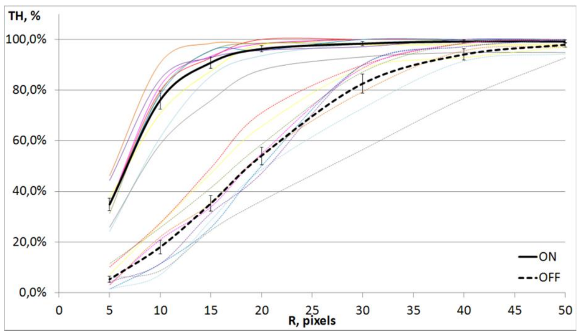

**Figure 1** *Experiment 1: The percentage of gaze cursor allocations within R pixels from a target center (thin curves – individuals, thick curves - the mean). The error bars indicate the standard error of the mean (also in the following figures)*

In the following analysis, we operate with the values averaged over each block. The average distance between the target and cursor centers was for was 8.0 (SD = 1.8) pixels when the correction algorithm was active. This means that the cursor usually partly overlapped with the target's central circle. However, when the head movement correction was not on, about half of the trials resulted in a cursor being outside the square target, ($D_{OFF}$ = 20.3 (SD = 3.6) pixels). The difference was statistically significant ($p < 0.001$). The selection times were 1541 (SD = 161) ms for "ON" and 1170 (SD = 284) ms for "OFF". This difference too was statistically significant ($p < 0.01$). The latter value was longer than expected, since based on literature (e.g. [Špakov and Miniotas 2005]) 400-700 ms should be a realistic range for the selection time when the correction is not used. The observed delay can be explained by the strategy selected by most participants. Instead of pressing the key immediately after the eye movement they were briefly monitoring the movement of the cursor with a hope that the tracker noise would move it closer to the target.

Using the actual distances of the cursor from the target when the spacebar was pressed we can visualize the effect of the $GCA_{HM}$ algorithm as shown in Figure 1. TH stands for target hit and R is the radius of a hypothetical target to hit. As $R$ decreases from 50 to 5 pixels in Figure 1, one can observe that with the algorithm on, the participants were able to move the cursor closer than 15 pixels to the central point of a target in 90.8% of the analyzed cases. Without the algorithm the percentage was only 35.3%. At 10 pixels with the algorithm the participants would have hit 76.1% of the targets. This would probably not be acceptable in real-world user interfaces. However, with these small targets one gets the best benefit from the algorithm. The hit rate without the algorithm was 18.0%. Allowing corrections by head movements made it possible to point by gaze at objects with a radius of 20 pixels as accurately as with the mouse (96.2%) (e.g., [Špakov and Miniotas 2005]). Without corrections only about half of the trials would have hit a target with a radius of 20 pixels (54.1%).

The t-test (percentages were transformed by arc-sinus function) revealed that the differences between algorithm on and algorithm off were statistically significant, when $5 \leq R \leq 40$ pixels ($p < 0.001$ for each $R$).

Figure 2 shows the dependency of the distance of the cursor from the target (D) on target location (the size of each circle is proportional to D but does not represent the absolute value). As expected, the distances recorded in trials with targets located on

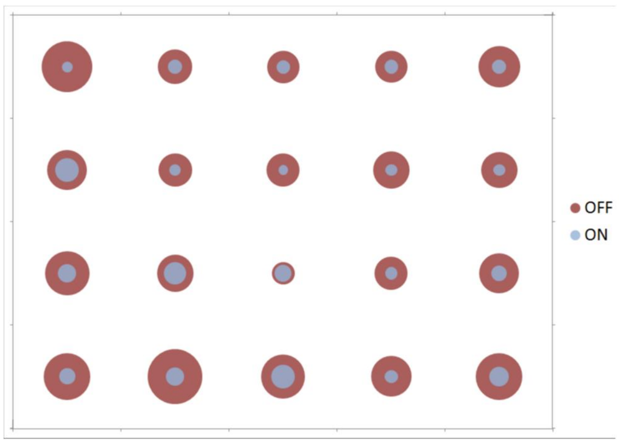

**Figure 2** *Experiment 1: Relative distances to target center grouped by target location.*

the periphery (the 14 outer cells; $D^P$) were greater than in trials with targets located in the central area (the 6 inner cells; $D^C$). However, the difference was not statistically significant in either condition: 17.8 vs. 22.2, $p = 0.069$ for the “OFF” condition, and 7.5 vs. 9.1, $p = 0.197$ for the “ON” condition.

Visual comparison of distances shown in Figure 2 leads to a conclusion that the proposed gaze cursor location correction method reveals its power mostly when the calibration is poor. It does, however, improve the pointing accuracy in all cases: the differences were statistically significant in almost all 20 screen cells. The percentage of gaze cursor allocations within *R* pixels from a target center, separately for central and peripheral screen areas, is shown in Figure 3.

After the test, the participants were briefly interviewed. All participants rated this correction method positively when asked whether it feels natural, easy to learn and use, and requiring only little effort.

## 4 Experiment 2: Revealing Head Movements Impact on Gaze Point Estimation

Data for one of the 10 participants of the first experiment was excluded from the analysis because he ignored instructions. He moved his head to adjust the gaze cursor location in the “OFF” condition, just like he did in the “ON” condition. The result was quite unexpected: there was almost no difference between the “ON” and “OFF” target hit percentages. This led us thinking that head movements alone without our algorithm can perhaps be used to greatly improve the pointing accuracy for the eye tracking system used in this study. A more worrisome interpretation was that it was the knowledge of the inability to make corrections with head movements that hurt the performance of all the other participants. Therefore, it was necessary to run another experiment in which trials with and without the algorithmic head movement support would be randomly mixed without informing the participants of the state of the system. They were instructed to always try to use head movements to improve the accuracy of gaze cursor.

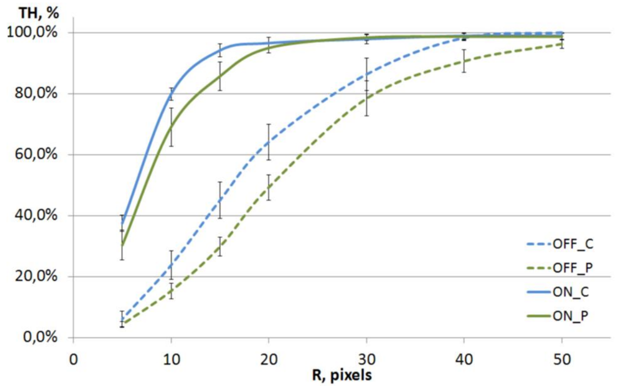

**Figure 3** *Experiment 1: The percentage of gaze cursor allocations within R pixels from a target center, separately for central (C) and peripheral (P) screen areas.*

### 4.1 Participants, Equipment and Procedure

All participants of the first experiment took part in the second experiment (5 males and 5 females between 29 and 61 years, M=42). The equipment, targets, and the way they were displayed remained the same. The only difference was the $GCA_{HM}$ algorithm activation: instead of two blocks of trials with the $GCA_{HM}$ algorithm turned on and off, a single block of 160 trials was generated in which 80 trials with the $GCA_{HM}$ algorithm were mixed randomly with 80 trials without it.

The participants were instructed to a) look directly at a target’s center, a) always try to use head movements to correct gaze cursor location, and c) to try to locate the gaze cursor as accurately as they could within 2.5 seconds and press the space bar before the target disappeared. They were also d) encouraged to spend more time correcting the location of the gaze cursor if it was not over the blue area, thus intentionally increasing the risk of not pressing the key within the 2.5 second time interval. The participants spent about 7-8 minutes completing the 160 tasks.

### 4.2 Results and Discussion

The first 10 trials were not included in the analysis (“warm-up” period). As in the first experiment, the same measured (selection time ST and distance D) and derivative (artificial target hit rate $TH^R$) variables were analyzed. However, unlike in the first experiment, we now included in the analysis the trials without the key press since we expected that the number of such trials for the condition “OFF” would be relatively high. Thus, the exclusion might distort the results due to rejection of the most inaccurate and time consuming trials. Indeed, the share of the key-not-pressed trials was 14.9% in this condition and only 5.4% in the “ON” condition.

The same threshold (70 pixels) was applied to filter out trials in which the high D was recorded due to eye losses or gaze cursor jitter. This filter caused 1.6% for “ON” and 2.2% for “OFF” trials to be removed from the analysis. The difference was not statistically significant.

We then operated with the values averaged over each condition. The participants were able to approach to the target to a distance of 11.3 (SD = 3.5) pixels in average when the correction was active, and 18.0 (SD = 6.2) pixels when the correction was not active ($p < 0.01$). In comparison to the first experiment, accuracy

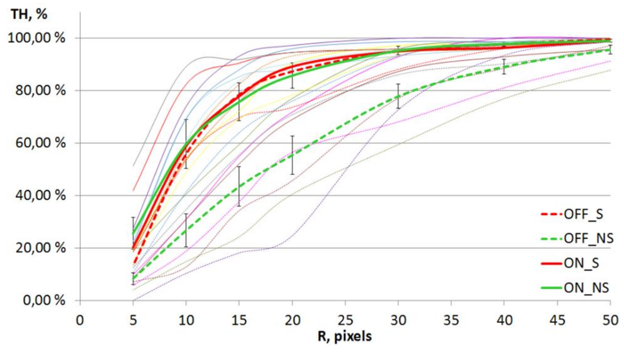

**Figure 4** *Experiment 2: The percentage of gaze cursor allocations within R pixels from a target center, separately for two groups of participants, with the ability to control the gaze cursor by head movements (S) and without such ability (NS).*

in "ON" condition increased and accuracy in "OFF" condition decreased. Interestingly, there were 3 participants (including the one whose data were removed from the analysis in the first study) for whom the gaze cursor distances from the target center were almost same in both conditions, 11.5 pixels for "ON" and 11.8 pixels for "OFF". For others (index NS) the difference between these values remained about the same as in the first experiment (11.3 pixels for "ON" and 20.6 pixels for "OFF"). We hypothesized that some reduction in pointing accuracy in the "ON" condition may have happened because of the frustration due to the varying effectiveness of the head movements. The participants may have used the head movements conservatively, which may have led to less effective corrections.

The selection times were 1463 (SD = 224) ms for "ON" and 1631 (SD = 263) ms for OFF. This difference was statistically significant ($p < 0.01$). The value for "OFF" increased notably in comparison to the first experiment. However, for the three participants who had the ability to correct the cursor position without our algorithm the selection times were about the same, while for the rest the difference was about 250 ms.

Figure 4 shows the dependency of target hit rate on *R* between 5 and 50 pixels. This rate drops quicker as *R* decreases than in the first experiment. This may indicate the importance of consistency in system response for interaction, as we hypothesized earlier.

The method that the tree participants used for the gaze point correction without our algorithm remains a mystery that should be investigated in further work. It may be a combination of eye and head movements that interact in clever ways with weaknesses in the trackers head position compensation. The participants themselves were not fully aware of what was going on either.

## 5 Experiment 3: Do Users Naturally Move Head When Pointing Accuracy is Insufficient?

After the head movements proved to be helpful in improving the pointing accuracy (either with or without the supporting algorithm), we hypothesized that novice users could perhaps notice and use it too when interacting with gaze-controlled interface even if they are not informed about the ability. This speculation was based on our long-term observations of participants during years of conducting tests on gaze-controlled interfaces: some users instinctively move their head and body to "reach" the target when the eye-controlled cursor does not hit it.

In experiment 3 we wanted to verify whether the head and body movements needed for compensating inaccuracies of eye pointing indeed appear naturally without instruction. Experiment 3, was almost identical to experiment 1. We were interested in the comparison of selection time (ST), cursor distance to target (D) and target hit rate ($TH^R$) with and without the $GCA_{HM}$ algorithm activated, and in whether the participants will notice that the head movements help in improving the pointing accuracy.

### 5.1 Participants, Equipment and Procedure

Nine students from the local university participated in the experiment (6 males and 3 females between 24 and 27 years). None had participated in experiments where gaze served as an input, but a few (3) had been involved in studies where their gaze was passively recorded.

The equipment, targets, and the way the stimulus was displayed were the same as in the first experiment.

The participants were instructed to point at the target by gaze as accurately as possible, and press the space bar before the target disappears. We did not mention to them that they should remain fairly still and avoid large or sudden body movements, as we usually do in other tests related to eye tracking. The participants were also not informed that they can use head movements to improve the pointing accuracy. Therefore, they were not restricted in their behavior during the test in any way.

The block of trials with the $GCA_{HM}$ algorithm activated was always completed first. Completing first the trials without it could give a wrong impression to the participants, indicating that head movements do not impact on the location of gaze cursor in case they would try to do so – especially, as the instructions were same for both blocks of trials.

### 5.2 Results and Discussion

Differently from the analysis of the data collected in the first experiment, we did not rejected data from the first trials. The selection time ST, distance D and artificial target hit rate $TH^R$ were analyzed as dependent variables on the only factor of two states: $GCA_{HM}$ algorithm on or off.

None of the participants noticed that they can use head move-

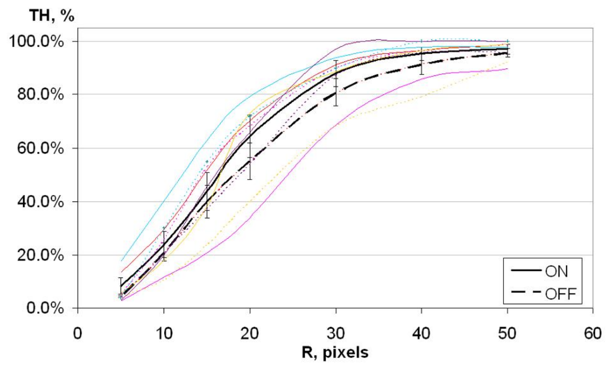

**Figure 5** *Experiment 3: The percentage of gaze cursor allocations within R pixels from a target center.*

ments to control the gaze cursor. Therefore, our hypothesis was not confirmed: it may take more time than the 4-5 minutes (the time spent to point at all 100 targets in a block) for novices to notice the effect of the $GCA_{HM}$ algorithm by a chance. It also appears that contrary to our prior anecdotal evidence people do not consistently apply "pointing by head" when the accuracy of eye pointing is insufficient.

Four of the participants kept their head very still during the test, while others were less restrained in their head movements. We found no evidence of a correlation between this behavior and their earlier participation in studies with eye tracking.

Even though the participants were not intentionally using the head movements to correct the gaze cursor location, we found that the $GCA_{HM}$ algorithm still slightly increased the pointing accuracy (see Figure 5). The average distances were 16.2 (SD = 2.8) pixels for "ON" and 20.6 (SD = 4.7) for "OFF" but the difference was not statistically significant. The selection times were about equal (M ≈ 1500, SD ≈ 300 ms).

## 6 Conclusions and Future Work

Our experiments showed that gaze cursor location correction by head movements dramatically improves pointing accuracy without hurting performance for most users. Head-assisted eye pointing makes it possible to select small interface objects that would otherwise require time consuming techniques such as zooming.

Part of the advantage of the proposed method comes from the fact, that it is applicable with any video-based eye tracking system that provides eye position in its camera view. Therefore, no extra devices or systems are required for recording the head movements. However, for example added face tracking might help to maintain the accuracy of both head and eye tracking in the long run and enable cursor control even if the gaze is lost momentarily. For simplicity, in this experiment we only used data given by the eye tracker.

We also found that for some users the head movements themselves may serve as a sufficient method to increase the pointing accuracy, although we cannot explain how exactly this happens and why only a part of users can utilize them. For example, the amount of experience did not explain this since some highly experienced eye tracker users were not able to exploit this technique. Further research is needed on what exactly some people do to achieve this. We could only speculate that the ability to naturally control the gaze cursor by head movement somehow depends on the eye-ball shape and thus on IR reflection from eyes. Furthermore, the tested algorithm may be safely activated for all users, as it does not hurt the pointing accuracy even if the user does not know about its availability. Users have to be informed about this feature in advance to benefit from it. However, learning and using the method is easy.

The time cost of the cursor location correction by head movements was less than 400 ms on average. The duration increases as the inaccuracy of the tracker data increases. The cost is comparable or lower that what is required by other methods used to increase the accuracy of pointing by gaze (refer, for example, to [Skovsgaard et al. 2010] and [Miniotas et al. 2006]) but further research is needed before any conclusions can be drawn of the superiority or inferiority of the proposed method. However, we believe that head movements, if possible to use, are a more natural and more comfortable way of improving the accuracy of gaze pointing than other methods that cause changes in the interface and require additional selections.

Since it is not clear whether the proposed method can improve the pointing accuracy without disproportionately increasing the pointing time, and thus decreasing the overall performance, we plan to conduct a study using a conventional Fitts' law -based pointing device test setup. This should give us a better insight into how the use of gaze cursor correction by head movement affects the throughput of this input channel. We also plan to include the mouse as a third pointing technique to have a well-known baseline for comparison.

It is important to note that the head-assisted eye pointing is most efficient if used with a selection method other than the dwell time. Users should be able to trigger a selection command "on the fly" (i.e., while the cursor passes through a target without a stop). A simple head gesture (e.g., nod) detected from the eye position in the camera view is one possible solution [Špakov and Majaranta 2012]. Head movements down and up will not hurt the quality of the eye tracking data, if the target for selection is chosen according to the one in focus right before the selection gesture started. On the other hand, the system needs to filter out unintentional, coarse head movements. There are differences in how well the eye tracking systems tolerate head movements and compensate for them. In any case, it is possible improve the robustness of the algorithm by only including adjustments that are made while the gaze is fairly stable. Experimenting with various real life use scenarios in long term use is a topic for future research.

The usefulness and scope of applications increase substantially if the head-assisted gaze pointing is used together with other modalities, e.g. for quick effortless annotation of photos.

We have shown that simple head movements can significantly improve the accuracy of eye pointing. We believe that there is still a lot of unexplored potential in exploiting the combination of eye pointing and different head movements in gaze-based user interfaces.

## 7 Acknowledgements

We appreciate our colleague [omitted for blind review] for expressing the original idea of using head movements for correcting the gaze cursor. The work was funded by [omitted for blind review]. We also thank the anonymous reviewers for their suggestions for improvements and ideas for further application of this technique.

## References

ASHMORE, M., DUCHOWSKI, A.T., AND SHOEMAKER, G. 2005. Efficient eye pointing with a fisheye lens. In *Proceedings of Graphics Interface 2005, GI'05*, Canadian Human-Computer Communications Society, University of Waterloo, Canada, 203-210.

BATES, R. AND ISTANCE, H. O. 2003. Why are eye mice unpopular? A detailed comparison of head and eye controlled assistive technology pointing device. *Universal Access in the Information Society 2*, 3, 280-290.

BIEDERT, R., DENGEL, A., AND KÄDING, C. 2012. Universal eye-tracking based text cursor warping. In *Proceedings of the 2012*

*Symposium on Eye Tracking Research and Applications, ETRA '12*, ACM, 361-364.

FONO. D. AND VERTEGAAL, R. 2005. EyeWindows: evaluation of eye-controlled zooming windows for focus selection. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '05*, ACM, 151-160.

HANSEN, J. P., HANSEN, D. W., AND JOHANSEN, A. S. 2001. Bringing gaze-based interaction back to basics. In *C. Stephanidis (Ed.), Universal Access in HCI: Towards an Information Society for All (Volume 3 of the Proceedings of the 9th International Conference on Human-Computer Interaction),* Lawrence Erlbaum Associates, 325-328.

HANSEN, D. W, AND JI, Q. 2010. In the eye of the beholder: A survey of models for eyes and gaze. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 3, 478-500.

KUMAR, M., KLINGNER, J., PURANIK, R., WINOGRAD, T., AND PAEPCKE. A. 2008. Improving the accuracy of gaze input for interaction. In *Proceedings of the 2008 Symposium on Eye Tracking Research & Applications, ETRA '08*, ACM, 65-68.

KUMAR, M., PAEPCKE, A., AND WINOGRAD, T. 2007. EyePoint: practical pointing and selection using gaze and keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '07*, ACM, 421-430.

LANKFORD, C. 2000. Effective eye-gaze input into Windows. In *Proceedings of the 2000 Symposium on Eye Tracking Research & Applications, ETRA '00*, ACM, 23-27.

MARDANBEGI, M., HANSEN, D.W, AND PEDERSON, T. 2012. Eye-based head gestures. In *Proceedings of the 2012 Symposium on Eye Tracking Research & Applications, ETRA '12*, ACM, 139-146.

MINIOTAS, D., ŠPAKOV. O., AND MACKENZIE, I.S. 2004. Eye gaze interaction with expanding targets. In *Extended Abstracts on Human Factors in Computing Systems, CHI'04*. ACM, 1255-1258.

MINIOTAS, D., ŠPAKOV, O., TUGOY, I., AND MACKENZIE, I.S. 2006. Speech-augmented eye gaze interaction with small closely spaced targets. In *Proceedings of the 2006 Symposium on Eye Tracking Research & Applications, ETRA '06*, ACM, 67-72.

MONDEN, A., MATSUMOTO, K., AND YAMATO, M. 2005. Evaluation of gaze-added target selection methods suitable for general GUIs. *International Journal of Computer Applications in Technology 24*, 1, Inderscience Publishers, 17-24.

ROZADO, D. 2013. Mouse and keyboard cursor warping to accelerate and reduce the effort of routine HCI input tasks. *IEEE Transactions on Human-Machine Systems 43*, 5, 487-493.

SKOVSGAARD, H., MATEO, J. C., FLACH, J. M., AND HANSEN, J. P. 2010. Small-target selection with gaze alone. In *Proceedings of the 2010 Symposium on Eye Tracking Research & Applications, ETRA '10*, ACM, 145-148.

SKOVSGAARD H., RÄIHÄ K.-J., TALL, M. 2012. Computer control by gaze. In *P. Majaranta et al. (eds.) Gaze Interaction and Applications of Eye Tracking: Advances in Assistive Technologies*, IGI Global, 78-102.

ŠPAKOV, O. 2012. Comparison of eye movement filters used in HCI. In *Proceedings of the 2012 Symposium on Eye Tracking Research & Applications, ETRA '12*, ACM, 281-284.

ŠPAKOV, O. AND MAJARANTA, P. 2012, Enhanced gaze interaction using simple head gestures. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing, UBICOMP'12*, ACM, 705-710.

ŠPAKOV, O., AND MINIOTAS, D. 2005. Gaze-based selection of standard-size menu items. In *Proceeding of International Conference on Mumtimodal Interfaces, ICMI'05*, ACM, 124-128.

STELLMACH, S. AND DACHSELT, R. 2012. Look & touch: gaze-supported target acquisition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '12*, ACM, 2981-2990.

STELLMACH, S. AND DACHSELT, R. 2013. Still looking: investigating seamless gaze-supported selection, positioning, and manipulation of distant targets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '13*, ACM, 285-294.

TURNER, J., BULLING, A., AND GELLERSEN, H. 2011. Combining gaze with manual interaction to extend physical reach. In *Proceedings of the 1st international workshop on pervasive eye tracking & mobile eye-based interaction, PETMEI '11*, ACM, 33-36.

WARE, C., AND MIKAELIAN, H.H. 1987. An evaluation of an eye tracker as a device for computer input. In *Proceedings of the SIGCHI/GI Conference on Human Factors in Computing Systems and Graphics Interface, CHI '87*, ACM, 183-188.

ZHAI, S., MORIMOTO, C., AND IHDE, S. 1999. Manual and gaze input cascaded (MAGIC) pointing. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems, CHI'99*, ACM, 246-253.

ZHANG, X., REN, X., AND ZHA, H. 2008. Improving eye cursor's stability for eye pointing tasks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '08*. ACM, 525-534.