國立臺灣大學管理學院資訊管理學系

碩士論文

Department of Information Management
College of Management
National Taiwan University

Master Thesis

利用漸進式視覺引導以完成連續型視線輸入

GazeBeacon: Enabling Smooth Pursuit Gaze Gestures by Gradual Visual Guidance

楊立銘
Li-Ming Yang

指導教授：陳炳宇 博士
Advisor: Bing-Yu Chen, Ph.D.

中華民國 105 年 6 月
June, 2016

doi:10.6342/NTU201700648

國立臺灣大學碩士學位論文
口試委員會審定書

利用漸進式視覺引導以完成連續型視線輸入

GazeBeacon: Enabling Smooth Pursuit Gaze Gestures by Gradual Visual Guidance

本論文係楊立銘君（學號 R03725028）在國立臺灣大學資訊管理學系、所完成之碩士學位論文，於民國 105 年 6 月 16 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

所　　長：

doi:10.6342/NTU201700648

致謝

謝謝陳炳宇老師，讓我有機會進到這個有趣的實驗室，鼓勵我們自己找感興趣的題目，在我進度落後時不放棄敦促我，讓我週週都有名言金句，您眞是最沒有架子的教授。謝謝梁容豪學長、詹力韋學長，和你們討論後總會有很實用的收穫，雖然總無可避免會加深自己眞是做出一堆垃圾的念頭，但你們眞是實驗室成員們在研究路上最好的榜樣。謝謝大源學長的指導，咪挺報題目報到心灰意冷時一句「我覺得這個滿有趣的」成爲帶三個小孩好爸爸重擔的開始。你說一次三個弄不過來有點抱歉，但我對你只有無限的感激。如果不是你，我大概現在還在報題目。

謝謝畢業小夥伴們順堯、曉楓、龍飛、以圻、湧達、王凡、維哲、佳昱，我忘不了卡娜赫拉馬克杯永無止盡的攪拌測試，還有每個博理留宿的夜晚後ㄊ早餐講八卦回家補眠下午再來的虛脫回憶。謝謝實驗室的學弟妹酸酸、駱駱、蛋哥、若曦、凱涵、求求，不只無限次數地幫助我做實驗，給予許多實用的建議，在我沮喪時一句「大不了和我們一起畢業啊」也深深撫慰我受傷的心。

謝謝在資管所畢業致詞時，致詞代表美美、天怡、張翔沒忘記我這資管邊緣人，讓我能在管院區同學的回憶裡插個幾秒鐘的花。謝謝所有臨時被我拉來做實驗的同學與學弟妹，緒暄、阿波、千千、于屏、偉宏和秀，看你們做使用者測試做到眼睛快脫窗的情景，我眞是在最危急的時刻才想起管院的老朋友啊。謝謝我的家人，就算知道兒子畢業迫在眉睫也不輕易問東問西施加壓力。謝謝 Facebook 群組：PEI LI MIN、輕旅行II、MANㄍㄍ、杜寶與他的快樂夥伴，儘管裡面充滿屁話、八卦與嘲諷，但好險有你們願意聽，我才不至於這段時間過後忘記怎麼說話。

謝謝。

楊立銘 BL7G 2016.06.26

doi:10.6342/NTU201700648

中文摘要

連續型視線輸入解決了以視線輸入時眼球顫動及誤觸操作的問題。過去的研究多將焦點集中在掃視式的視線輸入，透過將筆畫操作拆解爲階段直線運動的方式來達到增進操作效率的目的，但爲這種機制設計的視覺導引系統難以直接套用在任意的筆畫手勢上。因此我們結合連續型視線追蹤與動態視覺導引的概念，提出一個專爲連續型視線輸入設計的漸進式動態視覺導引系統 GazeBeacon，能在操作過程中於視線輸入端點周圍持續地提供即時的視覺回饋與前饋。在前導實驗中我們發現兩個問題。我們利用套用平滑濾波器及輸入點重新取樣的兩個方式減緩眼球顫動隊對長度計算帶來的影響，並在路徑導引前方增加一視線集中焦點以解決使用者錯誤判讀引導的問題，設計一個基於不同筆畫組成類型探討的實驗來驗證這個做法。最後我們將 GazeBeacon 與傳統快捷查詢表做操作時間、操作辨識率與操作正確率的比較，發現雖然前者讓使用者花費更多時間，但也顯著地降低了在連續型視線輸入介面上的操作失誤率。

關鍵字: 連續型視線輸入、連續型視線追蹤、動態視覺導引。

doi:10.6342/NTU201700648

Abstract

Gaze-gesture interaction solved the jittering and the Midas Touch problem of gaze-controlled interfaces. Previous works mainly focused on saccadic gaze gestures, increasing efficiencies by dividing gestures into segments. However, those guidance techniques could not directly be applied to any graffiti-like gestures. By combining the concept of smooth pursuit and dynamic guide, we propose GazeBeacon, a gradual visual guiding system designed for gaze-gesture interaction. It continuously provides real-time feedback and feedforward graphical cues under gaze points during the progress of the interaction. Two issues were found in our pilot study: miscalculation and misestimation. We mitigated the miscalculation problem by adding smoothing filters on gaze points and resampling the path before length calculation, while the misestimation problem was solved by adding a focus point at the end of the guidance path. These methods were verified by a user study based on different gesture primitives. Finally, we evaluated the completion time, the recognition rate and the selection accuracy of GazeBeacon, compared with the traditional crib-sheet guide. The result shows that although GazeBeacon makes users spent more time on execution, it significantly improves the accuracy of menu selections on gaze-gesture interfaces.

Keywords: Gaze gesture, Smooth pursuit, Dynamic guide.

doi:10.6342/NTU201700648

Table of Contents

doi:10.6342/NTU201700648

doi:10.6342/NTU201700648

List of Figures

doi:10.6342/NTU201700648

doi:10.6342/NTU201700648

doi:10.6342/NTU201700648

doi:10.6342/NTU201700648

List of Tables

doi:10.6342/NTU201700648

Chapter 1

Introduction

1.1 Motivation

Gaze-controlled interaction uses high-speed cameras detecting gaze movements to control interfaces. It has three main advantages compared with traditional input devices. To begin with, humans tend to move their gaze onto what attracts them, resulting in a direct and natural-to-use way of interaction of gaze control [27]. Then, since eye-tracking techniques compute the position of gaze points by projecting eye movements onto the display, the translation of gaze points are pretty fast when performing saccades between fixations. Finally, gaze-pointing interaction is hands-free. It can be used not only as the main input device for disabilities but also as a complement to other devices such as mouse devices or keyboards.

Although gaze-controlled interaction has several strengths mentioned above, it has two constraints due to the nature limit of eyes. On the one hand, unlike mouse cursors can perfectly fixate on a specific pixel, uncontrollable random eye jitters cause perturbations on the axis, ultimately limit the accuracy of gaze input. On the other hand, human eyes are

doi:10.6342/NTU201700648

always-on. When they are concurrently used as input sensors and output actors, the switch between these modes might be confusing and easily causes unintended actions, which were known as Midas Touch problem [19]. Hence, a better design of gaze-controlled interface must tolerate eye jitters and correctly distinguish between the actual intentions of the users.

A common attempt to prevent unintended actions is to setup dwell time. However, it usually reduces the efficiency and leads to a tedious feeling of users. Gaze gesture interaction [11] tried a totally different approach which was inspired by gesture-based interfaces. Users perform sequential relative eye movements on the interface, forming a gesture to trigger a corresponding command. Instead of relying on perfect fixations on tiny interface objects, gaze gesture interaction bases on relative movements, which can easily be distinguished from normal eye movements, perfectly puzzling out the problem of eye jitters and unintended actions. It also has several advantages inherits from gestural interfaces. First, gestures are easy to learn and memorize. Compared to menu-based interaction like pull-down menus, Gaze-gesture interfaces provides expert users a much more efficient way to interact with. Gaze gestures also further lower the chance of unintended actions by skipping the step of menu exploring and command finding.

Although gaze-gesture interaction solves the issues listed above, it is hard for users to generate system-interpretable gaze strokes without external visual cues. Existing gaze-gesture techniques usually focus on mark-based strokes, which are preliminarily defined and classified into simple linear navigation segments and can be performed as several discrete saccadic eye movements, slightly lowering down the physical execution efforts

doi:10.6342/NTU201700648

of the users. But their vocabularies as input devices might be limited due to the omitted semantic meanings of normal gesture patterns. And because of the lack of transferability between applications, the user experience of one interface can not be directly translated to another one.

Therefore, we aim to design a visual guiding system for gaze gestures. By combining the two concepts *smooth pursuit* and *dynamic guide*, It helps users to perform free-form graffiti-like gaze gestures in an efficient and consistent way.

1.2 Proposed Method

Smooth pursuit [2] is a distinctive form of eye movements which only occurs when people using gaze to track a slowly moving object. Pursuits [28] leveraged this concept to their eye-based interaction technique, whose users executed commands by following a moving object. It enabled a spontaneous gaze-controlled interface and significantly lowered the physical efforts of users.

Dynamic guide is a gesture-based interface guiding technique which dynamically and continuously updates itself throughout the whole progress of gestures. OctoPocus [3] was the first to introduce and organize this concept. By providing continuous on-screen feedforward and feedback, it helped users learn, execute and remember gesture sets.

We combine these two concepts into the design of GazeBeacon. Smooth pursuit indicates that human eyes can not perform smooth trajectories without external stimulus, while dynamic guide helps the users to know the current state of the recognizer. GazeBeacon adds dynamically updated graphical guidances under the gaze point of the users,

doi:10.6342/NTU201700648

gradually guides their eyes to perform smooth gaze movements along predefined gesture paths, ultimately increases the performance and the recognition rate of gaze gesture interactions.

1.3 Contribution

In summary, this paper offers two main contributions:

- Conducted two user studies and several modifications to explore the design decisions of the guiding system on the gaze-gesture interface.
- Proposed a dynamic guiding system, which can be applied on any free-form continuous gaze gestures. The guiding system was evaluated that significantly improves the accuracy of gaze gesture menu selections.

1.4 Organization

This paper is organized as follows.

Chapter 2 describes previous works in two different research regions. Chapter 3 explores the design space by conducting two user studies that solved the two issues found in the pilot study. The details of the design and the implementation of GazeBeacon as well as an evaluation study are presented in Chapter 4. we briefly summarize the contributions of this paper and discuss future works in Chapter 5.

doi:10.6342/NTU201700648

Chapter 2

Related Work

2.1 Visual Guidance for Gesture-Based Interaction

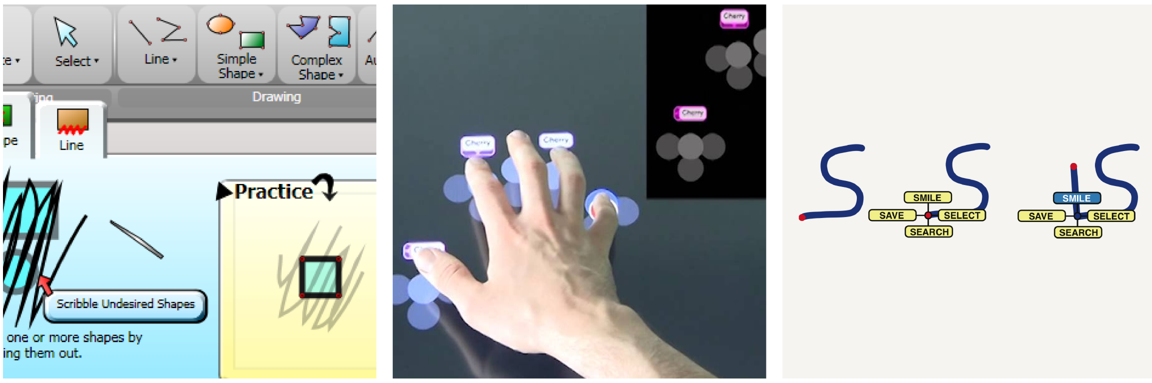

Figure 2.1: Examples of discrete multistep guiding systems. From left to right: Gesture-Bar [7], Arpège [13], Augmented Letters [22].

Gesture-based interfaces provide a direct and natural way to interact with, but most of them are not self-revealing to novice users. There must be hints to instruct the users (1) all available commands and (2) how to issue those commands.

One of the approaches is to simply list all of the available commands at the first time when users face the interface. Crib-sheet guide and Adaptive Guide [1] instructed novice users a set of gesture trajectories one-time right after the guidance was triggered. On the

doi:10.6342/NTU201700648

other hand, some of the techniques chose to display only part of the gesture commands and guide the users in a step-by-step way.

GestureBar [7] was like crib-sheet, but provided multistep animated demonstration of gestures for the users instead. Arpège [13] used finger-by-finger feedforward to gradually guide the users' fingers to multi-finger chords. Augmented Letters [22] combined mark-based menus onto free-form letter gestures to extend their functionalities. However, the feedforward mechanisms used above were all discrete and step-wised, which might not suit for being applied to gaze-gesture interfaces. To gradually guide the gaze points of users, a continuous moving target which can be tracked by users is necessary for guiding systems designed for gaze gesture interaction.

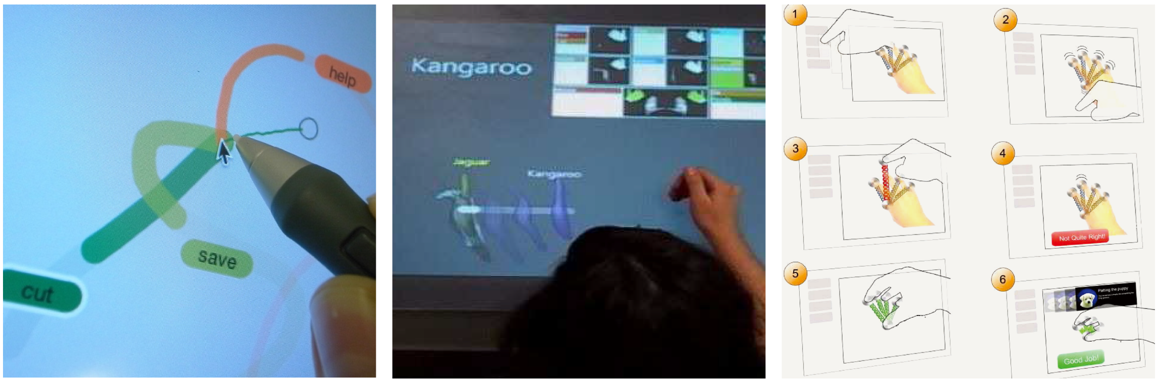

Figure 2.2: Examples of continuous guiding systems. From left to right: OctoPocus [3], ShadowGuides [12], Gesture Play [6].

OctoPocus [3] introduced the concept of dynamic guide. By displaying the octopus-like subsequent remaining path of the subtracted gesture template, it helped the users to learn, execute and remember gestures. TouchGhost [26] demonstrated available multi-touch interaction to the users by simulating real actions with virtual animated hands on the interface. ShadowGuides [12] showed the current gestures of the users and the predicted

doi:10.6342/NTU201700648

actions displayed by the virtual shadow feedforward, which indicated possible ways to finish multi-finger or whole-hand gestures. Gesture Play [6] used physical metaphor feedforward dynamically responded to the gesture input by the users, motivating them to perform multi-finger gestures in a funny way. SimpleFlow [4] provided a scale-free guiding system for uni-stroke gestural interface with auto-completion and gesture-prediction functionalities applied to the enhanced feedforward mechanism. Delamare et al. [10] systematically organized and unified possible design factors of the existing gestural interface guiding systems, then provided an online tool to help future researchers to design guiding systems on gestural interface. Still, neither guiding mechanisms they included had focused on gaze-controlled interfaces. Because of the particular nature of human eyes, there must be further discussions on gaze-gesture interfaces.

2.2 Visual Cues for Gaze Gesture Interface

Gaze-gesture interfaces use predefined sequential relative eye movements to distinguish the users' gesture commands from the natural usage of human eyes. Hence, it basically solves the accuracy issue caused by eye jitters and the problem of unintended actions on gaze-controlled interfaces.

Drewes and Schmidt [11] were the first to introduce gaze gestures. They designed an algorithm to process sets of location-independent discrete consecutive gaze gestures into interface commands. EyeWrite [31] further combined this concept with their previous work EdgeWrite [30] and proposed a sufficient and practical way of eye-typing. They both used square and saltire helping lines or the edges of the display as visual cues to

doi:10.6342/NTU201700648

guide the users to perform eight-direction gaze movements.

Møllenbach et al. [21] introduced single gaze gestures, which further simplified previous ones into saccades from one side to the opposite side of the screen. Istance et al. [18] revised the gestures into two-legged or three-legged gaze gestures in their work and tested them in an MMORPG. The technique resulted in more concentration at the center of the screen of the gamers. Both of them used hot zones of the display to indicate the begin and the end of the gestures.

Gazing with pEYEs [16] applied hierarchical marking menus to their gaze-controlled interface and instructed the users the available commands and the corresponding direction to execute them by several pie-formed slices. Now Dasher! Dash Away! [24] , Pies with EYEs [25] and the work of Best and Duchowski [5] used boundary-crossing to issue commands. The concept was inspired by the well-known text entry technique Dasher [29]. Item selection was performed by crossing a graphical boundary line on the interface.

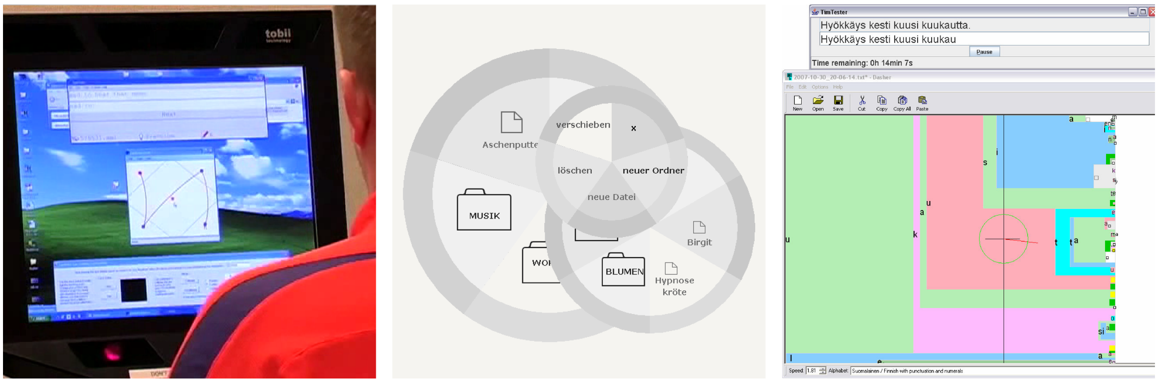

Figure 2.3: Examples of mark-based gaze-gesture interfaces. From left to right: Eye-Write [31], Gazing with pEYEs [16], Now Dasher! Dash Away! [24].

Isokoski et al. [17] pioneered to introduce off-screen gaze gestures. Benefited from utilizing the area outside the screen, they extended the virtual interaction space so the

doi:10.6342/NTU201700648

commands were easy to distinguish from on-screen normal eye movements. This efficient gaze-gesture interaction was further applied to many following works [15] [14] . The physical edge of the screen serves as obvious visual cues in these techniques, not only lowering the execution efforts of the users but also reducing the visual complexity of the interface.

The works mentioned in this subsection increased the performance and decreased the error rate of them by classifying gesture strokes into a sequence of linear directional movements, which can be performed as several saccades between fixations. But all of them are predefined by each different systems, which can not generally be applied to any form of gestures, leading to an inconsistent cross-system user experience for the users. Therefore, we attain to propose a guiding system designed for graffiti-based gaze gestures, which can be directly applied to any existing technique.

doi:10.6342/NTU201700648

Chapter 3

Design Space

3.1 Pilot Study: Understanding OctoPocus

To better investigate how users interact with the dynamic guiding system, we implemented a prototype interface inspired from OctoPocus, then conducted an in-lab pilot study for deeper observation.

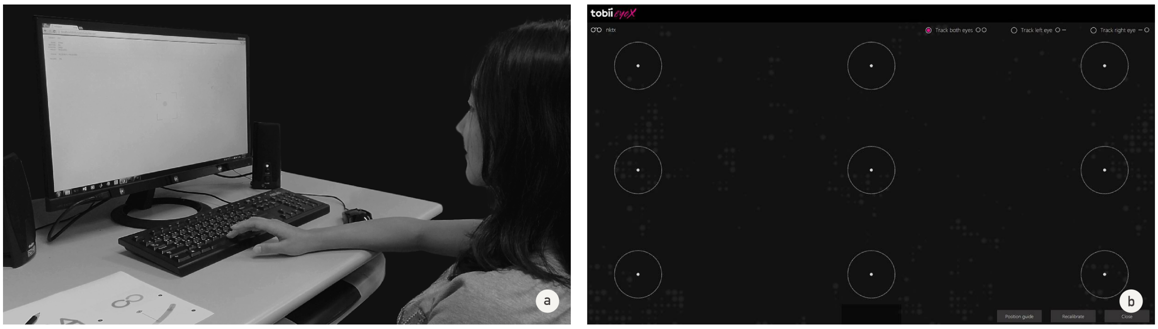

Figure 3.1: (a) The environment of the experiments in this paper. (b) The screenshot of the process of calibration.

Figure 3.1(a) shows the environment of the experiment. Each participant was instructed to sit comfortably at a desk at a distance of fifty centimeters to a 23” display with a resolution of 1920x1080. The gaze of participants was recorded by Tobii EyeX

doi:10.6342/NTU201700648

eye tracker mounted below the screen. The average gaze estimation error was reported as 0.4° of visual angle.

We instructed the participants to calibrate their gaze input by the Tobii built-in software before the experiment. The error of gaze input was controlled below fifty pixels at each of nine calibration points shown in Figure 3.1(b). All of the following experiments in this paper have this step as well.

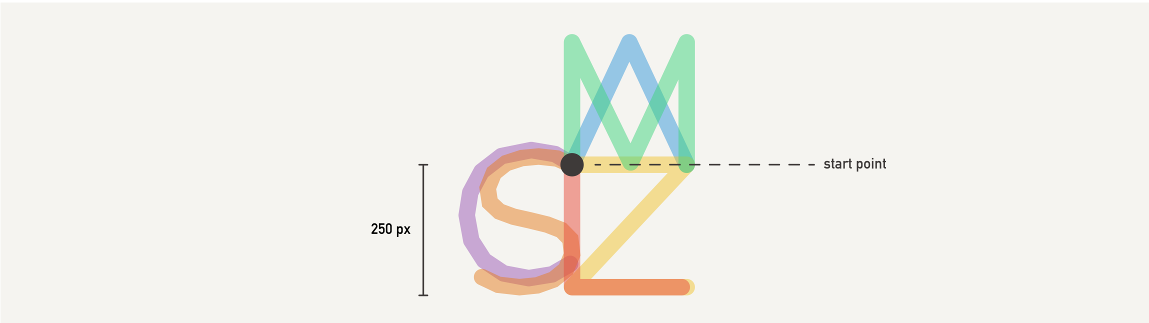

Figure 3.2: The gesture set used in the pilot study, including six gestures combining straight lines and curves.

As shown in Figure 3.2, we defined six basic uni-stroke single-character gestures in our pilot study including L, M, N, A, C, S to cover up possible combinations of straight and curved strokes. We instructed five in-lab participants to perform each of the gestures for three times, and debriefed them after the study.

We found two issues that primarily influence the efficiency and the execution effort of the gaze-gesture interface in our pilot study.

doi:10.6342/NTU201700648

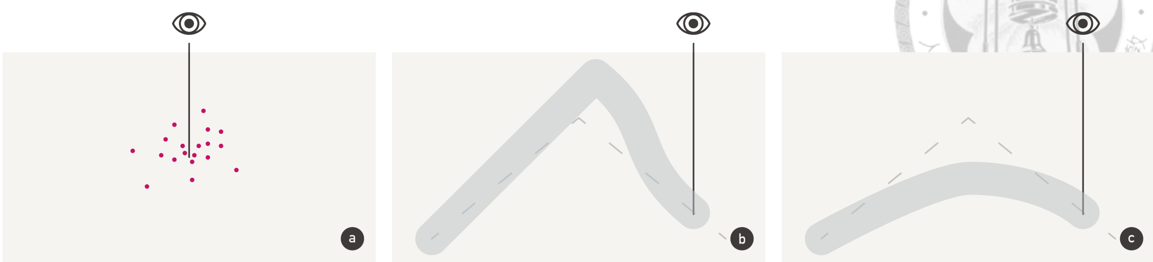

Figure 3.3: The two issues found in the pilot study. (a) The miscalculation problem (b)(c) The misestimation problem.

For one thing, unavoidable eye jitters make the path crooked and rotated, resulting in low recognition rate Figure 3.3(a). Moreover, since the guiding system used in OctoPocus was generated based on the length of the gesture path already input by the users. Unlike mouse can perfectly keep steady fixating on a specific place, natural eye jitters cause perturbations and make the length of the input be counted repeatedly,

For another thing, the prediction provided from the guiding system was sometimes misestimated by the users Figure 3.3(b)(c). The muscle of arms and hands bring better control on mouse-based or stylus-based interfaces and faster reaction to upcoming paths, while gaze control suffers from the misestimations at the corner due to the late reaction caused by the usage of human eyes as input sensors and output actors simultaneously.

From the system's view, these two issues make the gesture less likely to be recognized correctly. From the user's view, they make the experience of gaze-gesture interfaces to be more unsatisfying. By solving them, gaze gesture paths would become more smooth and not be misidentified by the system, raising the performance and the accuracy rate of the users.

doi:10.6342/NTU201700648

3.2 Solving the Miscalculation Problem of Path Length

We proposed two possible solutions: smoothing the path by adding a filter on gaze points, reducing the impact on guiding system from jitters by resampling the path before length calculation.

3.2.1 Gaze Point Smoothing Filter

We concerned One Euro Filter [9] and Kalman filter as our candidates of filtering mechanism. The former is a speed-based filter, it dynamically changes the cutoff frequency according to the speed. Since eye jitters mostly happened when the gaze is fixating or slowly moving, One Euro Filter was thought to be an appropriate filter for eye-switch approached gaze interactions.

However, in the case of performing gliding gaze gestures, which is not rely on saccades jumping between fixations, the slowly-following, less-sensitive-to-jumping Kalman filter helps the users better. Therefore, we choose it as our filtering mechanism (process error covariance = 0.3, measurement error covariance = 18). Although the filtering process lowers down the reaction speed to gaze moving, it is applied only when the guiding system is triggered, hence it don't influence the performance of normal eye movements.

3.2.2 Resample for Length Calculation

The perturbations of eye jitters cause the repeatedly calculated input length, continuously decrease the remaining subtracted path of the gesture templates. Thus, we resample the input points one time before length calculation, reducing the impact of eye jitters.

doi:10.6342/NTU201700648

Before we started, a user study was conducted to better understand the actual influence of eye jitters.

Task and Procedure

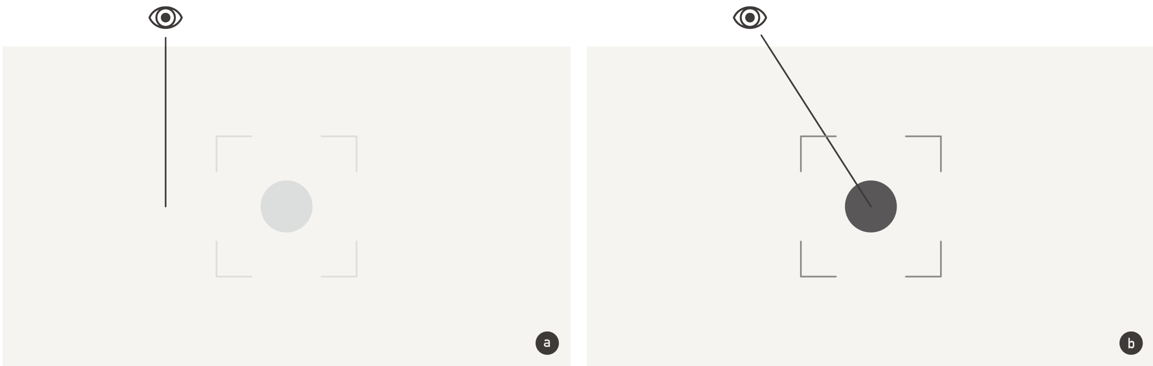

Figure 3.4: The interface and the procedure of the study of understanding the influence of eye jitters.

The environment of this study was the same as the one in the previous pilot study. Figure 3.4 shows the interface. We instructed the participants to fixate their gaze on the black dot at the center of the screen for five seconds, which was measured as the mean completion time of gesture paths, then recorded down (1) the variation of the calculated length of gesture paths through a period (2) the min and the max jittering distance of horizontal and vertical eye jitters.

Participants

Six in-lab participants (4 females) were recruited, ranging from 21 to 23 (mean age = 21.67, s = 1.21). All of them have normal or correct-to-normal vision.

doi:10.6342/NTU201700648

Result and Discussion

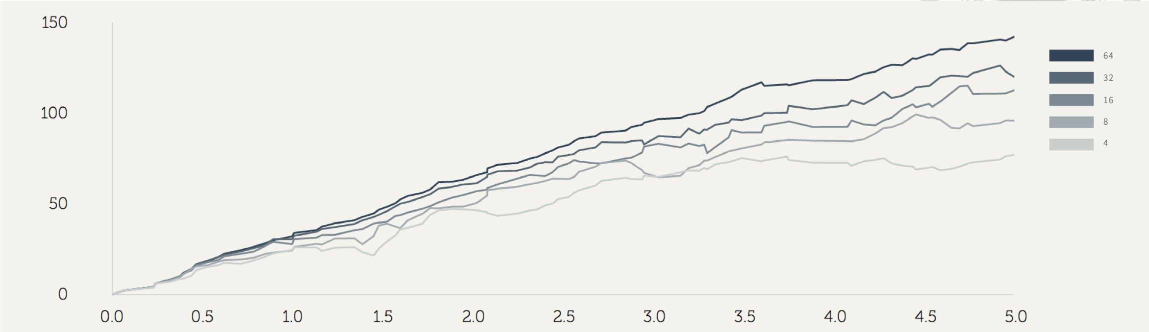

Figure 3.5: The result of the study of understanding the influence of eye jitters.

Figure 3.5 shows the mean curves of calculated input length caused by eye jitters of the six participants that different numbers of points being resampled.

When it comes to the choice of the number of resampling points, the lesser the points we set, the lower the impact from the jitters to length calculation, since the changes of the length were reduced by resampling. But if we resample down too many points, it may result in severe input length fluctuating, which leads to discontinuous graphical guiding paths when we concatenated the subtracted templates onto them based on the length calculation. From the debriefing of the participants, we found that resampling to sixteen points was a rational choice of resampling with the min length of six gestures was 500 px, the jumping of the guiding system was too subtle for the participants to notice.

3.3 Solving the Misestimation Problem of Path Guidance

we found that the user behavior using OctoPocus on gaze-controlled interfaces is quite different from the one on mouse-based or stylus-based interfaces because the conscious

doi:10.6342/NTU201700648

and control ability of the human is much higher to the latter. The behavior of gaze on gaze-gesture interfaces is much like quickly tracing a specific moving object rather of consciously traveling along a path. Thus, it might not able to take on-time reactions to the changes of the upcoming path guidance when the gaze of the users are concentratedly focused on the end point of the path.

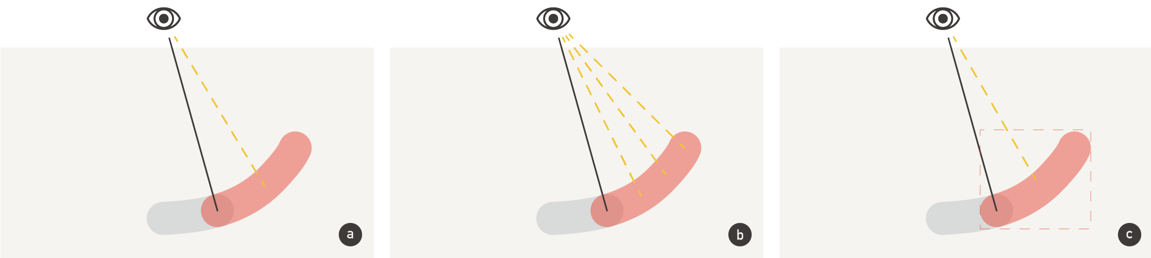

Figure 3.6: Different interpretations of the path guidance

Meanwhile, if the guiding system only provides guiding paths, we found that the users might (1) not sure where to put their focus on Figure 3.6(a) (2) randomly switch the focus on wherever inside the path (3) use an overall approximate form of the whole path as a focus, these inappropriate interpretations ultimately turn into misleading guidances to the users adversely.

Hence, we added a focus point at the end of the original guiding path, actively prompting the users to put their focus on a consistent place, unifying their variant interpretations to the guidance. Furthermore, we designed two different behaviors of the focus point and tested them in the following user study. By comparing with the guiding system that only provides the feedback of gesture inks, we wanted to know how the visual form of the guiding system might influence the operation of the users performing gaze gestures.

doi:10.6342/NTU201700648

3.3.1 Modifications for Gliding Interaction

Gliding describes a behavior of eye movements that the gaze smoothly *glides* along with a predefined path. To make sure the users performing better gesture paths, we proposed two modifications on the prototype based on the findings in the pilot before the experiment.

Precision of the Feedback of Gaze Point

The accuracy of eye trackers is limited due to natural and technical constraints. If the provided feedback of gaze points is in fairly accurate visual forms, such as a pointer or a crosshair, there might be a gap between the expected fixation place and the actual input point (Figure 3.7(a)). It easily misguides the users to unconsciously adjust and trace the feedback point, making the fixation or gliding eye movements less performable due to the unsteadiness.

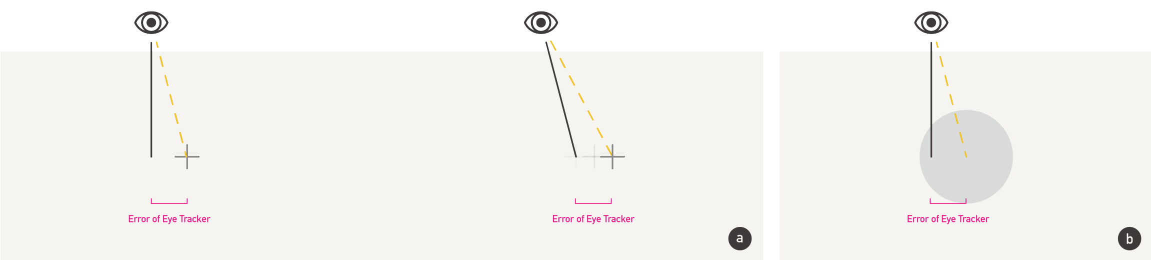

Figure 3.7: The modification on the feedback of gaze point. (a) Describes the tracing behavior of the users due to the offset error of eye trackers. (b) A general blurred form of the feedback of gaze point, covering up the offset error of eye trackers.

Therefore, we modified the feedback of the gaze point into a blurred and general circular form (Figure 3.7(b)). Unlike the original one, the new visual form blurred the accurate position of the expected fixation place and the actual input point, making the

doi:10.6342/NTU201700648

users less likely to notice the distance offset between them. The behavior was turned from directly projecting the gaze onto the display into slightly moving and pushing a circular object by the gaze.

Correction of the Feedforward of Guidance

The correction of the path is also an important part of guiding systems. OctoPocus [3] directly relocated the upcoming path guidance onto the current position of the users' gaze point (Figure 3.8(a)). The advantage of this method is that the operation of the users might not be restricted by the initial start position of the menu. However, the already-made offset mistakes might not be noticed by the users and might keep cumulating.

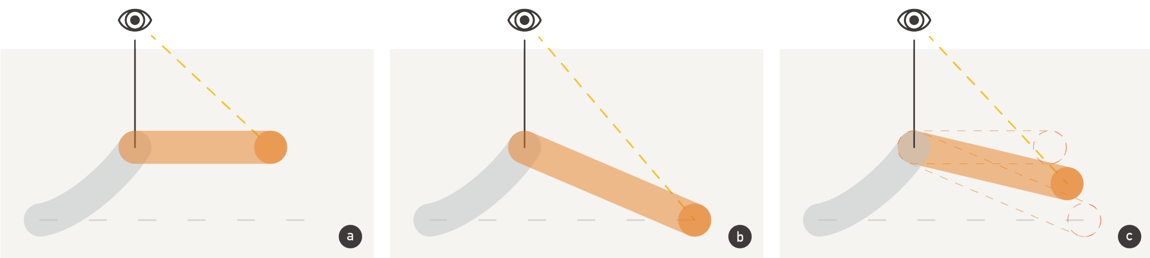

Figure 3.8: The modification on the feedforward of guidance. (a) Use relocated position to depart from the constrain of the intended gesture template. (b) Use smooth correction to subtly redirect the users back to the intended gesture template. (c) Combine both mechanisms based on realtime recognition rates.

Multimodal motion guidance [23] used a presentation form of feedforward called smooth correction. By combining the initial gesture guidance and the relocated position with linear interpolation, it gradually redirected the already-offset path of the users back to the intended trajectory with smooth guiding paths to increase the recognition rate (Figure 3.8(b)).

doi:10.6342/NTU201700648

But the smooth correction method puts too much weight on the initial gesture path. Therefore, we used the current recognition rate as a weighted score to dynamically balance the proportion of these two methods (Figure 3.8(c)). When the recognition rate is high, that is, the gesture ink by the users is still closely matched to the corresponding gesture template, the technique tended to use smooth correction subtly redirecting the current position. On the contrary, the correction prone to use relocated position when the recognition score is low. Since the path had already deviated from the intended one, it chose to ignore the former cumulated offset mistakes and open up a new gesture trajectory.

3.3.2 Task and Procedure

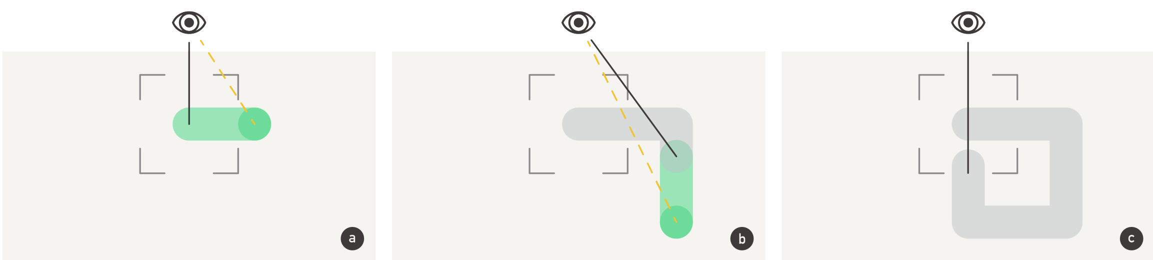

Figure 3.9: The interface and the procedure of the study of solving misestimation problem.

The environment of this study was the same as the one in the previous pilot study. The interface is shown in Figure 3.9. At each trial, the participants were instructed to perform a gaze gesture with one specific guiding technique. After they moved their gaze into the trigger area at the center of the screen, they could start generating a gesture path by pressing the space key, holding it until they satisfied with the current form of the gesture ink. The recognizer used in this study is $1 Recognizer, a 2D single stroke recognizer based on instance-based nearest-neighbor classifier with a Euclidean scoring function. It

doi:10.6342/NTU201700648

compared the gesture ink with the corresponding gesture template after the users released the key, and the gesture result and the output recognition rate were recorded.

The goal of this study is focused on the influence of guiding techniques on the recognition rate, hence the participants were instructed to perform a path matched perfectly to the gesture template, no need to concern the completion time. Moreover, cause saccade is a natural behavior for human eyes to complete straight-line direction movements, eye-switch is not encouraged but neither restrained.

Guidance Techniques

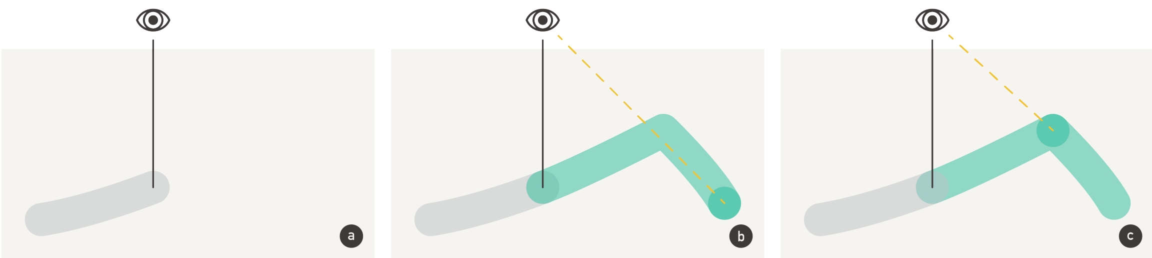

Figure 3.10: The three different guidance techniques. (a) Crib-sheet guide, already-input gesture ink provided only. (b) Path Feedforward guidance. (c) Adaptive Path Feedforward Guidance.

We compared three different guiding techniques Figure 3.10.

(a) The first one only provided already input gesture ink feedback, the users understood and performed the available gaze gestures with an external crib-sheet guide.

(b) The second added an upcoming path guidance feedforward onto the current gaze point. And we added a circular guidance called focus point at the end of the path prompting the users to put their focus on, then gradually moving gaze along the path by tracking the object.

doi:10.6342/NTU201700648

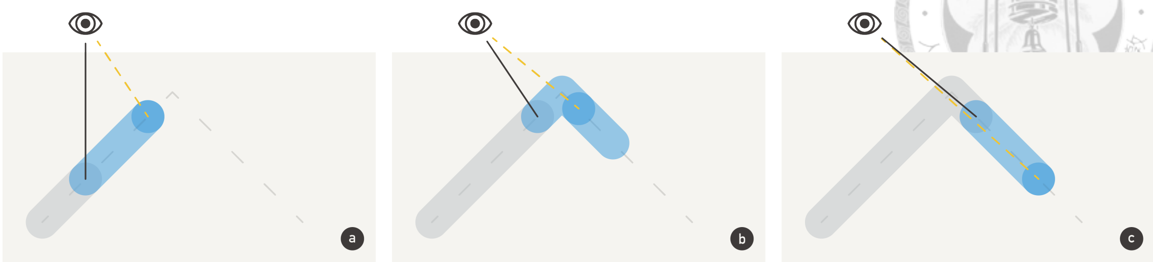

Figure 3.11: A demonstration of the focus point of the adaptive feedforward guidance waiting at the corner.

(c) The last was an alternative version of the second, by dynamically moving the focus point along the path guidance, the focus point would wait at corners (Figure 3.11) pulling back the users' focus to help them perform a better gesture while the users could still be aware of the upcoming direction changes.

Since the last guiding techniques split the feedforward path guidance into two parts: ahead of and behind the focus point, we described these three individual visual feedforward guiding techniques as crib-sheet feedforward guidance, dynamic path feedforward guidance, and dynamic adaptive path feedforward guidance.

Gesture Primitives

Any free-form single stroke gesture can be divided into three basic composite elements: line, corner and arc [8].

doi:10.6342/NTU201700648

Figure 3.12: Three gesture primitives used in the study. (a) Four gestures contained in line section are straight lines pointed to different directions. (2) Four gestures contained in corner section are clockwise and counterclockwise squares and triangles. (c) Two gestures contained in arc section are clockwise and counterclockwise circles.

Therefore, this study was organized into three sections based on the three primitives (Figure 3.12).

(a) There were four gestures in line section: straight lines pointed to four different directions (top-right, bottom-right, bottom-left and top-left). We wanted to know the influence of guiding system on gaze gestures in different directions. We removed the rotation correction process of the stroke recognizer in this section so that the four gestures would not be confused with each other, meanwhile, the angle of the direction was also taken into the concern of the degree of gesture completion.

(b) There were four gestures in corner section: clockwise and counterclockwise rectangles and triangles. We wanted to know if the waiting behavior of the focus point had influences on the corners of gaze gestures.

(c) There were two gestures in arc section: clockwise and counterclockwise circles. We wanted to know whether the curvature of the path guidances give a hint or a metaphor of the whole gaze gestures or not.

There were ten gestures in total in the user study. We instructed the participants to

doi:10.6342/NTU201700648

repeat five times for each gesture paired with different forms of guiding techniques. Because the guiding behavior of path and adaptive feedforward were the same in line and arc sections, each participant received (4×2 + 4×3 + 2×2) (gestures × guidance techniques) × 5 (repeats) = 120 trials overall.

3.3.3 Participants

Nine participants (7 females) were recruited, ranging from 21 to 25 (mean = 22.44, s = 1.51). All of them have normal or correct-to-normal vision.

3.3.4 Result and Discussion

To analyze the performance of three guidance techniques, we conducted two-way ANOVA and used Tukey-Kramer for multiple comparisons.

doi:10.6342/NTU201700648

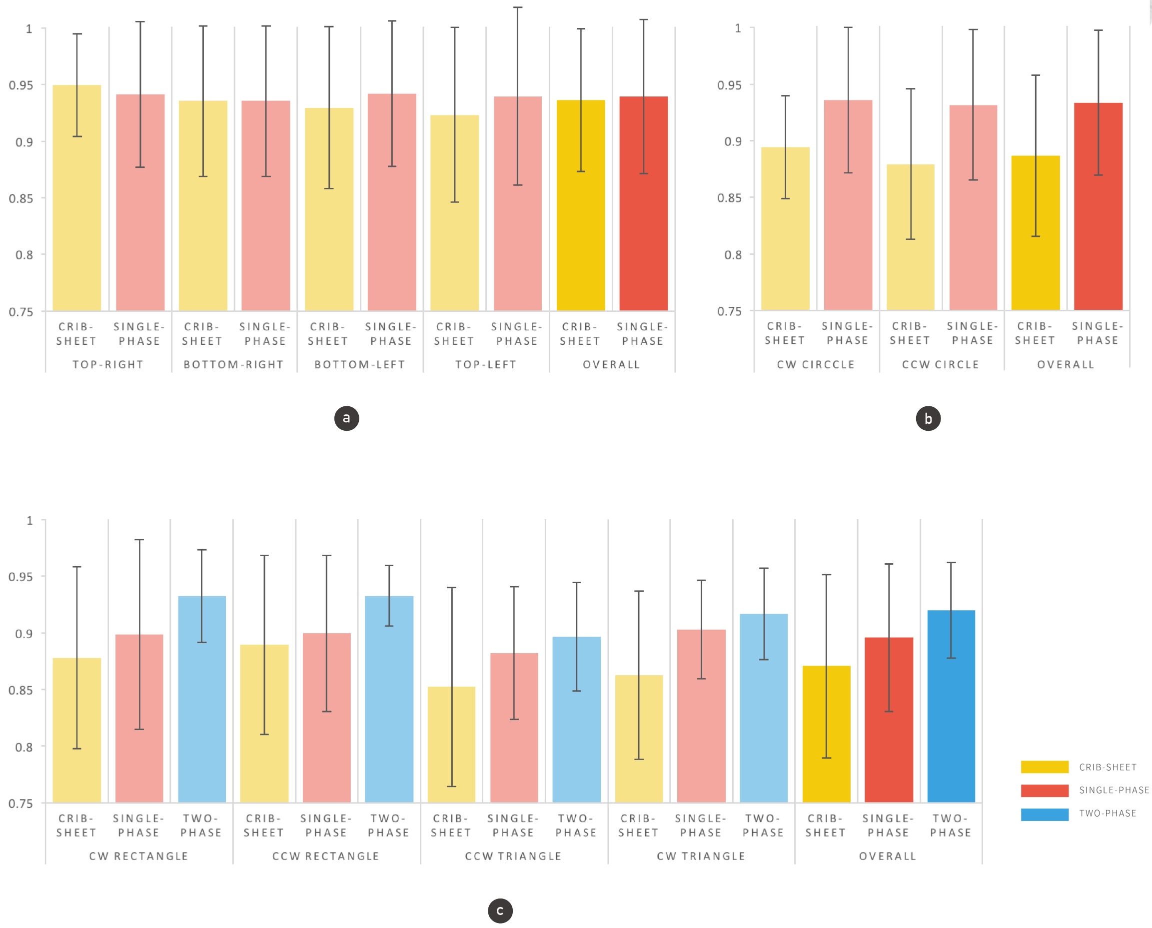

Figure 3.13: The result of the study of solving misestimation problem. (a) The result of line section. (b) The result of arc section. (c) The result of corner section.

Straight Line Vectors

The result of line section is shown in Figure 3.13(a). There was no significant difference between guidance techniques for the mean completion degree (F(1,359) = 0.215, p = 0.643). This might because the jumping between fixations is already a strength of gaze interaction, the recognition rate of line gaze gestures with the crib-sheet guide was high enough so that adding guidance did not help.

doi:10.6342/NTU201700648

Gestures that Contain Corner

The result of corner section is shown in Figure 3.13(c). There was a significant difference among guidance techniques for the mean completion degree (F(2,539) = 26.486, p<0.001). All pairs of the guidance techniques also had a significant difference ((crib, single): p<0.001, (crib, two): p<0.001, (single, two): p = 0.002). Except for the gaze gestures with feedforward were performed better than the ones with the crib-sheet guide, the adaptive feedforward also had impacts on the gestures. By waiting at the corner, the focus point guided the users to perform clipped corners instead of rounded ones.

Gestures that Contain Arc

The result of arc section is shown in Figure 3.13(b). There was a significant difference between guidance techniques for the mean completion degree (F(1,179) = 54.908, p<0.001). The paths with path guidance were also much smoother, we infer that the curvature of the path guidances may give a hint of a general impression of the gesture to the users.

doi:10.6342/NTU201700648

Chapter 4

GazeBeacon

Based on the findings from previous studies, by combining the concept of smooth pursuit and dynamic guide, we design a gradual visual guiding system for gaze gesture interaction. It continuously provides real-time feedback and feedforward cues around the gaze point during the progress of gaze gestures.

4.1 GazeBeacon

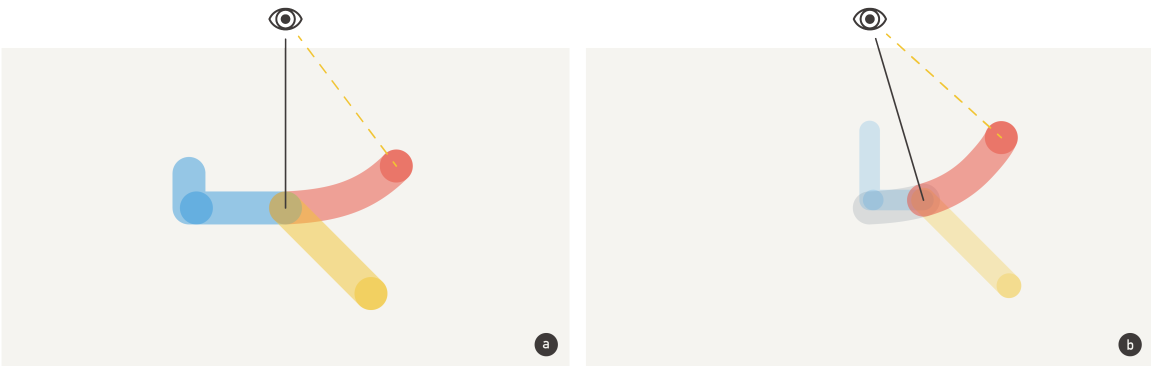

Figure 4.1: A demonstration of the interaction of GazeBeacon. (a) The initial start form of GazeBeacon. (b) the changes of the guidance through the process of gaze gesture input.

doi:10.6342/NTU201700648

Figure 4.1(a) shows the process of GazeBeacon. When the menu is triggered, GazeBeacon will spread up an octopus-like guidance under the current position of the gaze point, listing all currently available commands and their corresponding gestures.

Figure 4.1(b) shows how GazeBeacon using gradual transitions of visual elements to guide the users to perform gaze gestures. When the user starts generating stroke path, the commands will dynamically change their form, responding to the users' input. The thinner and the more transparent guidances are less unlikely to be interpreted. GazeBeacon real-timely indicates the state of the recognizer by changing its visual presentation. If the possibility of one command is lower than the predefined threshold, the guide to it will directly disappear.

4.1.1 Guidance Design

Dynamic Guide is usually categorized as feedback and feedforward. Feedback mechanisms provide the information of the results of the already-executed actions in the past, while feedforward mechanisms offer the info of all the available commands and the way to execute them in the future.

The visual cues we used on the interface of GazeBeacon will be described as followings.

doi:10.6342/NTU201700648

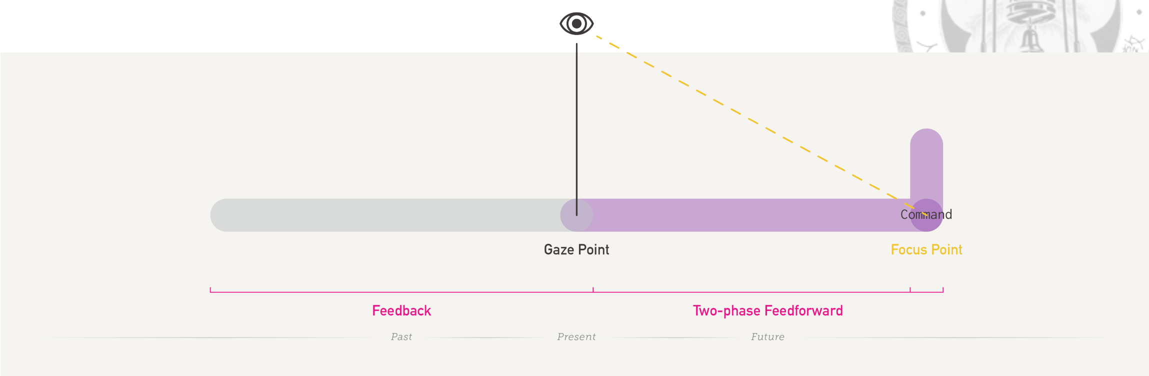

Figure 4.2: The components of the feedback and the feedforward guiding techniques of GazeBeacon.

Visual Feedback Mechanism

- Gaze point: The current position of the user's gaze input, presented as a dark gray circle with a twenty-pixel radius.
- Already-drawn stroke path: The already performed portion of a gesture path, presented as a light gray path with forty pixel width.
- Final recognition result: After the user commits the input path, the system will receive a recognition result. If the path is interpreted as one of the gesture commands, the already executed path by the user will turn into the corresponding color of the command indicating the final result.

Visual Feedforward Mechanism

- Path guidance: The upcoming path of the gesture subtracted from the template indicating the path to execute the available commands, presented as fifty-pixel-wide path coded with the predefined color of the command.
- Focus point: A circular guidance prompting the users to put their focus on, which

doi:10.6342/NTU201700648

also serves as the moving object in smooth pursuit eye movement that the users' gaze should track of, presented as a circle with twenty pixels radius coded with the predefined color of the gesture.

- Command description: Display the name of the command in dark gray texts.

Impact of Feedback onto Feedforward

GazeBeacon not only displays already-executed gesture paths but also provides real-time recognition results. Inspired by OctoPocus [3], we use the thickness and the opacity of the guidances to indicate the current state of the recognizer, let the users know how their input been processed.

The dynamical visual changes of the guidances are based on the recognition rate returned from the recognizer, which is ranging from zero to one, indicating the coincidence between the users' input and the gesture template.

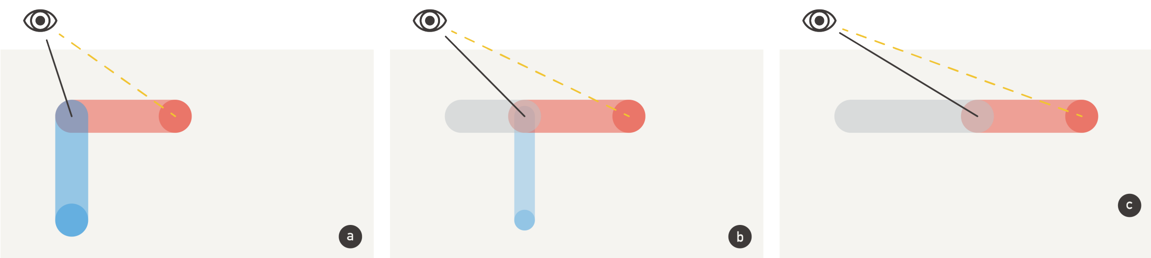

Figure 4.3: The graphical changes based on the recognition rate help the users to focus on the currently performing gesture.

At the beginning of the gesture process, all of the commands have the same possibility. As the user moved along one path, the tracked one remained the possibility and kept the original thickness and the opacity while others gradually got thinner and more transparent.

doi:10.6342/NTU201700648

This design let users put much focus on the current path but still have the chance to instantly change to the other path with conscious.

4.1.2 Possible Applications

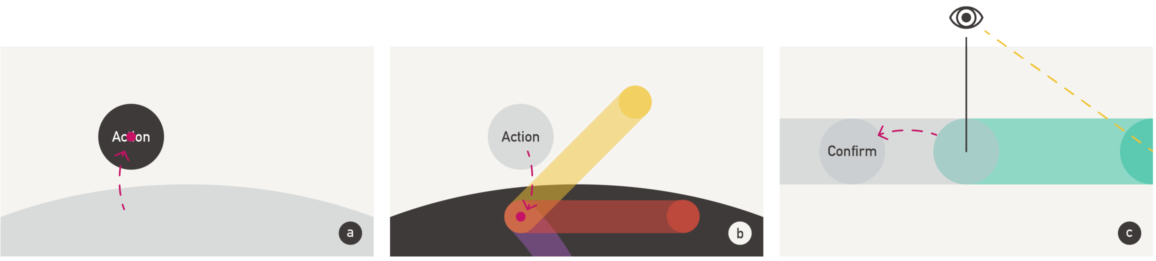

Figure 4.4: The possible trigger method of GazeBeacon.

Reverse crossing [20] was a dwell-time-free selection mechanism which used border-crossing in-and-out gaze actions to differentiate intended selections from normal movements.

When applied to our guiding system, we can generate reverse crossing buttons near gaze points, using the mechanism to trigger specific actions or confirm on-going gliding movements shown in Figure 4.4, then the guiding system can be used on real-case applications.

Figure 4.5: The possible usage scenarios of GazeBeacon.

4.2 Evaluation

In order to determine whether the dynamic feedforward and feedback guiding system indeed help the users to perform better gaze gestures, we conducted a within-subjects study to compare GazeBeacon with Crib-Sheet guide on completion time, recognition rate, and selection accuracy.

4.2.1 Task and Procedure

Figure 4.6: The interface of GazeBeacon and the gesture-command pairs used in the evaluation study.

We created a simple five-item gesture-command shortcut menu, the interface, and the gesture set is shown in Figure 4.6. Each of the gesture was mapped to a common command of word processing applications including 'Copy', 'Paste', 'Undo', 'Save' and 'Select All'. The gestures to execute them also covered up possible combinations of straight lines and curved lines.

we instructed the participants to perform gaze gesture selections on the shortcut menu with the guidance of GazeBeacon or with the traditional Crib-Sheet. When the study began, a task command would pop up at the center, the participants should follow the

doi:10.6342/NTU201700648

assignment and the results were recorded.

The order of the guiding techniques was counterbalanced, ensuring the training effect for subsequent conditions were avoided. Each gesture was repeated for ten times and randomly ordered, overall, each participant received 5 (gestures) × 10 (repeats) × 2 (guidance techniques) = 100 trials in the study.

4.2.2 Participants

Eight participants (all females) with novice-level eye tracking experience were recruited, ranging from 22 to 24 (mean age = 22.875, s = 0.83). All of them have normal or correct-to-normal vision.

doi:10.6342/NTU201700648

4.2.3 Result and Discussion

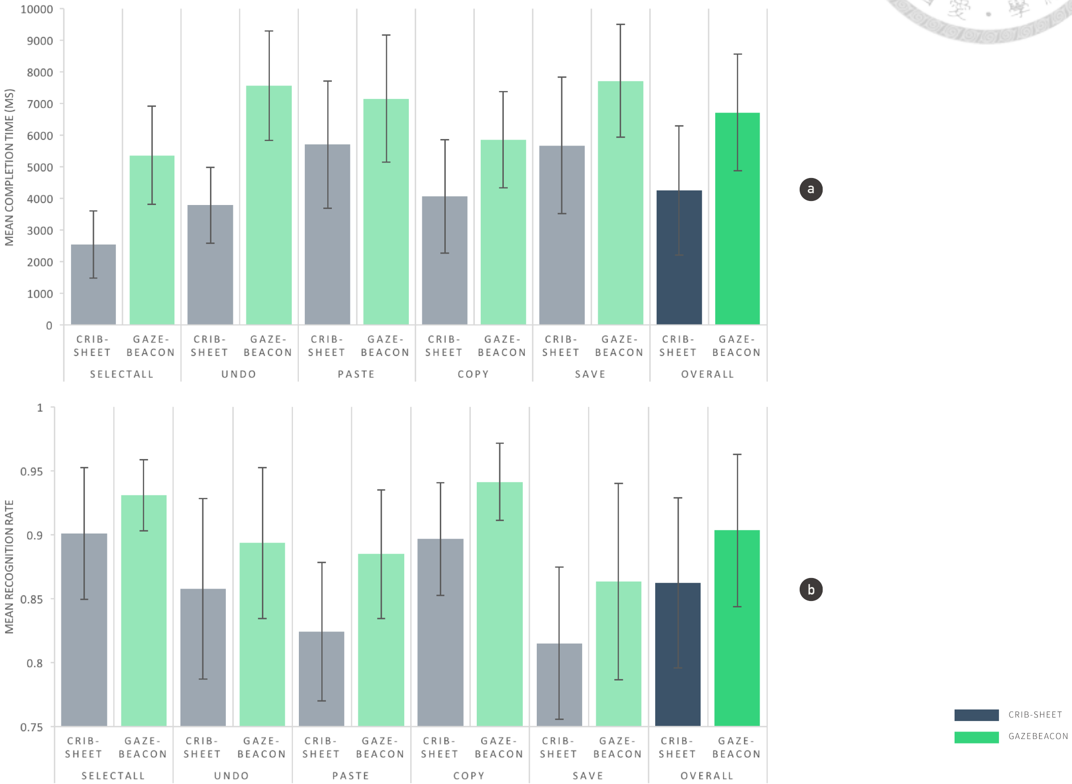

Figure 4.7: The mean completion time and mean recognition rate of Crib-Sheet and GazeBeacon.

The result of the experiment is shown in Figure 4.7. The completion time of GazeBeacon is significantly longer than Crib-Sheet (p<0.001), while the recognition rate of the former is significantly higher the latter (p<0.001). It might due to the trade-off between the correct rate and performance of menu execution. With the dynamic visual guide opening, the users tended to be more cautious on the gesture path, resulted in the lower performance but the higher recognition rate.

doi:10.6342/NTU201700648

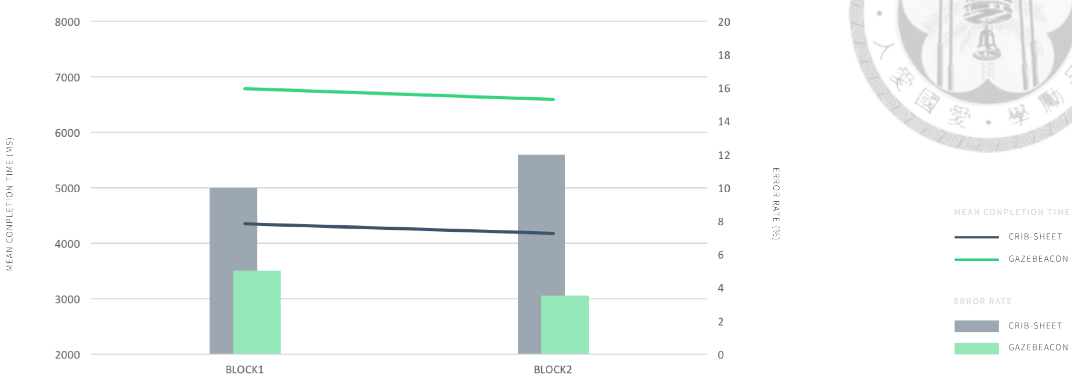

Figure 4.8: The mean completion time and error rate of Crib-Sheet and GazeBeacon in block 1 and block 2.

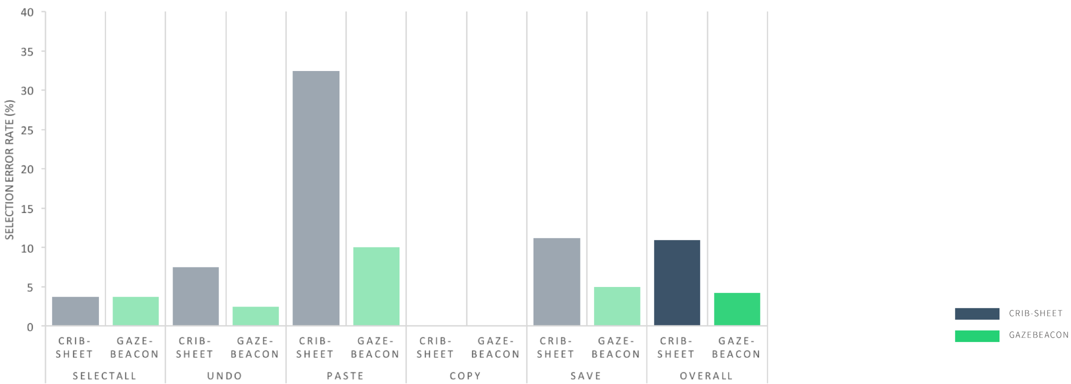

Figure 4.9: The error rate of Crib-Sheet and GazeBeacon.

TASK ASSIGNMENT

(a) CRIB-SHEET — RECOGNITION RESULT

	SelectAll	Undo	Paste	Copy	Save
SelectAll	77		3		
Undo		74	1	5	
Paste	2	14	54		
Copy				80	
Save	6		1	2	71

(b) GAZEBEACON — RECOGNITION RESULT

	SelectAll	Undo	Paste	Copy	Save
SelectAll	77	1	1		1
Undo	2	78			
Paste	3	5	72		
Copy				80	
Save	3			1	76

Table 4.1: Confusion matrix by gesture for different guiding techniques. (a) *Crib-Sheet*. (b) *GazeBeacon*.

doi:10.6342/NTU201700648

We used chi-square tests to analyze the result of the error rates of the two techniques. The error rate of GazeBeacon is significantly lower than the Crib-Sheet guide (Figure 4.9) (p<0.001), inferring that GazeBeacon improves the correct rate of gaze gestures.

Enhancing the Features of Gestures

For deeper understanding, Table 4.1 shows the confusion matrix by gesture. The error rate of 'Select All' and 'Copy' has no difference between the two techniques, this might because the two were both single-element gestures, they were more likely to be interpreted.

However, 'Undo' tended to be misinterpreted as 'Copy' if the first corner element was not performed well. 'Save' tended to be misinterpreted as 'Select All' if the second arc element was not performed well. The error rate of the complex gestures were significantly different between the two techniques (p<0.001), especially the one of 'Paste', combining corner and arc make it much harder to perform and be correctly recognized.

Figure 4.10: GazeBeacon enhanced and emphasized the essential features of the gestures.

GazeBeacon enhanced and emphasized the essential features of the gestures, decreasing the error rate of the recognition. We believe that the results suggest that GazeBeacon takes more advantage on improving the selection accuracy of complex graffiti gesture input.

doi:10.6342/NTU201700648

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this paper, we proposed GazeBeacon, a gradual visual guiding system designed for gaze gesture interaction. It continuously provides real-time feedback and feedforward graphical cues under gaze points during the progress of the interaction by combining the concept of smooth pursuit and dynamic guide. We found two issues in our pilot study: miscalculation and misestimation. First, we mitigated the miscalculation problem by adding smoothing filters on gaze points and resampling the path before length calculation. Second, we solved the misestimation problem by adding a focus point at the end of the guidance path, prompting the users to put their focus on the continuously tracking object. A user study was conducted to further verify the influence of different guidance techniques on various gesture primitives. After solving the problems, we describe the details of the design and the implementation of GazeBeacon. Then we further evaluated the completion time, the recognition rate and the selection accuracy of GazeBeacon, compared with the traditional crib-sheet guide. The result shows that although GazeBeacon

doi:10.6342/NTU201700648

makes users spent more time on execution, it significantly improves the accuracy of menu selections on gaze-gesture interfaces.

5.2 Future Work

Possible future works can be organized as follows.

5.2.1 Continuous v.s. Discrete Gaze Gestures

One of the common approaches of gaze gestures is saccadic eye movements. By Simplifying the gestures and mapping them onto existing encoding system such as EdgeWrite, the gestures are composed of only straight lines, and take benefits from the nature of human eyes being skilled in saccadic straight direction movements.

However, the straight-line-formed gestures should be predefined by each different systems and lead to inconsistent user experiences for the users switching among applications. On the other hand, if the pattern was not defined by the system yet, the users might be confused about the actual gesture of the commands.

Although the efficiency of GazeBeacon in this study is lower, the continuous feedforward and feedback of it can be used on every single-stroke gaze gestures, including graffiti-based ones. Since the users remember the paths not only the dots, GazeBeacon can keep inheriting the benefits of gesture-based interfaces like various semantic meanings.

doi:10.6342/NTU201700648

5.2.2 Modifications on the Recognizer

In the implementation of GazeBeacon, we used a well-known stroke recognizer called \$1. It was widely applied on many gesture-based interfaces. However, due to the nature of human eyes, the recognizer needs further modifications for gaze-controlled interactions, the shape matching method used in it might be extended based on the features of the gaze gestures.

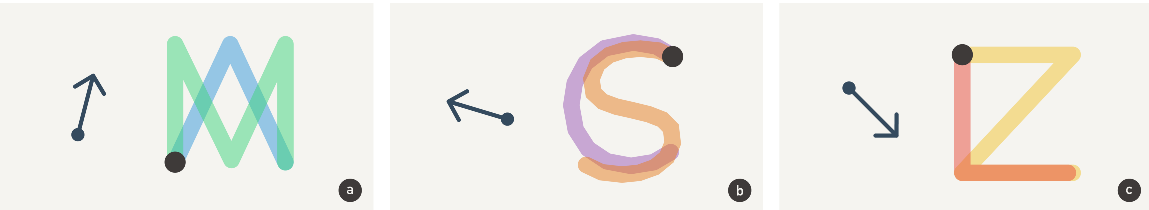

Figure 5.1: The recognizer can pre-filtering the guidances of the gesture input by its initial movement.

For instance, the rotated angle should be limited in a specific range so that the A shape may not be confused with C shape since their direction of the initial move is totally different. By the modifications of the used recognizer, it might improve the accuracy of the response from the guidance system to the users' input.

5.2.3 General v.s. Specific Guidance Design

In the guidance design of GazeBeacon, we proposed a method could be directly applied to general usage cases. However, we noticed that the behavior of the users on various gesture elements was quite different from each other in the study. For example, the proper value of the waiting threshold of the focus point on squares and rectangles are not the same as the one on triangles, since acute angles need less emphasize. And

doi:10.6342/NTU201700648

for horizontal straight lines, the users tends to move a little bit higher than the intended gestures.

On the contrary, The offsets within one individual user are quite similar, they were inclined to repeat the offset mistakes previously made by themselves. Therefore, if we want to extend the guiding ability based on the general design, we can further discuss some specifications. On gesture elements, we might adjust the trajectory of the guidance based on the mentioned findings, make it not fully matched to the gesture template. On individual users, we can dynamically tune the guidance by neutralizing the offset to gradually redirect the users back to the intended one.

doi:10.6342/NTU201700648

Appendix A

Mean Paths of the Gesture Input

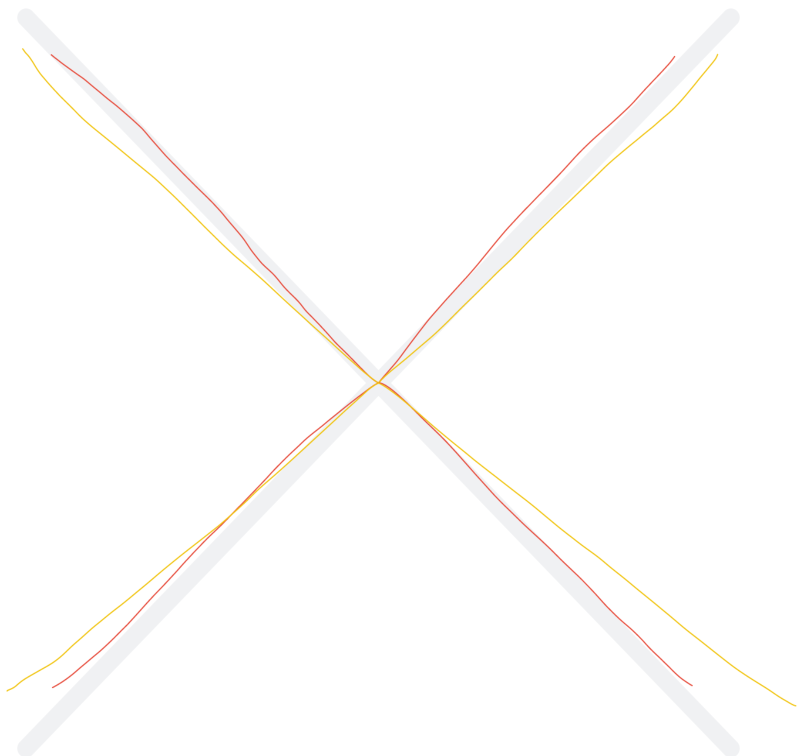

Figure A.1: The mean paths of the gestures input with different guiding techniques in line section.

doi:10.6342/NTU201700648

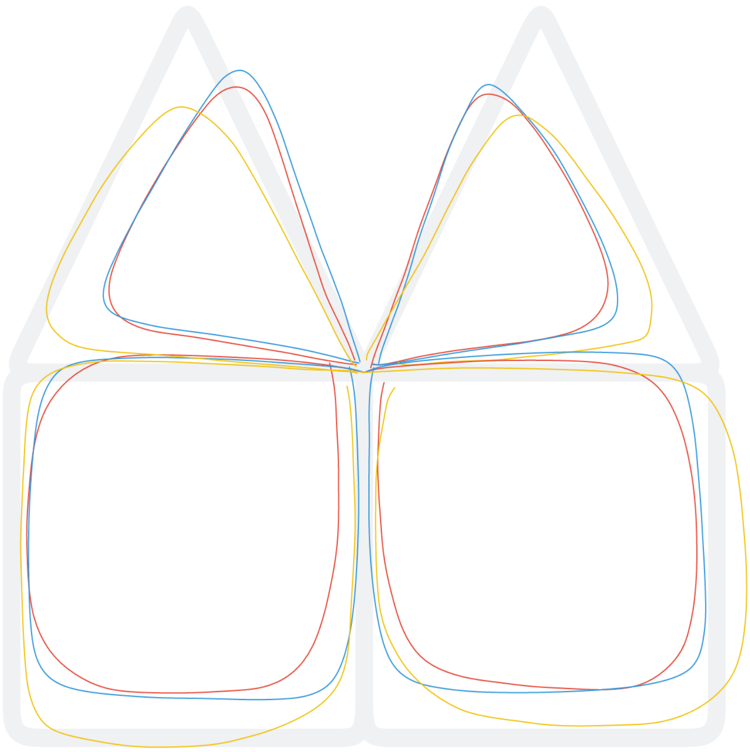

Figure A.2: The mean paths of the gestures input with different guiding techniques in corner section.

doi:10.6342/NTU201700648

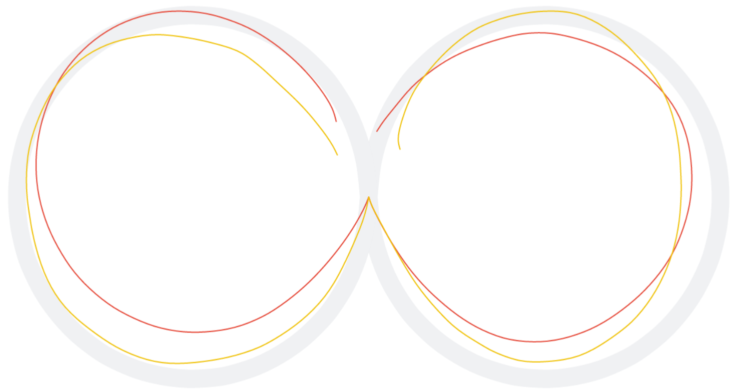

Figure A.3: The mean paths of the gestures input with different guiding techniques in arc section.

doi:10.6342/NTU201700648

Bibliography

[1] F. Anderson and W. F. Bischof. Learning and performance with gesture guides. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1109–1118. ACM, 2013.

[2] G. Barnes. Rapid learning of pursuit target motion trajectories revealed by responses to randomized transient sinusoids. *Journal of Eye Movement Research*, 5(3):4, 2012.

[3] O. Bau and W. E. Mackay. Octopocus: a dynamic guide for learning gesture-based command sets. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 37–46. ACM, 2008.

[4] M. Bennett, K. McCarthy, S. O'Modhrain, and B. Smyth. Simpleflow: enhancing gestural interaction with gesture prediction, abbreviation and autocompletion. In *Human-Computer Interaction–INTERACT 2011*, pages 591–608. Springer, 2011.

[5] D. S. Best and A. T. Duchowski. A rotary dial for gaze-based pin entry. In *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, pages 69–76. ACM, 2016.

[6] A. Bragdon, A. Uguray, D. Wigdor, S. Anagnostopoulos, R. Zeleznik, and R. Feman. Gesture play: motivating online gesture learning with fun, positive reinforcement and physical metaphors. In *ACM international conference on interactive tabletops and surfaces*, pages 39–48. ACM, 2010.

[7] A. Bragdon, R. Zeleznik, B. Williamson, T. Miller, and J. J. LaViola Jr. Gesturebar: improving the approachability of gesture-based interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2269–2278. ACM, 2009.

[8] X. Cao and S. Zhai. Modeling human performance of pen stroke gestures. In *Proceedings of*

doi:10.6342/NTU201700648

the SIGCHI conference on Human factors in computing systems, pages 1495–1504. ACM, 2007.

[9] G. Casiez, N. Roussel, and D. Vogel. 1€ filter: a simple speed-based low-pass filter for noisy input in interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2527–2530. ACM, 2012.

[10] W. Delamare, C. Coutrix, and L. Nigay. Designing guiding systems for gesture-based interaction. In *Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, pages 44–53. ACM, 2015.

[11] H. Drewes and A. Schmidt. Interacting with the computer using gaze gestures. In *Human-Computer Interaction–INTERACT 2007*, pages 475–488. Springer, 2007.

[12] D. Freeman, H. Benko, M. R. Morris, and D. Wigdor. Shadowguides: visualizations for in-situ learning of multi-touch and whole-hand gestures. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 165–172. ACM, 2009.

[13] E. Ghomi, S. Huot, O. Bau, M. Beaudouin-Lafon, and W. E. Mackay. Arpège: Learning multitouch chord gestures vocabularies. In *Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces*, pages 209–218. ACM, 2013.

[14] 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*, pages 57–64. ACM, 2016.

[15] 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*, pages 147–154. ACM, 2012.

[16] A. Huckauf and M. H. Urbina. Gazing with peyes: towards a universal input for various applications. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, pages 51–54. ACM, 2008.

[17] P. Isokoski. Text input methods for eye trackers using off-screen targets. In *Proceedings of the 2000 symposium on Eye tracking research & applications*, pages 15–21. ACM, 2000.

[18] H. Istance, A. Hyrskykari, L. Immonen, S. Mansikkamaa, and S. Vickers. Designing gaze

doi:10.6342/NTU201700648

gestures for gaming: an investigation of performance. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, pages 323–330. ACM, 2010.

[19] R. J. 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*, pages 11–18. ACM, 1990.

[20] A. Kurauchi, W. Feng, A. Joshi, C. Morimoto, and M. Betke. Eyeswipe: Dwell-free text entry using gaze paths. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 1952–1956. ACM, 2016.

[21] E. Mollenbach, J. P. Hansen, M. Lillholm, and A. G. Gale. Single stroke gaze gestures. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems*, pages 4555–4560. ACM, 2009.

[22] Q. Roy, S. Malacria, Y. Guiard, É. Lecolinet, and J. Eagan. Augmented letters: mnemonic gesture-based shortcuts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2325–2328. ACM, 2013.

[23] C. Schönauer, K. Fukushi, A. Olwal, H. Kaufmann, and R. Raskar. Multimodal motion guidance: techniques for adaptive and dynamic feedback. In *Proceedings of the 14th ACM international conference on Multimodal interaction*, pages 133–140. ACM, 2012.

[24] O. Tuisku, P. Majaranta, P. Isokoski, and K.-J. Räihä. Now dasher! dash away!: longitudinal study of fast text entry by eye gaze. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, pages 19–26. ACM, 2008.

[25] 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*, pages 93–96. ACM, 2010.

[26] D. Vanacken, A. Demeure, K. Luyten, and K. Coninx. Ghosts in the interface: Meta-user interface visualizations as guides for multi-touch interaction. In *Horizontal Interactive Human Computer Systems, 2008. TABLETOP 2008. 3rd IEEE International Workshop on*, pages 81–84. IEEE, 2008.

[27] R. Vertegaal et al. Attentive user interfaces. *Communications of the ACM*, 46(3):30–33, 2003.

doi:10.6342/NTU201700648

[28] M. Vidal, A. Bulling, and H. Gellersen. Pursuits: spontaneous interaction with displays based on smooth pursuit eye movement and moving targets. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 439–448. ACM, 2013.

[29] D. J. Ward, A. F. Blackwell, and D. J. MacKay. Dasher—a data entry interface using continuous gestures and language models. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 129–137. ACM, 2000.

[30] J. O. Wobbrock, B. A. Myers, and J. A. Kembel. Edgewrite: a stylus-based text entry method designed for high accuracy and stability of motion. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, pages 61–70. ACM, 2003.

[31] J. O. Wobbrock, J. Rubinstein, M. W. Sawyer, and A. T. Duchowski. Longitudinal evaluation of discrete consecutive gaze gestures for text entry. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, pages 11–18. ACM, 2008.

doi:10.6342/NTU201700648