Real-Time Hidden Gaze Point Correction

Oleg Špakov, Yulia Gizatdinova
TAUCHI, School of Information Sciences,
University of Tampere
{firstname.lastname}@uta.fi

Abstract

The accuracy of gaze point estimation is one of the main limiting factors in developing applications that utilize gaze input. The existing gaze point correction methods either do not support real-time interaction or imply restrictions on gaze-controlled tasks and object screen locations. We hypothesize that when gaze points can be reliably correlated with object screen locations, it is possible to gather and leverage this information for improving the accuracy of gaze pointing. We propose an algorithm that uses a growing pool of such collected correlations between gaze points and objects for real-time hidden gaze point correction. We tested this algorithm assuming that any point inside of a rectangular object has equal probability to be hit by gaze. We collected real data in a user study to simulate pointing at targets of small (<30px), medium (~50px) and large (>80px) size. The results showed that our algorithm can significantly improve the hit rate especially in pointing at middle-sized targets. The proposed method is real-time, person- and task-independent and is applicable for arbitrary located objects.

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

Keywords: eye tracking; pointing; algorithms; gaze point; accuracy correction; cumulative distribution function

1 Introduction

The accuracy of gaze point estimation is a permanent issue in the development of user interfaces that utilize gaze input. Limited by biological features of the human vision system, the pointing accuracy of modern eye trackers is far behind that of mouse or touchpad. Perhaps the simplest way to overcome the inaccuracy issue is to discard gaze as the main technique used for precise pointing. In such systems gaze remains as a mean for fast yet imprecise pointing, whereas manually operated pointing devices are used for final adjustment of the cursor position [Zhai et al. 1999, Monden et al. 2005]. Few studies [e.g. Miniotas et al. 2006] applied a limited set of voice commands for correcting cursor position when the accuracy of gaze pointing was insufficient [Miniotas et al. 2006]. Speech-augmented gaze pointing supported a completely hands-free interaction but was rather slow. Another way to deal with the inaccuracy issue is to make graphical objects large enough to compensate for imprecise gaze pointing [Bates and Istance 2002, Ashmore et al. 2005, Skovsgaard et al. 2010]. The proposed zooming methods resulted in the increased accuracy of gaze pointing at the cost of the increased pointing time.

Further, studies were dedicated to silently correct gaze point by attracting it to the nearest target [Miniotas et al. 2004, Zhang et al. 2008]. However, this solution is applicable only if objects are located sparsely (i.e. there are spatial gaps between the objects). An efficient way of improving gaze pointing accuracy was demonstrated by Mackenzie and Zhang [2008]. They used a next-letter prediction in text typing by gaze for adjusting a virtual size of buttons, thus manipulating the probability to hit a particular target. Another example of a neat way to bind gaze point and an object was presented by Vidal et al. [2013] who used correlation between trajectories of moving targets and eye movements. Unfortunately, the described two methods can be applied only if a gaze-controlled task can be modeled.

It has been shown [Hornoff and Halverson 2002] that a spatio-temporal correlation exists between gaze points and object screen locations, called required fixation locations (RFL). Implicit RFLs correspond to objects (e.g., buttons, menus and other interface elements) that the user must look at in order to successfully accomplish a task. The authors suggested examining a disparity between a gaze point and the nearest RFL to monitor the deterioration in the calibration accuracy and automatically invoke a recalibration procedure when necessary. RFLs were also found useful in measuring participant's unique error signature (i.e. drift in calibration) to reduce a systematic error in the eye movement data collected for that participant. We take the idea of implicit RFLs further and hypothesize that when gaze points can be reliably correlated with RFLs, it is possible to leverage this information for improving the accuracy of gaze pointing on a fly without the need for a recalibration. We propose a person- and task-independent method that uses a growing pool of such collected links between gaze points and implicit RFLs for real-time hidden gaze point correction.

2 Correction Algorithm

In gaze-based interfaces, selection is typically done by gazing at a target for some period of time. As it has been shown in [Hornoff and Halverson 2002], selection of certain key elements of the interface can be treated as reliable with nearly 100% confidence (i.e., a user must look at the object in order to proceed with a task). In this case, *RFL* that is the center of such an object, the gaze point G_{RFL} that triggers a selection of the object and the disparity error distance D between G_{RFL} and *RFL* can be measured and recorded into a database. The number of records in the database increases with time. It is expectable that the database will contain multiple D_i entries for a single *RFL* (see Figure 1, left). Every newly calculated D_i will result into a new record in the database. The corrected gaze point $G' = G'(t)$ is calculated as a sum of the current gaze point $G = G(t)$ and a weighted mean of N recorded disparity error distances D_i:

$$G' = G + \frac{\sum_{i=1}^{N} W_i \cdot D_i}{\sum_{i=1}^{N} W_i}, \qquad \sum_{i=1}^{N} W_i \neq 0 \qquad [1]$$

where i denotes the index of a record from the database and W_i is a weight assigned to the corresponding D_i entry.

According to Equation 1, D_i entries with high weights will contribute more to the adjustment of the current gaze point than entries with low weights will do. The weights are Gaussian functions which depend on the distance between the current gaze point G and the recorded gaze point G_{RFL}: large weights are given to those records which have the recorded G_{RFL} locations in close proximity to the current gaze point and vice versa. In case of exceptionally large distances between G and G_{RFL} the weight can be set to 0, thus completely eliminating these records from the calculation of G':

$$W_i = \begin{cases} e^{-d_i^2/2\sigma^2}, & d_i \leq 2\sigma \\ 0, & d_i > 2\sigma \end{cases}, \quad d_i = |G - G_{RFL_i}| \qquad [2]$$

where σ spans the largest yet realistic for the interaction value of d_i that can be determined empirically.

Equation 1 is correct only if targets are very small and can be treated as single points. Further we propose an algorithm that allows utilizing the recorded *RFL*, G_{RFL} and D to correct gaze point also for arbitrary-sized objects. For this, we assume that any point inside of a rectangular target has equal probability to be hit by gaze and, therefore, we apply a probabilistic method to compute the corrected gaze point G'. For this, an intersection rectangle $R = T_{RFL} \cap T'$ is analyzed, where T' is a "shaded" target that has the same size as the current target T but is located in the same way relatively to G as T_{RFL} is located relatively to G_{RFL} (see Figure 1, right). If the intersection is not empty, a probability P of the current gaze point G falling into this intersection is estimated. If P is high enough, it is stated that the gaze hits the current target. The cumulative distribution function F with $\sigma_{CDF} = 50$ is used to estimate P separately for X and Y coordinates. The resulting values of P_X and P_Y are the normalized differences between the values of F, when right and left (or bottom and top) values of rectangle R are used as parameters (see also Algorithm 1, *function CalcP*):

$$P_X = \frac{F(R.right) - F(R.left)}{F(T.right) - F(T.left)}, \quad P_Y = \frac{F(R.bottom) - F(R.top)}{F(T.bottom) - F(T.top)} \qquad [3]$$

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx \qquad [4]$$

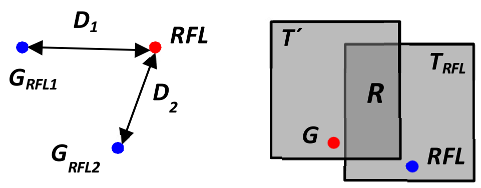

Figure 1 Left: *Disparity error distances D_i in selecting RFL.*
Right: *Calculation of the intersection R between T_{RFL} and T'*

Probabilities P_X and P_Y are then multiplied by two weights, W_X and W_Y, correspondently. Both weights consist of two multipliers that are calculated according to Equation 2. The first multiplier is the same for both weights and accounts for d distance between G and G_{RFL}. The second multiplier accounts for the width of T_{RFL} in case of W_X and the height of T_{RFL} in case of W_Y. Then the probability P of the current gaze point G hitting the current target is calculated as a product of sums of all P_X and P_Y divided by the sum of the corresponding weights:

$$P = \frac{\sum_{i=1}^{N} W_{X_i} \sum_{i=1}^{N} P_{X_i}}{\sum_{i=1}^{N} W_{X_i}} \cdot \frac{\sum_{i=1}^{N} W_{Y_i} \sum_{i=1}^{N} P_{Y_i}}{\sum_{i=1}^{N} W_{Y_i}} \qquad [5]$$

$$\sum_{i=1}^{N} W_{X_i} \neq 0, \quad \sum_{i=1}^{N} W_{Y_i} \neq 0$$

Algorithm 1 *The pseudo code for the correction algorithm*

```
given $G$, $T$, $selections$, $\sigma_{CDF}$, $\sigma_{RFL}$, $\sigma_D$
define $p$ as array

function CalcP ( $G_{RFL}$, $T_{RFL}$, $G$, T )
    var $d \leftarrow$ distance between $G_{RFL}$ and $G$
    var $W_D \leftarrow \exp(-d^2 / (2 \cdot \sigma_D^2))$, $\sigma_D = 150$
    var $W_{SIZE\_X} \leftarrow \exp(-T_{RFL}.width^2 / (2 \cdot \sigma_{RFL}^2))$, $\sigma_{RFL} = 85$
    var $W_{SIZE\_Y} \leftarrow \exp(-T_{RFL}.height^2 / (2 \cdot \sigma_{RFL}^2))$
    var $\{ W_X, W_Y \} \leftarrow \{ W_{SIZE\_X} \cdot W_D, W_{SIZE\_Y} \cdot W_D \}$
    var $R \leftarrow$ FindIntersection ( $G_{RFL}$, $T_{RFL}$, $G$, $T$ )

    var $P_X \leftarrow$ CalcPX ( $G_{RFL}$, $R$, $G$, $T$ ) $\cdot W_X$
    var $P_Y \leftarrow$ CalcPY ( $G_{RFL}$, $R$, $G$, $T$ ) $\cdot W_Y$
    return $\{ P_X, P_Y, W_X, W_Y \}$

function CalcPX ( $G_{RFL}$, $R$, $G$, $T$ )
    var $S_R \leftarrow$ GetArea ( $R.left$, $R.right$, $G_{RFL}.x$ )
    var $S_T \leftarrow$ GetArea ( $T.left$, $T.right$, $G.x$ )
    return $S_T > 0$ ? $S_R / S_T$ : 0

function CalcPY ( $G_{RFL}$, $R$, $G$, $T$ )
    var $S_R \leftarrow$ GetArea ( $R.top$, $R.bottom$, $G_{RFL}.y$ )
    var $S_T \leftarrow$ GetArea ( $T.top$, $T.bottom$, $G.y$ )
    return $S_T > 0$ ? $S_R / S_T$ : 0

function FindIntersection ( $G_{RFL}$, $T_{RFL}$, $G$, $T$ )
    define $R$ as rectangle
    $R.left \leftarrow G_{RFL}.x + ( T.left - G.x )$
    $R.top \leftarrow G_{RFL}.y + ( T.top - G.y )$
    $R.right \leftarrow R.left + T.width$
    $R.bottom \leftarrow R.top + T.height$
    $R.left \leftarrow$ max ( $T_{RFL}.left$, $R.left$ )
    $R.top \leftarrow$ max ( $T_{RFL}.top$, $R.top$ )
    $R.right \leftarrow$ max ( $R.left$, min ( $T_{RFL}.right$, $R.right$ ) )
    $R.bottom \leftarrow$ max ( $R.top$, min ( $T_{RFL}.bottom$, $R.bottom$ ) )
    return $R$

function GetArea ( $from$, $to$, $mean$ )
    return CDF ( $to$, $mean$ ) – CDF ( $from$, $mean$ )

function CDF ( $x$, $mean$ )
    var $a \leftarrow$ [0.2548295, –0.2844967, 1.4214137, –1.4531520, 1.0614054]
    var $p \leftarrow$ 0.3275911
    $x \leftarrow (x - mean) / (\sigma_{CDF} \cdot \sqrt{2})$
    var $t \leftarrow 1.0 / ( 1.0 + p \cdot abs(x) )$
    var $erf \leftarrow 1 - (((( a_4 \cdot t + a_3 ) \cdot t + a_2 ) \cdot t + a_1 ) \cdot t + a_0 ) \cdot t \cdot \exp(-x^2)$
    return ( 1 + sign of $x \cdot erf$ ) / 2

foreach ( $G_{RFL}$, $T_{RFL}$ ) in $selections$
    var $\{ P_X, P_Y, W_X, W_Y \} \leftarrow$ CalcP ( $G_{RFL}$, $T_{RFL}$, $G$, $T$ )
    add $\{ P_X, P_Y, W_X, W_Y \}$ to $p$

return ( sum of $P_X$ / sum of $W_X$ ) * ( sum of $P_Y$ / sum of $W_Y$ )
```

The exact decision of whether the calculated probability denotes that the gaze point falls into the target depends on its size and, more importantly, on the probabilities calculated for other neighboring objects. In other words, the absolute *P* value calculated for a certain object makes sense only in relation to other closely located objects. Therefore, the verification of the ability of this algorithm to improve gaze-to-object mapping requires a procedure in which there are many candidates to be recognized as an object in focus. We verified this idea by running a test in which we gathered gaze points of participants observing small dots, and then mapped it on larger virtual targets surrounded by eight other virtual target candidates.

3 Algorithm Verification

3.1 Collecting Gaze Data

Twelve volunteers experienced in using eye tracking technology took part in the test. Target presentation and data collection from Tobii T60 eye tracking system were implemented in a webpage using JavaScript to be shown in the Mozilla Firefox browser with a plugin for ETU-Driver[1] installed. The chair was set so that the participant's eyes were at approximately 60 cm from the 17-inch monitor (screen resolution was 1280x1024 pixels).

A full-screen webpage with a grey background was divided virtually into 4x5 grid with 20 cells. Targets T_D were presented in random location in each cell in random order. They were displayed as black circles of radius ø = 6 pixels to ensure the direction of participants' gaze. One block of the experiment consisted of presenting 200 targets (10 targets per cell).

We used three setups (2, 5 and 9 points) when calibrating the system to get the maximum use out of a single system. We also hypothesized that the proposed algorithm could be especially beneficial in case when the gaze direction estimation procedure suffers from user movements, such as turning off the screen to perform some actions, or even shortly leaving the chair. Each participant performed three blocks of tasks per calibration, and between blocks we asked them 1) to reach a pen on the table forcing them to change body position still remaining in the chair, and 2) stand up, walk three meters away and then sit back in the chair. We did not recalibrate the eye tracking system between the blocks; however, before starting the second and the third blocks we ensured that eyes are visible for the eye tracker. We expected that with the increase of the block ordinal number and the decrease of the number of calibration points, our real-time hidden gaze point correction (RTHGC) algorithm will show an increasing efficiency in mapping gaze points to the targets.

At the beginning of the experiment a supervisor explained the task for participants (simply stair at dots appearing in random places on the screen). Then the eye tracker was calibrated. The order of calibration setups was counterbalanced using Latin square. After the calibration was recognized as successful, the first block started by displaying the first point. The experiment continued displaying 200 targets one by one. Each target remained visible for 1.3 seconds; they were shown and hidden automatically. The gaze point was not displayed. After the first block was finished, participants had a chance to rest, if needed, and then made the required movement remaining in the chair. After the second block participants walked across the laboratory and then continued with the third block. Then the eye tracker was recalibrated using another number of calibration points, and participants continued with another three blocks. Finally, after the third recalibration and next three blocks the experiment was over. Gaze points (timestamp, location) and the displayed target properties (location, size) were logged. The experiment took about one hour. Most of the participant took 2-3 minutes of resting, typically before calibrating the system for the third time.

3.2 Emulation of Large Target Selection

The collected data was used to simulate gazing at a rectangular target and eight surrounding competitors (no gaps). The location of a simulated target T_{RFL} was random, yet the displayed target T_D was located completely inside of it. We supposed that the ambiguity in the simulated target location models the gaze pointing better than the assumption that a user is always looking at the center of a target despite its dimensions and content. This assumption also implies that every pixel of a target has an equal probability to be gazed. We have simulated nine sizes of targets, ranging from 16 to 144 pixels with a step of 16 pixels.

A naïve gaze-to-object mapping was used for the first target in a block. The list of selection for the RTHGC algorithm (applied starting from the second target) was populated by the records (G_{RFL}, T_{RFL}) consisted of the fixation detected by dispersion based algorithm (D=50 pixels) and the size and location of T_{RFL}. We excluded the records with the distance between G_{RFL} and the center of T_D greater than 100 pixels. There was no threshold set to allow the algorithm to reject all candidates.

The RTHGC algorithm was applied to all nine simulated objects. The object with the highest *P* was recorded as the mapping result. Also, we recorded the result of naïve mapping for each trial.

4 Results and Discussion

First of all, we removed the invalid gaze points from the collected data. The amount of invalid points was about the same in each block (μ_{inv}=0.25%, σ_{inv}=0.15%). The grand mean rate of hitting a target *HR* using naive algorithm was HR_N = 42% (σ = 5.7%). The RTHGC algorithm increased it by 15.7% in average (HR_{RTHGC} = 57.7%, σ = 5.7%), and by 20% for the last 25 trials. It was beneficial for each participant (at least 4.8%, at most 30%, see Figure 2) and in each condition (see Figure 3), although in three blocks (out of 81) the mapping accuracy has slightly worsened (-2.5% at most). In few blocks the improvement was over 40%.

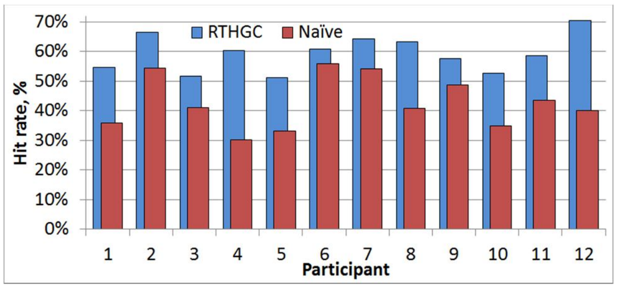

Figure 2 *Hit rates for naive and RTHGC algorithms, by participant*

[1] Downloaded from http://www.sis.uta.fi/~csolsp/downloads.php

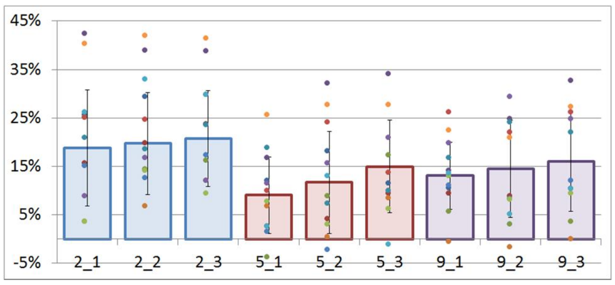

Figure 3 *Improvement in hit rates, by condition*
Bars: mean values with STD. Dots: per participant

The data analysis showed that the block number had no statistically significant impact ($F_{2,8}$=1.03, p=0.36) on *HR* when either mapping algorithm (i.e. RTHGC or naïve) was applied, meaning that movements done by participants did not aggravate much the tracking quality. However, the mean HR_N was lower with every next session (within the same calibration condition), and the RTHGC algorithm improved it by 13.5%, 14.5% and 16% accordingly.

The number of calibration points had a significant effect on *HR* ($F_{2,8}$=9.29, p<0.001). Interestingly, the 5-points calibration resulted, in average, in slightly more accurate tracking than 9-points calibration, and the RTHGC algorithm improved the *HR* by about 11% and 13% accordingly. For 2-points calibration condition our algorithm improved the *HR* by almost 20%.

The number of trials when the RTHGC algorithm corrected the mapping result of the naïve algorithm was about 23.5%, and it showed a clear dependency on the simulated target size (see Figure 4). When targets were of 48 pixels size, the improvement was about 23% (maximum), while it was only about 8% when targets were of 16 or 144 pixels size. We found that the equation

$$\boldsymbol{HR_{RTHGC}} - \boldsymbol{HR_N} = \frac{\mathbf{0.006}x}{\mathbf{1}+(\mathbf{0.01467}x)^{\mathbf{3}}}, \quad x = [16 \div 144, 16] \quad [6]$$

models the observed effect quite well (R=0.991). We found no regularity or dependency on either factor in cases when RTHGC mapped gaze points incorrectly, while the naive mapping results in a correct hit (about 8% of trials).

As expected, the RTHGC algorithm performed best when the original accuracy was low. The location-based analysis supports this finding: most of the mapping improvements have occurred when the target was shown on the left side of a screen, the area of the least eye tracking accuracy for our equipment as it has been found by our previous study (not published yet).

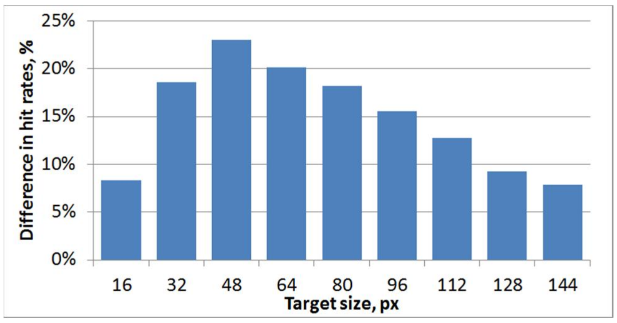

Figure 4 *Hit rate improvement by RTHGC, by target size*

5 Conclusions

This paper describes the algorithm that utilizes the history of reliable gaze-to-object mappings to improve the hit rate of targets surrounded by several candidates by 23.5% on average in comparison to the naïve mapping when middle-size targets (48 pixels) were used. It works in real-time, is independent of a task and object locations, and does not penalize the performance in term of the increased pointing task. We suppose that this algorithm may work in parallel with other algorithms that improve gaze pointing, although a new study should be conducted to inspect this idea. Such study could also verify its performance in real-application condition.

References

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

BATES, R. AND ISTANCE, H. 2002. Zooming interfaces!: enhancing the performance of eye controlled pointing devices. In *Proceedings of ASSETS '02*, ACM, 119-126.

HORNOF, A. J., AND HALVERSON, T. 2002. Cleaning up systematic error in eye tracking data by using required fixation locations. *Behavior Research Methods, Instruments, and Computers*, 34 (4), 592-604.

MACKENZIE, I. S. AND ZHANG, X. 2008. Eye typing using word and letter prediction and a fixation algorithm. In *Proceedings of ETRA '08*. ACM, 55-58.

MINIOTAS, D., ŠPAKOV. O., AND MACKENZIE, I.S. 2004. Eye gaze interaction with expanding targets. In *Proceeding of CHI EA '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 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.

SKOVSGAARD, H., MATEO, J. C., FLACH, J. M., AND HANSEN, J. P. 2010. Small-target selection with gaze alone. In *Proceedings of ETRA '10*, ACM, 145-148.

VIDAL, M., PFEUFFER, K., BULLING, A. AND GELLERSEN, H. 2013. Pursuits: Eye-Based Interaction with Moving Targets. In *Proceedings of CHI EA '13,* ACM, 3147-3150.

ZHAI, S., MORIMOTO, C., AND IHDE, S. 1999. Manual and gaze input cascaded (MAGIC) pointing. In *Proceedings of 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 CHI '08*. ACM, 525-534.