2022 IEEE Conference on Virtual Reality and 3D User Interfaces (VR) | 978-1-6654-9617-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/VR51125.2022.00105

GazeDock: Gaze-Only Menu Selection in Virtual Reality using Auto-Triggering Peripheral Menu

Xin Yi*
Tsinghua University

Yiqin Lu†
Tsinghua University

Ziyin Cai‡
Beijing University of Post and Telecommunication

Zihan Wu§
Tsinghua University

Yuntao Wang¶
Tsinghua University

Yuanchun Shi‖
Tsinghua University

ABSTRACT

Gaze-only input techniques in VR face the challenge of avoiding false triggering due to continuous eye tracking while maintaining interaction performance. In this paper, we proposed GazeDock, a technique for enabling fast and robust gaze-based menu selection in VR. GazeDock features a view-fixed peripheral menu layout that automatically triggers appearing and selection when the user's gaze approaches and leaves the menu zone, thus facilitating interaction speed and minimizing the false triggering rate. We built a dataset of 12 participants' natural gaze movements in typical VR applications. By analyzing their gaze movement patterns, we designed the menu UI personalization and optimized selection detection algorithm of GazeDock. We also examined users' gaze selection precision for targets on the peripheral menu and found that 4–8 menu items yield the highest throughput when considering both speed and accuracy. Finally, we validated the usability of GazeDock in a VR navigation game that contains both scene exploration and menu selection. Results showed that GazeDock achieved an average selection time of 471ms and a false triggering rate of 3.6%. And it received higher user preference ratings compared with dwell-based and pursuit-based techniques.

Index Terms: Human-centered computing—Human computer interaction (HCI)—HCI design and evaluation methods—User studies; Human-centered computing—Human computer interaction (HCI)—Interaction paradigms—Virtual reality

1 INTRODUCTION

With the increasing popularity of virtual reality (VR) and augmented reality (AR), menu selection – one of the most basic interactions – has attracted more and more attention from both the academia and the industry. To support this, mainstream products tend to use handheld controllers or hand gestures. However, there is also a great demand for hands-free menu selection in VR/AR, as it is preferred or required when the users' hands are occupied by other tasks or disabled (situationally or permanently).

With recent advances in eye-tracking technologies, gaze interaction in VR/AR has become one of the major solutions for hands-free interaction. Different from using specialized interaction devices, gaze input actions are highly mixed with natural gaze movements, which introduces the challenge of balancing efficiency and robustness: efficient gaze input techniques require the designed gaze actions to be simple and fast, which, at the same time, causes the "Midas Touch" problem [23] (i.e., false triggering of the input).

Corresponding with this challenge, a crucial feature of practical gaze-based input techniques is the ability to coexist with other tasks: users should be able to switch between the current task (e.g., menu selection) and other interaction tasks. To achieve this, some researchers leveraged auxiliary input modalities (e.g., manual input [30,42]) to "enable" the technique before use, and "disable" them after use. However, these solutions cannot be applied to gaze-only interactions (e.g., people with disabilities). And although there were gaze-only techniques that facilitate the selection of items from a UI layout (e.g., dwelling on an optimized menu layout [24, 57]), it is not trivial to design interactions to "call out" the menu before selection. Alternatively, some researchers proposed to trigger commands without a menu layout (e.g., pursuit [25] and gaze gestures [5]), but at the cost of lower speed or higher memorizing efforts.

In this paper, we proposed **GazeDock**, a technique for enabling fast and robust gaze-only menu selection in VR. GazeDock features a **peripheral menu** – a ring-shaped menu layout fixed in the peripheral region of the user's field of view (FOV, see Fig. 10a), with menu items arranged radially. By default, GazeDock is invisible to the user to avoid distraction in other tasks. When the user's gaze approaches the periphery of his/her FOV, GazeDock will fade in automatically, and the menu item that is gazed at will be selected and highlighted. The selection would be confirmed and GazeDock fades out when the user moves his/her gaze out of the menu region.

GazeDock has three advantages: 1) The menu layout only occupies the periphery region of FOV when displayed, which minimizes the occlusion of background contents, and improves the immersion in VR scenes; 2) Selection on GazeDock can be completed without an explicit operation for menu triggering, which facilitates input efficiency and the users' learning process; 3) As we will show in this paper, the specially designed layout and gaze selection mechanism of GazeDock makes it highly distinguishable from users' natural gaze movements, making it compatible with other tasks in real VR applications.

We iteratively conducted three user studies when designing and evaluating GazeDock. In Study 1, we collected users' natural gaze movement data in four typical types of real VR applications, and analyzed the distribution of their gaze points. Based on the results, we designed the personalized menu interface and selection detection algorithms of GazeDock. In Study 2, we examined users' gaze selection precision on the peripheral menu, and optimized the layout of GazeDock. In Study 3, we validated the performance and usability of GazeDock in a VR game, where both exploration and menu selection tasks existed.

Our contributions were three-folded: 1) we examined the patterns of user's natural gaze movement in typical VR tasks, as well as their target selection ability on the peripheral of their FOV; 2) we proposed GazeDock, which leveraged a personalized peripheral menu and specially designed selection mechanism to enable fast and robust gaze input in VR; 3) we provided the first empirical results that evaluated and compared the usability of GazeDock, dwell-based

*e-mail: yixin@tsinghua.edu.cn
†e-mail: lu-yq16@mails.tsinghua.edu.cn
‡e-mail: caiziyin1998@gmail.com
§e-mail: ziwu@umich.edu
¶e-mail: yuntaowang@tsinghua.edu.cn, the corresponding author
‖e-mail: shiyc@tsinghua.edu.cn

978-1-6654-9617-9/22/$31.00 ©2022 IEEE
DOI 10.1109/VR51125.2022.00105

and pursuit-based techniques in VR applications where different interaction tasks coexist.

2 Related Work

2.1 Gaze-Based Input Techniques

A number of gaze-only approaches have been proposed including dwelling [4, 23, 46], gaze gestures [3, 5, 17, 21, 38, 61], gaze pursuit [9, 10, 25, 44, 55], vergence eye movement [27, 28] and hybrid [6, 20]. However, commercial products suffer from the "Midas Touch" problem [23], i.e., humans' natural and unintentional gaze movement may cause false triggering of these techniques. This problem becomes more severe in VR because of the continuous gaze tracking. Aiming at this problem, an explicit mode switch operation (e.g., button click) is often required to indicate intentional gaze input or trigger the "gaze mode" [30, 42]. In addition, the gaze input area was often separated from the main area [20] or device [16], and gaze input patterns were often complicated to avoid confusion with natural gaze movements [5]. As a result, previous works mainly focused on the gaze input performance in controlled tasks (e.g., target acquisition) while paying less attention to the performance in real scenarios where natural gaze movements exist. GazeBar [7] explored to leverage the edge of the PC monitor for robust gaze-based menu selection, which shared similar ideas with this work. However, it has not yet been optimized with regard to the user's gaze behavior, especially in virtual reality. In this work, we not only modelled the users' gaze behavior on a peripheral menu, but also provided the evaluation result of GazeDock in compound tasks where both natural gaze movement and intended gaze selection co-existed.

To improve the usability of gaze, researchers have also explored combining gaze input with other modalities (e.g., head movement). Previous researches [30, 45, 48, 56] have studied the performance of hybrid techniques with gaze and head movements in target acquisition tasks. Results showed pointing with only gaze was error prone, but could be improved by coupling with the head. Gaze input can also be enhanced by head turn actions [35, 39] or head gestures [34, 50] to increase the expressiveness or help the disambiguation. Manual input tasks could also be assisted by gaze (e.g., cursor control [41, 51, 60], interest identification [29] and remote pointing in VR [42]). Although effective in specific scenarios, unfortunately, these techniques could not be applied to gaze-only input tasks.

2.2 Menu Selection With Gaze

Menu selection is a fundamental input task in VR. Menus in VR are often presented in a view-fixed layout [36] to support gaze selection despite of head rotation. Previous works have explored different kinds of menus. Gaze on a standard-size linear menu could be improved by the dynamic expansion of menu items [57] with a 39% speed increase and higher accuracy. However, the problem of false triggering has not been considered. A pie menu works well for gaze input due to its circular range for selection. Previous researches [24, 54] have tested the selection performance on such menus with different layout, densities and input modalities. Results showed that gaze-only input was superior to gaze+speech input, and the optimal number of items on a pie menu was six. Snap Clutching [20] allowed the users to glance at different directions to enter different input modes, and then use dwelling to trigger different commands. This design could avoid false triggering in most cases. However, the performance of this technique was not formally evaluated. In comparison with these works, in this paper, we explored gaze-based menu selection on a novel menu layout where items are located radially on the periphery of FOV, and we tested the user's interaction performance with different settings. The results not only served as foundations of GazeDock, but also could benefit other gaze-based VR techniques.

2.3 Other Hands-Free Input Techniques in VR

Hands-free input is needed when users' hands are engaged in other tasks. Voice input is a common approach for command invoking on smart devices, but it suffers from environmental noise and lower privacy. In comparison, head movement is more widely-adopted in VR and comes with the ability of motion tracking with only the headset device. Input techniques using only head [8] have been well explored in literature, which often served as the substitute of hand input (head tilt for navigation [52] and head gesture for command input [58]). Input techniques using other body parts have also been proposed. For example, VR navigation can be achieved *in situ* by walking-in-place with feet [53] or leaning the torso [31]. Among these techniques, Periphery Menus [37] was the most similar with GazeDock, which allowed the users to rotate their heads to call out the hidden menu. However, they did not optimize the menu layout or selection mechanism for gaze input.

2.4 Understanding Gaze Behaviors

To better understand human's gaze behaviors, we mainly go through researches in the neuroscience field. Freedman [12] found that human's natural eye movement range was within 40˜45° from the center, while head movement would be involved for wider targets. Bdler et al. [2] found that when playing games in virtual environments showed on computer screens, user's gaze positions mostly distributed in the central region, but peripheral gaze also existed. Recently, Sitzmann et al. [49] found that gaze statistics in VR seemed to be in good agreement with those in conventional displays. However, the results mainly focused on saliency rather than gaze movement.

Eye-head coordination is an important feature when analyzing gaze movement. In a study conducted in a natural environment with rhesus monkey subjects [13], researchers found that in general, eye movement did not exceed an amplitude of 35° because head movement would contribute more when the shift of the focused target was larger. A study about eye, head and torso coordination during gaze shifts [47] showed more than 90% of eye-in-head angles were distributed in -20˜20°, and the authors argued that eye, head and torso movements are not unimodal but coupled. A comparison of eye-head coordination between VR and physical world [40] showed head movements played more roles in VR when viewing stimuli.

In this work, we collected and analyzed users' natural gaze data in real VR scenarios, which would be helpful for the design and implementation of VR techniques.

3 Study 1: Examining Natural Gaze Behaviour in VR

In this study, we examined users' natural gaze movement pattern in typical VR applications, with the aim to facilitate the design of false-triggering-prevention algorithms for GazeDock. The collected data can also be helpful for future researchers to analyze other patterns or to test the false-triggering-prevention ability of their own techniques.

3.1 Apparatus

We used HTC Vive Pro Eye [1] as the apparatus. The resolution of the display was 1440×1600 pixels per eye (2880×1600 pixels combined) with an FOV of 110°. The embedded eye-tracking system in the device can detect the user's gaze direction with an error of 0.5-1.1° in the frequency of 120Hz. The trackable field of view was also 110°. We recorded gaze data with the C++ interface of HTC SRanipal SDK.

3.2 Participants

We recruited 12 participants (8 male, 4 female, aged 19-28) from the campus. Participants had experience with VR for 2.2 years on average, ranging from 0 to 4 years.

3.3 Experiment Design and Procedure

In order to observe the most natural gaze movement pattern, the participants were asked to use different VR applications to complete the corresponding tasks freely. Referring to previous works [62], we chose four different types of applications (see Fig. 1) which covered a majority of fundamental interactions in VR (e.g., navigation, object manipulation, distant pointing and menu selection): 1) Island Journey [19]: a first-person shooting game. The task was to look around and shoot the enemies with the gun (VR controller) on the user's hand; 2) Drop [18]: a working simulation application. The task was to type on a virtual keyboard to search for three news websites and one video website, and browse the contents; 3) Rec Room [15]: a multi-player online platform. The task was to transport between two rooms and explore them, as well as interacting with objects in the room (e.g., stairs, drawers and basketballs); 4) Blocks [14]: a 3D design tool. One hand controls a drawing cursor and the other hand is used as a toolbox. The task was to try all tools and build a creation.

Each participant used the four applications in random order. For each app, they first familiarized themselves with the application for several minutes, and then used it for 5–10 minutes to complete the tasks. A short break was enforced between different applications. The whole experiment lasted for about forty minutes.

Figure 1: The four applications used in Study 1: (a) Island Journey, (b) Drop, (c) Rec Room, (d) Blocks.

3.4 Results

In total, we collected 2,873,288 frames, 399 minutes of gaze data from all 12 participants. We obtained the 3D direction of the participant's gaze (x,y,z) in each frame, and calculated gaze angle θ and polar angle ϕ using:

$$\begin{cases} \theta = \arccos((x,y,z)\cdot(0,0,1)) = \arccos(z) \\ \phi = \text{atan2}(y,x) \end{cases} \quad (1)$$

We labeled all the data by participant ID and application, and made the dataset publicy available,[1] which would be helpful for not only understanding user's gaze behaviors, but also designing and evaluating future gaze-based interaction techniques (e.g., testing the false triggering rate).

3.4.1 General Gaze Distribution

We visualized the collected gaze points as a heat map in Fig. 2. A large proportion of gaze points gathered near the center of FOV (the area colored in red), and most of the gaze points distributed in a slightly flat elliptical range. This result was in line with the range of eye movement from previous works [26], where the range of gaze was horizontally symmetric, while the range in up direction was smaller than that in down direction. Furthermore, we examined the range including a certain proportion of gaze points. The angular ranges which included 99.0%, 99.7% and 99.9% of all participants' gaze data in left, right, up and down directions were showed in Fig. 2 and Table 1.

[1] anonymous for review.

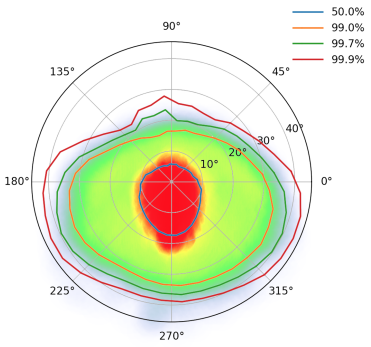

Figure 2: Heat map of the gaze points from all participants. The concentric circles indicated gaze angles θ. The contour indicated the range covering a certain percentage of gaze points.

This result inspired us to design a menu in the periphery of users' FOV. As most gaze points scattered around the center of FOV, the possibility that the user's gaze enters the menu unintentionally would be low. Meanwhile, previous work [26] has showed that humans can intentionally move their gaze to a larger range than in natural condition: 44° in left, right and down directions, and 28° in up direction. Therefore the peripheral menu can still be accessed by users using intentional gaze movement.

Table 1: The angular ranges of all/individual participants which covered 99.0%, 99.7% and 99.9% of gaze data in different directions. Minimum and maximum range was showed in parentheses.

		Left	Right	Up	Down
Collective	**99.0%**	33.4°	29.5°	16.5°	33.8°
	99.7%	37.3°	36.0°	23.5°	36.7°
	99.9%	41.9°	38.9°	29.7°	39.0°
Individual	**99.0%**	32.9±4.4° (26.4-40.1°)	27.1±4.7° (21.6-36.0°)	18.9±5.1° (12.7-28.1°)	31.5±3.6° (26.3-39.0°)
	99.7%	36.3±4.6° (29.4-43.4°)	34.7±5.9° (22.4-43.4°)	22.5±6.6° (13.8-36.7°)	34.4±3.4° (29.8-41.7°)
	99.9%	39.3±4.6° (31.6-47.2°)	36.0±6.5° (22.6-44.8°)	27.8±7.2° (13.8-37.7°)	37.0±3.2° (33.5-43.9°)

3.4.2 Individual Difference

We listed the minimum and maximum gaze ranges of individual participants in left, right, up and down directions in Table 1. The result showed a high diversity of gaze ranges of the participants in all directions. The angle of the participant with the largest gaze range was 50% larger than the angle of the participant with the smallest gaze range. This highlighted the necessity of designing personalized menu range that fits the gaze movement ability of different users.

4 GazeDock: Design and Implementation

In this section, we described the design and implementation of GazeDock, including interaction design, UI design and interaction algorithms.

4.1 Interaction Design

The interface of GazeDock was a ring-shaped menu located on the periphery of the user's FOV (see Fig. 10a). The outer boundary of GazeDock lied on the edge of FOV, and the inner boundary was determined according to the user's eye movement ability, which would be discussed later. This interface made it less occlusion in the user's central vision compared to a traditional pop-up menu (also saved screen occupation) [59]. Accordingly, menu items were arranged radially around the menu, in sectors.

GazeDock worked like the docking menu in desktop GUIs, which was hidden by default, and would be showed when the user's gaze approaches it. The user can then select a menu item by moving the gaze onto it. The selected item would be highlighted to inform the selection. When the user moves his/her gaze back from the menu, the last selected item would be triggered, and the menu would hide automatically. In practice, a "cancel" item can be arranged in the menu to quit the menu without triggering other commands, in case of false triggering. To provide a smooth experience of transition, we designed GazeDock to fade in (decrease transparency) and fade out (increase transparency) when showing and hiding, respectively.

4.2 Menu Boundary Personalization

A key problem of GazeDock was to determine the inner boundary of the menu that suits the gaze movement ability of the users. To explore this, we define *menu width* as the radial distance between the inner boundary and the outer boundary (the edge of FOV).

According to Fig. 2, the menu width in the up direction should be larger than that in left, right and down directions in order to fit the user's natural gaze range. Meanwhile, the gaze ranges were highly diverse across users (see Table 1). Therefore, if the menu width was too small, some users may not be able to reach the menu region; however if the width was too large, other users may frequently encounter false-triggering. Aiming at this problem, we decided to employ a personalized menu width for individual users.

As showed in Fig. 2, the distribution of users' gaze was similar to a flat ellipse. Therefore, we modeled the users' gaze range using an ellipse with user-specific parameters. Specifically, we used least-squares fitting [11] to find an optimal ellipse to fit the boundary of the user's gaze range including a certain proportion of gaze points. And as the gaze was horizontally symmetric, we constrained the rotation of the ellipse to zero:

$$\frac{(x-c_x)^2}{w^2}+\frac{(y-c_y)^2}{h^2}=1 \tag{2}$$

where (c_x, c_y) denoted the center of the ellipse, w and h denoted the width and height of the ellipse, respectively. This model requires only four parameters, therefore is easy to build by measuring the movement ranges in left, right, up and down directions. We built the models on the data from Study 1, Fig. 3 showed the fitted ellipses, and Table 2 showed the average parameters of ellipses of individual users.

This result confirmed that the X coordinates of fitted ellipse centers were close to zero (horizontally symmetric), therefore we constantly set $c_x \equiv 0$ for simplicity. Meanwhile, the other three parameters should be personalized for different users. According to user feedback, a "general" ellipse would be too large for some users, or would cause severe false triggering problems for other users. Therefore, we designed a calibration process to determine the menu boundary of GazeDock for each user:

Step 1: We arranged 11 spheres on the top middle and bottom middle of FOV, with gaze angles of $8^\circ - 28^\circ$ (up) and $20^\circ - 45^\circ$ (down) respectively (Fig. 4a). The user identified the highest/lowest spheres he/she could gaze at without discomfort or fatigue as the upper and lower bound, respectively. The identified sphere would be highlighted in red. h was calculated as the distance between the upper and lower bounds, and c_y was calculated as the midpoint of the them.

Step 2: Similar with Step 1, 22 spheres were arranged horizontally symmetrically with gaze angles of $20^\circ - 45^\circ$ (Fig. 4b). The user identified the left/right most spheres he/she could gaze at without discomfort or fatigue as the left and right bound, respectively. w was calculated as the distance between the left and right bounds.

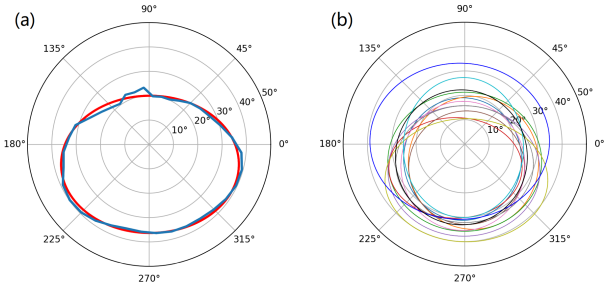

Figure 3: Fitted ellipses on the data in Study 1. (a) Fitted ellipse (red) and the actual boundary (blue) covering 99.7% of gaze points from all users. (b) Fitted ellipses for individual users covering 99.7% of gaze points.

Table 2: The parameters of fitted ellipses of individual participants covering 99.0%, 99.7% or 99.9% of gaze data.

	Center X	Center Y	Width	Height
99.0%	-0.68±1.88°	-6.91±3.69°	32.2±4.80°	23.7±2.57°
99.7%	-0.42±2.22°	-7.47±4.18°	35.7±5.12°	26.0±2.92°
99.9%	-0.10±2.25°	-8.65±4.06°	37.6±4.85°	27.6±3.18°

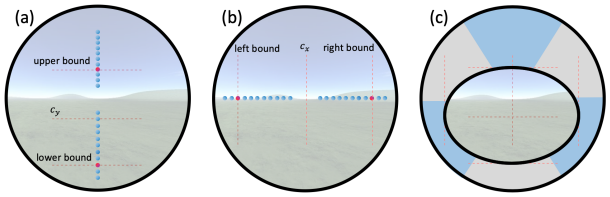

Figure 4: The calibration process. (a) Determine the upper and lower bounds. (b) Determine the left and right bounds. (c) The personalized boundary of GazeDock.

This calibration process generated an ellipse of the user's perceived "comfortable range" of gaze. According to pilot study, we further reduced the size (w and h) of the ellipse by multiplying a scaling factor of 0.9, which ensured comfortable interaction experience. The visual menu boundary of GazeDock was set according to the fitted ellipse (Fig. 4c). When using GazeDock, this process was performed only once for each user after the standard calibration process of the eye tracker, and took less than one minute.

4.3 Menu Triggering Algorithm

Ideally, we hope GazeDock to fade in when the user's gaze enters the menu region. However, Study 1 showed that although this works in most of the time, there are occasionally gazes that unintentionally

enters this region, which may cause false triggering. Therefore, besides gaze point location, we also extracted features from the moving process to help resolve input ambiguity.

Fig. 5 showed an example of gaze movements with large angles in natural movement and when invoking GazeDock, respectively. According to pilot study, intentional gaze pointing consisted of two phases (Fig. 5b): 1) Gaze angle increased rapidly; 2) Gaze angle kept for a period of time. In comparison, phase 2 did not exist in natural gaze movements (e.g. glance), since the user tended to rotate the head to help keep his/her gaze in the comfort zone.

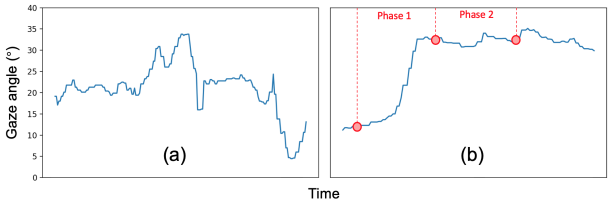

Figure 5: Examples of (a) natural gaze movement; and (b) intentionally approaching GazeDock. Red dots indicated the three key frames the algorithm monitored.

We designed the menu triggering algorithm of GazeDock according to this observation. Specifically, we used two connected time windows with size W to detect phase 1 and phase 2 respectively, yielding three key frames (red dots in Fig. 5): $t-2W$ (before gaze angle increases), $t-W$ (after gaze angle increases but before plateau) and t (after plateau). During usage, if the gaze angle at $t-2W$ was smaller than TH (a threshold), and gaze angles at $t-W$ and t were both greater than the inner boundary of GazeDock (details in the above section), we considered this process an intentional invoking behavior, and GazeDock would be showed automatically. We set $TH = 10^{\circ}$ and $W = 100ms$ according to pre-testing.

We validated the false-triggering-prevention performance of this algorithm on the data from Study 1. The personalized menu boundary for each participant was fitted using an ellipse on the 99% boundary. For each user, we took all his/her gaze data series as the input, and calculated the false triggering rate. In total, GazeDock was triggered 103 times during 399 minutes of natural gaze movement in VR applications, equivalent to a false triggering rate of 0.26 times/min. In comparison, a naïve algorithm that detects triggering action based on only menu boundary yielded a false triggering rate of 0.72 times/min. We also tried to further improve the algorithm by tuning parameters and involving more features (e.g., the variance of polar angles during triggering action), but the improvement was not significant.

4.4 Selection Confirmation Mechanism

We designed GazeDock to confirm the selection when the user's gaze left the menu from a menu item. This required the users to keep the gaze within the menu when browsing the items (i.e., curved trajectory). However, as the menu was ring-shaped, it was likely that the user's gaze may leave the menu unintentionally when going for items that were not adjacent.

To address this problem, we also applied an algorithm that shared the same principle with the menu triggering algorithm. Three key frames corresponding to another two phases ($W = 100ms$) were monitored: 1) Gaze angle decreased rapidly; 2) Gaze kept a small angle for a period of time. In addition, we set that if the user's gaze re-entered the menu within 200ms, the confirmation would not be triggered. Although resulting in a slight delay, this was very helpful for resolving unexpected triggers due to gaze jitter at large gaze angles.

5 Study 2: Gaze Pointing Performance on Peripheral Menus

So far, we have designed the menu layout and selection mechanism of GazeDock, which minimized the possibility of false triggering. In this section, we further investigated the users' gaze pointing ability within the peripheral menu of GazeDock, with the aim to facilitate the interaction performance using the optimized menu item layout. Specifically, we conducted two stages of experiments: 1) examining the users' selection precision in different directions; 2) examining the users' selection performance on GazeDock with different item densities.

5.1 Participants and Apparatus

10 participants in Study 1 (7 male, 3 female, aged 20-27) were recruited for this study. They had VR experience for 1.8 years on average, ranging from 0 to 4 years. None of them had experience in gaze-based menu selection. We used the same apparatus as in Study 1 for VR rendering and eye-tracking. The experiment platform was developed in Unity 2018.1.0b12, gaze data was obtained from the Unity interface of HTC SRanipal SDK.

5.2 Stage 1: Selecting Precision in Different Directions

In this stage, we asked the participants to gaze at precise directions on the periphery of FOV, and calculated the polar error. This result reflects the users' intrinsic gaze control ability for precise pointing.

5.2.1 Experiment Design and Procedure

Before the experiment, each participant first calibrated the eye tracker using the default calibration process of HTC Vive Pro Eye. They then completed the calibration process of GazeDock to generate their personalized menu boundaries. The experiment consisted of three sessions with 30 trials in each session. We evenly divided the polar angle of the FOV into 30 parts. In each trial, the participant was asked to gaze at an indicator in one part. The presented order of the indicators was randomized.

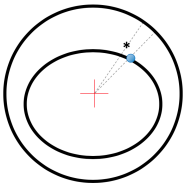

Figure 6: Experiment platform in stage 1. We measured the polar error as the angle between the indicator and the participant's actual gaze point (asterisk).

Fig. 6 showed the experiment platform. At the beginning of each trial, the participants were asked to keep their gaze at the center of the FOV (indicated by a red cross). They then moved the gaze to the target indicator (a blue circle with a visual angle less than 2°) on the menu's inner boundary) and kept for 500ms. After that, the participants were allowed to move the gaze back to the center and go to the next trial. A one-minute break was enforced between different sessions.

5.2.2 Results

We collected 10 (participants) × 3 (sessions) × 30 (trials) = 900 trials in total. We removed two obviously wrong trials and used 898 trials for analysis. We averaged the gaze points during the keeping phase (i.e., the velocity of gaze below a threshold) as the participant's actual gaze point in a trial.

We measured the precision of gaze in a trial using **polar error**, which was calculated as the polar angle between the participant's

actual gaze point and the indicator. A positive polar error indicated that the actual gaze lied to the left of the target (in polar system), and vice versa. We also measured the **absolute polar error** to analyze the amplitude of polar errors, which corresponded to the absolute value of polar error.

Fig. 7a showed the distribution of polar error across all targets, which roughly followed a Gaussian distribution. The mean polar error was 0.02° (SD = 5.42°). The 90%, 95% and 99% confident interval of polar error were −7.21°–7.98°, -11.3°–11.4°, and -22.8°–17.5°, respectively. This result could been seen as the upper bound of the gaze pointing precision on a peripheral menu, and would determine the upper bound of the number of items on the menu. For example, if we assume 18 items on the menu, then each item would be 20° on average. Therefore, about 5% of gaze points would fail to hit the target correctly. In practice, researchers could also choose the corresponding number of items given the acceptable error rate based on the result.

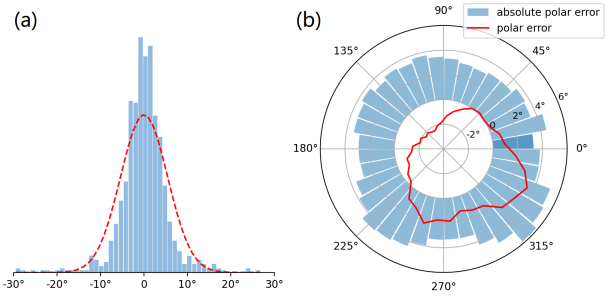

Figure 7: (a) The distribution of polar error. The red curve showed a fitted Gaussian distribution. (b) Polar error and absolute polar error in different directions.

We also visualized the average polar error and absolute polar error in different directions (Fig. 7b). Polar error showed tendencies in different directions: the participant's actual gaze tended to yield a negative deviation when gazing at the upper-left part and a positive deviation when gazing at the lower-right part. Meanwhile, absolute polar error was not observably different in different directions. The average absolute polar error in all directions ranged from 2.84° to 5.84°, while the standard deviations even reached between 3.35° and 7.14°. As the pointing accuracy in real use was mainly affected by absolute polar error, we concluded that the menu item arrangement of GazeDock could be uniform across different directions.

5.3 Stage 2: Comparing Different Item Densities

In this stage, we conducted another experiment that required the participants to select from different numbers of menu items on a peripheral menu. This not only complemented the above results with results on interaction speed, but also verified the results by testing targets with different angle sizes.

5.3.1 Experiment Design and Procedure

In this stage, we implemented the menu triggering and selection confirming algorithms as described above to provide the real interaction experience of GazeDock. We used a within-subjects experiment design with only one factor *Density* (number of menu items). Specifically, according to the results in stage 1, we tested five densities: 4, 6, 8, 12 and 16 menu items (see Fig. 8).

Before the experiment, we employed the same calibration process for the eye tracker and menu boundaries as in stage 1. After that, the participants spent 2–3 minutes to familiarize themselves with the interaction of GazeDock. They then completed five sessions of selection tasks in random order, each corresponding to one density.

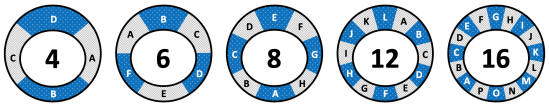

Figure 8: An example of GazeDock with different numbers of menu items. Letters labelled menu items clockwise, while "A" located randomly in different sessions.

In each session, the menu showed the letters 'A', 'B', ... clockwise on each menu item, with the start position (i.e. 'A') randomly chosen (see Fig. 8). Each session contained 24 trials. In each trial, a random target letter would appear in the center of FOV, the participant was required to select the corresponding menu item using GazeDock as quickly and as accurately as possible. A one-minute break was enforced between different sessions.

5.3.2 Results

The average selection error rate was 0.8% (SD = 1.7%), 2.9% (SD = 3.4%), 5.8% (SD = 4.0%), 12.5% (SD = 8.3%) and 20.8% (SD = 5.9%) for increasing number of menu items respectively. RM-ANOVA found a significant effect of *Density* on error rate ($F_{4,36} = 28.8$, $p < .001$). As expected, error rate increased monotonously with item density. Linear fitting between density and error rate yielded an R^2 of 0.99. For density ≤ 8, the error rate was relatively low (< 6%), but it dramatically increased to over 12% for 12 items, and even over 20% for 16 items. Therefore, we considered the maximum number of items to be 8 in practical use.

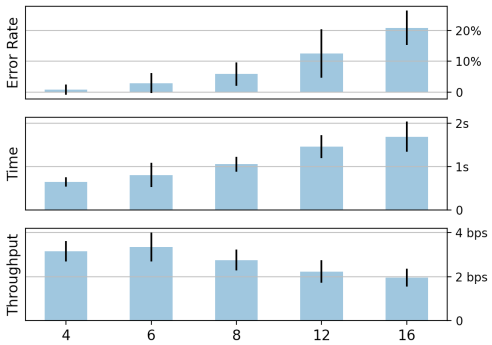

Figure 9: The error rate, selection time and throughput for different densities. Error bar indicated one standard deviation.

Selection time was defined as the elapse between the moment the user's gaze starts moving and the confirming of the selection. The average selection time was 0.64s (SD = 0.11), 0.80s (SD = 0.29), 1.05s (SD = 0.18), 1.46s (SD = 0.28) and 1.69s (SD = 0.37) for increasing item densities respectively. RM-ANOVA found a significant effect of *Density* on selection time ($F_{4,36} = 52.2$, $p < .001$). Linear fitting between density and selection time yielded an R^2 of 0.98. We speculated that browsing the menu items affected the selection time more than triggering the menu.

We also measured the throughput [45] to analyze the tradeoff between speed and accuracy, which was calculated as:

$$\text{throughput} = \frac{\text{Number of items}}{\text{Selection time}} \times (1 - \text{error rate}) \qquad (3)$$

The average throughput was 3.16bps (SD = 0.48), 3.35bps (SD = 0.69), 2.76bps (SD = 0.49), 2.24bps (SD = 0.54), and 1.95bps (SD = 0.43) for increasing densities, respectively. RM-ANOVA found a significant effect of *Density* on throughput ($F_{4,36} = 31.4$,

$p < .001$). Post-hoc analysis with Bonferroni correction found no significant difference between 4, 6 and 8 items. This results implied that arranging 4–8 items on the peripheral menu would yield the highest performance.

6 Study 3: Usability Evaluation in Real Tasks

So far, we have examined the false-triggering-prevension ability of GazeDock through simulation (see Section 4.3), and user's menu item selection performance (see Section 5). We now evaluated the usability of GazeDock in a VR game scenario. Compared with tasks only in constrained conditions (e.g., [3,4]), this also involved the switch between gaze interaction and other tasks (e.g., natural gaze movement during scene exploration), which required a balance between input speed and robustness. Therefore, the results were more indicative of the usability of gaze-only techniques in real scenarios.

6.1 Participants and Apparatus

We recruited 8 participants (6 male, 2 female, aged 21-28) from the campus. None of them have participated in the previous studies. The same apparatus as in Study 1 and 2 was used in this experiment. The VR game was developed in Unity 2018.1.0b12.

6.2 Experiment Platform

We developed a gaze-based VR game as the experiment platform (see Fig. 10). We designed three rooms in the scene: laboratory, magic house and bedroom, and we placed a number of props that belong to different rooms (e.g., laboratory: flask, test tubes and tweezers; magic house: candle, totem and poison; bedroom: football, speaker and potted plants) in random places. The goal of this game was to explore the rooms, find the props that do not belong to the current room, and move them back to their target positions (highlighted by a halo). The supported interactions in this game were:

- *Explore*: move the user's head and body to observe the props in the current room.
- *Navigate*: invoke one of the three 'Move" commands ("Move to Lab", "Move to Magic House" or "Move to Bedroom") to teleport between different rooms.
- *Pick*: face a prop and invoke "Pick" command to pick it up.
- *Drop*: face a potential position of the prop and invoke "Drop" command to put it there. If the prop matches the position, the prop will be successfully dropped. Otherwise, the user has to look for another position to drop.

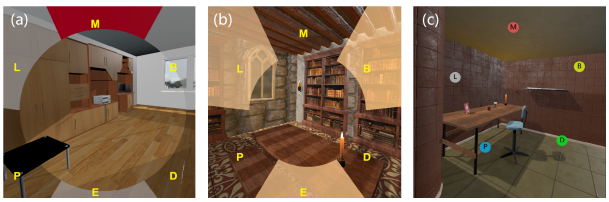

Figure 10: Experiment platform with different techniques: (a) GazeDock; (b) Dwelling; (c) Pursuit.

6.3 Techniques

The experiment platform involved both scene exploration (with natural gaze movement) and command invoking. However, most gaze-only interaction techniques (e.g. [3]) did not consider how to "enable" and "disable" themselves, making them incapable of the compound task. Therefore, we chose *Dwelling* and *Pursuit*, the two most widely-adopted gaze-only interaction paradigms as the baseline techniques for menu interaction, and used similar designs as GazeDock for a fair comparison. We arranged 5 menu items for each technique, corresponding to the 5 commands above. We also added a 6th command "Cancel" for GazeDock and dwelling, in case of false triggering. The labels on each item were represented by a character in the corresponding command. The details of the techniques were:

GazeDock: We used a 6-item GazeDock with the algorithms and parameters described in Section 4 (see Fig. 10a).

Dwelling: Dwelling-based techniques use a menu that is always displayed, and triggers an item when the gaze dwells on the menu item. As menus in the central of the user's FOV (e.g. [24, 54]) could not be used in our scenario due to severe occlusion, we used a translucent peripheral menu referring to previous works [20, 22, 32]. We tested the menu boundary of GazeDock for dwelling, but did not achieved acceptable performance as dwelling on the edge of FOV was much more difficult than glancing. Therefore, for each participant, we expanded the width of the peripheral menu as they desired (see Fig. 10b). The dwelling threshold was set to 500ms according to pilot study, which was also a typical value for stable dwelling performance [33].

Pursuit: Pursuit-based techniques require the users to perform gaze gestures following the unique trajectories of different objects or items [10, 55] to select them. We designed five round targets with different colors and labels for the commands. To avoid occlusion, we arranged the targets to move along a circle in the periphery of FOV following Orbits [9] (see Fig. 10c) and used the same parameters (e.g., window size: 1000ms, moving speed: 120°/s, correlation threshold: 0.8) to maintain optimal performance. We also verified the usability of the parameter values through pilot study.

We did not test gaze gesture techniques as they required careful designing of the gesture strokes and gesture-command mapping to achieve good performance while preventing false triggering [5], which is out of the scope of this paper. Also, menu selection techniques (e.g. GazeDock) can be easily generalized to increasing number of items (e.g. by using hierarchical menu [24]), which was usually not the design goal of gaze gesture techniques.

6.4 Procedure

Before the experiment, each participant first completed the calibration process of the eye tracker and GazeDock respectively. They then spent 5–10 minutes to familiarize themselves with the interactions of the experiment platform and the three techniques.

The experiment was consisted of three sessions, each corresponding to one technique, in random order. In each session, the participants were required to place six random props to the correct positions. As the goal of this experiment was to mimic the most natural and realistic use scenario, we did not restricted the time or order of the tasks, the participants were allowed to explore and complete the tasks freely. Accordingly, they were asked to report all the errors during the experiment for analysis (e.g., false triggering of a command or failed to trigger a command). The experiment was finished after the participants have placed all the six props at the correct positions. And the participants filled out questionnaires to give their subjective ratings on the different techniques.

6.5 Results

6.5.1 Interaction Performance

As the difficulty of the random tasks varied significantly (some may took longer time to explore), and the participants were free to perform different actions, we did not analyze the total time they spent. Focusing on the interaction performance, we measured the **selection time** of GazeDock as in Study 2. The average selection time was 471ms (SD = 142ms), which was faster than Dwelling ($\geq 500ms$)

and Pursuit ($\geq 1000ms$). Note that we allowed the users to complete the tasks freely. Therefore the results were only indicative of the relative performance of different techniques.

It is noteworthy that the selection speed of GazeDock in Study 3 was faster than that in Study 2 (see Fig. 9). We observed that as the menu layout in Study 3 was fixed, the participants could get familiar with the location of different items quickly with practice, and could learn to trigger the items with little or no aiming. This suggested that GazeDock could facilitate a novice-to-expert transition, and achieve potentially higher interaction speed when used "eyes-freely".

We also measured the robustness of the different techniques. As the ground truth of user's interaction intention was not obtainable, we calculated the user-report **false positive** (false triggering of a command) and **false negative** (tried to invoke a command but failed) rate, respectively. These metrics were not affected by the frequency of the user's interaction, thus is more suitable for the free interaction scenario in this study.

The false positive rates of GazeDock, Dwelling and Pursuit were 3.6% (SD = 4.2%), 2.8% (SD = 3.7%) and 4.9% (SD = 4.9%), respectively. RM-ANOVA found no significant difference between different techniques ($F_{2,14} = 0.436$, $p = .65$). Meanwhile, the false negative rates of GazeDock, Dwelling and Pursuit were 1.9% (SD = 2.6%), 2.2% (SD = 2.6%) and 2.4% (SD = 2.2%), respectively. Still, no significant difference between different techniques was found ($F_{2,14} = 0.056$, $p = .95$).

Combining the results, we could found that the three techniques achieved competitive performance when preventing false triggering and responding to user's interactions. Therefore, they were all successful in distinguishing intentional and unintentional gaze interactions. Meanwhile, GazeDock yielded potentially higher interaction speed than the other two techniques. As the parameters of all the three techniques have been fine-tuned for our scenario, we believe these results implied that GazeDock could effectively balance input speed and robustness, and could achieve satisfying performance in realistic compound tasks. Of course, in real use, the performance of all the three techniques can still be adjusted for specific demands by changing the parameter values. However, increasing the performance in one aspect (e.g. shorter time threshold for higher speed) would sacrifice other metrics (e.g., more false triggering), given the speed-robustness trade-off.

6.5.2 Subjective Feedback

Subjective ratings were a very important metric for our semi-structured experiment, as we emphasize more on a "realistic user experience" rather than "optimal performance". The subjective ratings were gathered through a 7-point Likert-scale questionnaire. Dimensions include perceived efficiency ("Are you satisfied with the selection speed of this technique?"), perceived robustness ("Are you satisfied with the false triggering prevention ability of this technique?"), ease of learning ("Do you agree that this technique is easy to learn?"), effortless ("Do you feel tired when using this technique?"), presentation ("Are you satisfied with the appearance and behavior of the UI of this technique?") and overall preference. Fig. 11 showed the subjective ratings of the techniques.

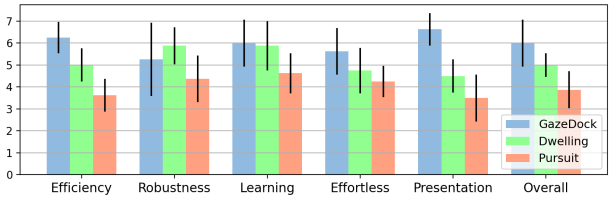

Figure 11: Subjective ratings of different techniques (7: most positive, 1: most negative). Error bars indicate one standard deviation.

GazeDock received a significantly higher rating than the other two techniques on all dimensions except for perceived robustness (*Efficiency*: $\chi^2(2) = 15.0$, $p < .001$, *Learning*: $\chi^2(2) = 13.2$, $p < .001$, *Effortless*: $\chi^2(2) = 6.50$, $p < .05$, *Presentation*: $\chi^2(2) = 13.3$, $p < .001$ and *Overall*: $\chi^2(2) = 8.87$, $p < .05$. And although the *Robustness* score of GazeDock was lower than that of Dwelling, no significant difference was found ($\chi^2(2) = 4.22$, $p = .121$), and the score of all techniques were above 4.3, indicating that the participants were positive towards all the techniques. We did not formally evaluate the metrics that would also be affected by the scene design (e.g., ocular discomfort and cybersickness), but during interview, the participants were generally satisfied with the user experience, and did not report significant discomfort.

7 Discussion

7.1 Feasibility of GazeDock

GazeDock enables fast and robust gaze-only menu input. The design goal of GazeDock was to leverage gaze movement patterns that distinguished from natural gaze movements to prevent false triggering. However, the demand for interaction speed makes it difficult to design such "abnormal" gaze patterns (e.g., long and complex gaze gestures were not acceptable). Therefore, in Study 1, we first collected the user's natural gaze movement data in typical kinds of VR applications, and analyzed the movement patterns. We found that gaze points mainly distributed in the central area of FOV, and rarely reached the peripheral zone.

Based on the above findings, we proposed the peripheral menu design of GazeDock. Compared with conventional rectangle menus, this not only reduced the conflict with natural gaze movement, but also introduced an advantage of less screen occupation. We also designed menu triggering and selection confirming algorithms that considered both gaze location and gaze movement speed to further prevent false triggering from glancing. Simulation results showed a false triggering rate of only 0.26 times/min during natural gaze movement. The design of peripheral menu not only provided opportunities for preventing false triggering, but also improved the input speed by omitting explicit menu triggering and confirming actions. This was only achievable as GazeDock was able to distinguish between intentional and unintentional gaze movements.

To further verify the input speed and usability of GazeDock, we conducted Study 2 and Study 3 in controlled and free conditions, respectively. Results showed that GazeDock could achieve a selection speed of 1.05s with 5.8% error rate for 8 items. This suggested that although in the peripheral zone of FOV, the users could still accurately point at targets on the menu. And in compound tasks, GazeDock were more preferred by the participants compared with Dwelling and Pursuit. These results proved that GazeDock could effectively support menu selection without disturbing natural gaze movements, and could achieve satisfying performance and user experience in real use scenarios.

7.2 GazeDock vs. Gaze Gesture Techniques

GazeDock was designed as a menu technique. However, in Study 3, we observed that participants could enter an "expert mode" when they get familiar with the menu layout: the users could memorize the direction of the target item, and can perform the selection without adjusting within the menu, similar as a "gaze crossing gesture". There are two intrinsic difference between GazeDock and other gaze gesture techniques:

First, menu layout was a necessary component of any menu techniques including GazeDock, as memorizing all the menu items was not easy. However, typical gaze gesture techniques (e.g., [5]) do not involve visible guides as they feature eyes-free usage. Accordingly, GazeDock were more flexible when handling arbitrary number of items (e.g., by appending items on an "dynamic inner menu", GazeDock could support hierarchical menus). In comparison, gaze

gesture techniques were more specific to a fixed number of gesture set with specially designed gesture-command mappings.

Second, gaze gesture techniques require the users to perform the correct gesture to trigger the corresponding commands. In comparison, GazeDock always triggers the last selected item before confirmation. Therefore, GazeDock does not require a 1:1 mapping between the gaze trajectory and the target item, as users could adjust the selected item within the menu before confirmation, which compensated the imprecision in the initial gaze movement.

GazeDock has an intrinsic advantage over saccades: novice users could use GazeDock with the help of the visual menu layout, thus minimizing the memorization and learning effort, achieving a satisfying pick-up speed ($< 1.5s$ for 12 items as in Fig. 9). And with practice, experienced users could estimate the direction of the target item and perform a "forth-back" gesture for selection, therefore achieving much higher interaction performance (471ms in Study 3) without adjusting the algorithms. Also, with the help of the visual guidance, selecting items with GazeDock would be more accurate than simply glancing at different directions eyes-freely. Although we only tested 6 items in Study 3 to evaluate the usability of GazeDock, the results in Study 2 suggested that supporting more items was also achievable. And in real use, both the pick-up usability and the generalizability of a technique were very important. Therefore, the visual menu played a necessary part in the usability of GazeDock.

7.3 Personalization

As we showed in Study 1, the size and location of the comfort zone of different users varied significantly (see Table 1), which highlighted the necessity of personalization for GazeDock. In this paper, we designed a calibration process to measure the comfort zone of different users, which was effective, but at the cost of more preparation time. Further improvements of personalization include: 1) combining the calibration process of GazeDock with the calibration process of the eye tracker by containing the peripheral of FOV in the moving trajectory of the argets [43]; 2) introducing a dynamic algorithm for GazeDock that starts with a default menu boundary, and dynamically adjusts the boundary according to the user's gaze movement patterns. These solutions were expected to further improve the user experience of GazeDock, but the interaction performance we examined in this paper would not be affected. Therefore, we defer them to future work.

8 Conclusion

In this paper, we proposed GazeDock, a fast and robust gaze-only menu technique in VR that leveraged auto-triggering peripheral menu. We iteratively conducted three user studies to facilitate the design and evaluation of GazeDock. Study 1 examined the user's natural gaze movement pattern in typical VR applications, which verified the peripheral menu design of GazeDock. Study 2 examined user's menu selection performance with GazeDock, and found that 4–8 items would yield optimal performance. Study 3 evaluated the usability of GazeDock in compound tasks, and found that GazeDock were more preferred by the participants compared with Dwelling and Pursuit techniques. We concluded that GazeDock could effectively support gaze-only menu selection in VR while preventing false triggering, making it a potential solution for real-world VR applications.

Acknowledgements

This work was supported by the Natural Science Foundation of China (NSFC) under Grant No. 61902208, No.6213000120, No.62002198, and the grant from the Institute for Guo Qiang, Tsinghua University No. 2019GOG0003.

References

[1] Vive pro eye. Website, 2019.

[2] J. B. Badler and A. Canossa. Anticipatory gaze shifts during navigation in a naturalistic virtual environment. In *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*, CHI PLAY '15, pp. 277–283. ACM, New York, NY, USA, 2015. doi: 10.1145/2793107.2793136

[3] N. Bee and E. André. Writing with your eye: A dwell time free writing system adapted to the nature of human eye gaze. In E. André, L. Dybkjær, W. Minker, H. Neumann, R. Pieraccini, and M. Weber, eds., *Perception in Multimodal Dialogue Systems*, pp. 111–122. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[4] N. Cournia, J. D. Smith, and A. T. Duchowski. Gaze- vs. hand-based pointing in virtual environments. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03, pp. 772–773. Association for Computing Machinery, New York, NY, USA, 2003. doi: 10.1145/765891.765982

[5] H. Drewes and A. Schmidt. Interacting with the computer using gaze gestures. In *IFIP Conference on Human-Computer Interaction*, pp. 475–488. Springer, 2007.

[6] M. L. Dybdal, J. S. Agustin, and J. P. Hansen. Gaze input for mobile devices by dwell and gestures. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, pp. 225–228. Association for Computing Machinery, New York, NY, USA, 2012. doi: 10.1145/2168556.2168601

[7] C. Elmadjian and C. H. Morimoto. *GazeBar: Exploiting the Midas Touch in Gaze Interaction*. Association for Computing Machinery, New York, NY, USA, 2021.

[8] A. Esteves, Y. Shin, and I. Oakley. Comparing selection mechanisms for gaze input techniques in head-mounted displays. *International Journal of Human-Computer Studies*, 139:102414, 2020. doi: 10.1016/j.ijhcs.2020.102414

[9] A. Esteves, E. Velloso, A. Bulling, and H. Gellersen. Orbits: Gaze interaction for smart watches using smooth pursuit eye movements. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15, pp. 457–466. Association for Computing Machinery, New York, NY, USA, 2015. doi: 10.1145/2807442.2807499

[10] A. Esteves, D. Verweij, L. Suraiya, R. Islam, Y. Lee, and I. Oakley. Smoothmoves: Smooth pursuits head movements for augmented reality. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, pp. 167–178. Association for Computing Machinery, New York, NY, USA, 2017. doi: 10.1145/3126594.3126616

[11] A. Fitzgibbon, M. Pilu, and R. B. Fisher. Direct least square fitting of ellipses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):476–480, May 1999. doi: 10.1109/34.765658

[12] E. G. Freedman. Coordination of the eyes and head during visual orienting. *Experimental Brain Research*, 190(4):369, Aug 2008. doi: 10.1007/s00221-008-1504-8

[13] E. G. Freedman and D. L. Sparks. Eye-head coordination during head-unrestrained gaze shifts in rhesus monkeys. *Journal of neurophysiology*, 77(5):2328–2348, 1997.

[14] Google. Blocks. Website, July 2017.

[15] A. Gravity. Rec room. Website, June 2016.

[16] J. P. Hansen, H. Lund, F. Biermann, E. Møllenbach, S. Sztuk, and J. S. Agustin. Wrist-worn pervasive gaze interaction. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, ETRA '16, pp. 57–64. Association for Computing Machinery, New York, NY, USA, 2016. doi: 10.1145/2857491.2857514

[17] H. Heikkilä and K.-J. Räihä. Simple gaze gestures and the closure of the eyes as an interaction technique. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, pp. 147–154. Association for Computing Machinery, New York, NY, USA, 2012. doi: 10.1145/2168556.2168579

[18] D. S. Inc. Drop. Website, February 2018.

[19] Innoarea. Island journey. Website, October 2017.

[20] H. Istance, R. Bates, A. Hyrskykari, and S. Vickers. Snap clutch, a moded approach to solving the midas touch problem. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, pp. 221–228. ACM, 2008.

[21] H. Istance, A. Hyrskykari, L. Immonen, S. Mansikkamaa, and S. Vick-

ers. Designing gaze gestures for gaming: An investigation of performance. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, ETRA '10, pp. 323–330. Association for Computing Machinery, New York, NY, USA, 2010. doi: 10.1145/1743666.1743740

[22] H. Istance, A. Hyrskykari, S. Vickers, and T. Chaves. For your eyes only: Controlling 3d online games by eye-gaze. In T. Gross, J. Gulliksen, P. Kotzé, L. Oestreicher, P. Palanque, R. O. Prates, and M. Winckler, eds., *Human-Computer Interaction – INTERACT 2009*, pp. 314–327. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[23] R. J. K. Jacob. What you look at is what you get: Eye movement-based interaction techniques. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '90, pp. 11–18. ACM, New York, NY, USA, 1990. doi: 10.1145/97243.97246

[24] Y. Kammerer, K. Scheiter, and W. Beinhauer. Looking my way through the menu: The impact of menu design and multimodal input on gaze-based menu selection. In *Proceedings of the 2008 Symposium on Eye Tracking Research & Applications*, ETRA '08, pp. 213–220. ACM, New York, NY, USA, 2008. doi: 10.1145/1344471.1344522

[25] M. Khamis, C. Oechsner, F. Alt, and A. Bulling. Vrpursuits: interaction in virtual reality using smooth pursuit eye movements. In *Proceedings of the 2018 International Conference on Advanced Visual Interfaces*, p. 18. ACM, 2018.

[26] J. H. Kim and H. W. Lim. Range of eye movement in a normal population and its relationship to age. *J Korean Ophthalmol Soc*, 58(6):698–705, June 2017. doi: 10.3341/jkos.2017.58.6.698

[27] D. Kirst and A. Bulling. On the verge: Voluntary convergences for accurate and precise timing of gaze input. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, pp. 1519–1525. Association for Computing Machinery, New York, NY, USA, 2016. doi: 10.1145/2851581.2892307

[28] S. Kudo, H. Okabe, T. Hachisu, M. Sato, S. Fukushima, and H. Kajimoto. Input method using divergence eye movement. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pp. 1335–1340. Association for Computing Machinery, New York, NY, USA, 2013. doi: 10.1145/2468356.2468594

[29] M. Kumar, A. Paepcke, T. Winograd, and T. Winograd. Eyepoint: practical pointing and selection using gaze and keyboard. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 421–430. ACM, 2007.

[30] M. Kytö, B. Ens, T. Piumsomboon, G. A. Lee, and M. Billinghurst. Pinpointing: Precise head-and eye-based target selection for augmented reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, p. 81. ACM, 2018.

[31] J. J. LaViola, Jr., D. A. Feliz, D. F. Keefe, and R. C. Zeleznik. Hands-free multi-scale navigation in virtual environments. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, I3D '01, pp. 9–15. ACM, New York, NY, USA, 2001. doi: 10.1145/364338.364339

[32] C. Lutteroth, M. Penkar, and G. Weber. Gaze vs. mouse: A fast and accurate gaze-only click alternative. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15, pp. 385–394. Association for Computing Machinery, New York, NY, USA, 2015. doi: 10.1145/2807442.2807461

[33] P. Majaranta, A. Aula, and K.-J. Räihä. Effects of feedback on eye typing with a short dwell time. In *Proceedings of the 2004 Symposium on Eye Tracking Research & Applications*, ETRA '04, pp. 139–146. ACM, New York, NY, USA, 2004. doi: 10.1145/968363.968390

[34] D. Mardanbegi, D. W. Hansen, and T. Pederson. Eye-based head gestures. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, pp. 139–146. ACM, New York, NY, USA, 2012. doi: 10.1145/2168556.2168578

[35] D. Mardanbegi, T. Langlotz, and H. Gellersen. Resolving target ambiguity in 3d gaze interaction through vor depth estimation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pp. 612:1–612:12. ACM, New York, NY, USA, 2019. doi: 10.1145/3290605.3300842

[36] D. Mardanbegi, B. Mayer, K. Pfeuffer, S. Jalaliniya, H. Gellersen, and A. Perzl. Eyeseethrough: Unifying tool selection and application in virtual environments. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pp. 474–483, 2019.

[37] P. Mitchell and B. Wilkinson. Periphery triggered menus for head mounted menu interface interactions. In *Proceedings of the 28th Australian Conference on Computer-Human Interaction*, OzCHI '16, pp. 30–33. ACM, New York, NY, USA, 2016. doi: 10.1145/3010915.3010964

[38] E. Møllenbach, M. Lillholm, A. Gail, and J. P. Hansen. Single gaze gestures. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, ETRA '10, pp. 177–180. Association for Computing Machinery, New York, NY, USA, 2010. doi: 10.1145/1743666.1743710

[39] T. Nukarinen, J. Kangas, O. Špakov, P. Isokoski, D. Akkil, J. Rantala, and R. Raisamo. Evaluation of headturn: An interaction technique using the gaze and head turns. In *Proceedings of the 9th Nordic Conference on Human-Computer Interaction*, p. 43. ACM, 2016.

[40] K. Pfeil, E. M. Taranta, A. Kulshreshth, P. Wisniewski, and J. J. LaViola. A comparison of eye-head coordination between virtual and physical realities. In *Proceedings of the 15th ACM Symposium on Applied Perception*, SAP '18. Association for Computing Machinery, New York, NY, USA, 2018. doi: 10.1145/3225153.3225157

[41] K. Pfeuffer, J. Alexander, M. K. Chong, and H. Gellersen. Gaze-touch: Combining gaze with multi-touch for interaction on the same surface. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pp. 509–518. ACM, New York, NY, USA, 2014. doi: 10.1145/2642918.2647397

[42] K. Pfeuffer, B. Mayer, D. Mardanbegi, and H. Gellersen. Gaze+ pinch interaction in virtual reality. In *Proceedings of the 5th Symposium on Spatial User Interaction*, pp. 99–108. ACM, 2017.

[43] K. Pfeuffer, M. Vidal, J. Turner, A. Bulling, and H. Gellersen. Pursuit calibration: Making gaze calibration less tedious and more flexible. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pp. 261–270. Association for Computing Machinery, New York, NY, USA, 2013. doi: 10.1145/2501988.2501998

[44] T. Piumsomboon, G. Lee, R. W. Lindeman, and M. Billinghurst. Exploring natural eye-gaze-based interaction for immersive virtual reality. In *2017 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 36–39, 2017.

[45] Y. Y. Qian and R. J. Teather. The eyes don't have it: an empirical comparison of head-based and eye-based selection in virtual reality. In *Proceedings of the 5th Symposium on Spatial User Interaction*, pp. 91–98. ACM, 2017.

[46] S. Schenk, M. Dreiser, G. Rigoll, and M. Dorr. Gazeeverywhere: Enabling gaze-only user interaction on an unmodified desktop pc in everyday scenarios. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pp. 3034–3044. Association for Computing Machinery, New York, NY, USA, 2017. doi: 10.1145/3025453.3025455

[47] L. Sidenmark and H. Gellersen. Eye, head and torso coordination during gaze shifts in virtual reality. *ACM Trans. Comput.-Hum. Interact.*, 27(1), Dec. 2019. doi: 10.1145/3361218

[48] L. Sidenmark and H. Gellersen. Eye&head: Synergetic eye and head movement for gaze pointing and selection. In *Proceedings of the 32Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, pp. 1161–1174. ACM, New York, NY, USA, 2019. doi: 10.1145/3332165.3347921

[49] V. Sitzmann, A. Serrano, A. Pavel, M. Agrawala, D. Gutierrez, B. Masia, and G. Wetzstein. Saliency in vr: How do people explore virtual environments? *IEEE Transactions on Visualization and Computer Graphics*, 24(4):1633–1642, 2018.

[50] O. Špakov and P. Majaranta. Enhanced gaze interaction using simple head gestures. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pp. 705–710. ACM, 2012.

[51] S. Stellmach and R. Dachselt. 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, pp. 285–294. Association for Computing Machinery, New York, NY, USA, 2013. doi: 10.1145/2470654.2470695

[52] S. Tregillus, M. Al Zayer, and E. Folmer. Handsfree omnidirectional vr navigation using head tilt. In *Proceedings of the 2017 CHI Conference*

on Human Factors in Computing Systems, CHI '17, pp. 4063–4068. ACM, New York, NY, USA, 2017. doi: 10.1145/3025453.3025521

[53] S. Tregillus and E. Folmer. Vr-step: Walking-in-place using inertial sensing for hands free navigation in mobile vr environments. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 1250–1255. ACM, 2016.

[54] M. H. Urbina, M. Lorenz, and A. Huckauf. Pies with eyes: The limits of hierarchical pie menus in gaze control. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, ETRA '10, pp. 93–96. ACM, New York, NY, USA, 2010. doi: 10.1145/1743666.1743689

[55] M. Vidal, K. Pfeuffer, A. Bulling, and H. W. Gellersen. Pursuits: eye-based interaction with moving targets. In *CHI '13 Extended Abstracts on Human Factors in Computing Systems*, pp. 3147–3150. ACM, 2013.

[56] O. Špakov, P. Isokoski, and P. Majaranta. Look and lean: Accurate head-assisted eye pointing. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '14, pp. 35–42. ACM, New York, NY, USA, 2014. doi: 10.1145/2578153.2578157

[57] O. Špakov and D. Miniotas. Gaze-based selection of standard-size menu items. In *Proceedings of the 7th International Conference on Multimodal Interfaces*, ICMI '05, pp. 124–128. ACM, New York, NY, USA, 2005. doi: 10.1145/1088463.1088486

[58] Y. Yan, C. Yu, X. Yi, and Y. Shi. Headgesture: Hands-free input approach leveraging head movements for hmd devices. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(4):198:1–198:23, Dec. 2018. doi: 10.1145/3287076

[59] X. Yi, C. Yu, W. Xu, X. Bi, and Y. Shi. Compass: Rotational keyboard on non-touch smartwatches. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pp. 705–715. Association for Computing Machinery, New York, NY, USA, 2017. doi: 10.1145/3025453.3025454

[60] S. Zhai, C. Morimoto, and S. Ihde. Manual and gaze input cascaded (magic) pointing. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 246–253. ACM, 1999.

[61] Y. Zhang, A. Bulling, and H. Gellersen. Sideways: A gaze interface for spontaneous interaction with situated displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pp. 851–860. Association for Computing Machinery, New York, NY, USA, 2013. doi: 10.1145/2470654.2470775

[62] Y. Zhao, E. Cutrell, C. Holz, M. R. Morris, E. Ofek, and A. D. Wilson. Seeingvr: A set of tools to make virtual reality more accessible to people with low vision. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19. Association for Computing Machinery, New York, NY, USA, 2019. doi: 10.1145/3290605.3300341