A study on algorithms and applications of eye gaze tracking

Liu, Yi

2018

Liu, Y. (2018). A study on algorithms and applications of eye gaze tracking. Doctoral thesis, Nanyang Technological University, Singapore.

<http://hdl.handle.net/10356/73328>

<https://doi.org/10.32657/10356/73328>

NANYANG
TECHNOLOGICAL
UNIVERSITY

**A STUDY ON ALGORITHMS AND APPLICATIONS
OF EYE GAZE TRACKING**

YI LIU
Interdisciplinary Graduate School
NTU Institute for Health Technologies

2017

A STUDY ON ALGORITHMS AND APPLICATIONS OF EYE GAZE TRACKING

YI LIU

**Interdisciplinary Graduate School
NTU Institute for Health Technologies**

A thesis submitted to Nanyang Technological University in partial
fulfillment of the requirement for the degree of
Doctor of Philosophy

2017

Abstract

The ability to express quickly and effectively in precise language is fundamental for quality of life. However, some physically challenged people have difficulty in communicating using our common method, e.g. Motor neuro disease (MND). People with these severe conditions may not be able to use even speech, and control of the eyes may be the only means to communicate. With the development of modern assistive technology, a number of eye-based typing systems have been proposed to help these physically challenged people to improve their communication ability, which is of tremendous benefit. However, due to the limitation of dwell time, speed of text entry in early eye-typing system tends to be relatively slow. Recently, dwell-free systems have been proposed that are much faster than existing dwell-based typing systems, but can be vulnerable to common text entry problems, such as selection of the wrong letters. Therefore, in this thesis, we first propose a dwell-free system, GazeTry, with two robust recognition algorithms respectively for inferring the words which the user intends to type. The proposed algorithms determine the intended word based on the sequential letters of gaze trace instead of pausing at each individual letter, which is more robust to missing letters and when a neighboring letter on the keyboard is incorrectly selected. Simulation and experiment results suggest that the algorithms have better accuracy and more resilience to common text entry errors than other currently proposed dwell-free systems.

However, the extra eye-tracking device is still an indispensable equipment in the typing system which is inconvenient in some cases. Therefore, we propose a prototype

of eye-based typing system using an off-the-shelf webcam without the extra eye tracker. In the system, the appearance-based algorithm is proposed to estimate the person’s gaze coordinates on the screen based on the frontal face images captured by the webcam. In addition, some critical issues of the appearance-based method are also investigated, which helps to improve the estimation accuracy and reduce computing complexity in practice. The experimental results show the feasibility of eye typing using the webcam, and the proposed method is comparable to the eye tracker under a small degree of head movement.

Gaze estimation with natural head movement is still a major research challenge. Although researchers have made a lot of effort into addressing it, there is one fundamental problem that has yet to be studied extensively in existing appearance-based algorithms that is how to integrate the left and right eye images. In this thesis, we develop a data-collection application that records the pair of face images and corresponding gaze points of subjects. Unlike other existing datasets, our data are collected while subjects are doing daily work under natural head movement without any constraint. We then propose a new reconstruction method that aligns the two appearance spaces of both eyes into the same local structure, and also investigate the impact of degree of head pose variation. The experimental results show that the proposed method outperforms existing combination methods under natural head movement.

Acknowledgments

It is my pleasure to acknowledge the roles of several individuals who were instrumental for completion of my Ph.D. research.

First of all, I would like to express my deepest appreciation and thanks to my main supervisor Prof. Bu-Sung Lee for your patience, motivation and immense knowledge. I still remember and would never forget how you continuously support me and encourage my research. During these years, you have given me invaluable advices on both research and my career. Without your precious support it would not be possible to conduct this research.

Besides my main supervisor, I would also like to thank my thesis advisory committee members, Prof. Martin McKeown, Prof. Deepu Rajan, Prof. Zhenchang Xing for serving as my Thesis Advisory Committee members even at hardship, especially for my co-supervisor Prof. McKeown due to the time difference between Singapore and Vancouver. I would like to thank you for your brilliant comments and suggestions, but also for the hard questions which incentivized me to widen my research from various perspectives. I would like to thank Prof. Bo An. When I was struggling with a research area, your advices and encouragement inspired me.

I would like to acknowledge the valuable input of Dr. Chonho Lee, who contributed to many discussions that helped to build fundamental knowledge. I would like to thank my partner Chi Zhang for your sharp mathematical mind that helped me a lot. My

sincere thanks also goes to my senior Dr. Shanjiang Tang, who has provided me valuable help since I started my Ph.D. research.

A special thanks to my family. Words cannot express how grateful I am to my parents and grandparents for your love, encouragements and confidences in me. I would like to thank Ms. Lu Lyu for your love and understanding when I was anxious, and I made the right choice. I would also like to thank all of my friends who supported me to strive towards my goal.

And finally, last but by no means least, I am grateful to Nanyang Technological University, Nanyang Institute of Technology in Health and Medicine (NITHM), and the Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY) for providing me the opportunity and scholarship to explore research and complete the Ph.D. programme.

Contents

Abstract	i
Acknowledgments	iii
List of Figures	viii
List of Tables	xiii
1 Introduction	1
1.1 Background and Motivations	1
1.2 Research Objectives	5
1.3 Core Contributions	6
1.4 Thesis Organization	7
2 Literature Review	9
2.1 Basics of Human Eye	9
2.1.1 Eye Structure	9
2.1.2 Eye Movement	11
2.2 Eye Tracking	13
2.2.1 Eye Detection	14
2.2.2 Eye Movement Tracking	16
2.3 Assistive Applications of Eye Tracking	18
2.3.1 Interaction and Communication	18
2.3.2 Eye Typing	21

2.4	Eye Gaze Estimation	24
2.4.1	Feature-based Estimation	25
2.4.2	Appearance-based Estimation	26
2.5	Summary	28
3	Robust Recognition Approaches in Eye-Based Dwell-Free Typing	29
3.1	Dwell-free Eye-Typing System	29
3.2	GazeTry System	32
3.2.1	Letter Sequence Generation Module	34
3.2.2	String Matching Module	36
3.2.3	Candidate Ranking Module	40
3.3	Longest Common States Mapping	41
3.3.1	Potential Transition Scoring	45
3.3.2	Candidate Ranking	47
3.3.3	Data Structure	48
3.4	Evaluation	48
3.4.1	Filtreyepding, FdpRevised and MoWing approaches	48
3.4.2	Experiments	50
3.5	Discussion	57
3.6	Summary	61
4	Assistive Text Entry Using Gaze with an Off-the-shelf Webcam	63
4.1	Gaze Estimation From Eye Appearance	63
4.1.1	Eye-Appearance Feature Vector Extraction	65
4.1.2	Eye-Appearance Manifold Verification	66
4.1.3	Manifold Learning Method	68
4.2	Practical Issues	70

4.2.1	Artefacts Reduction	72
4.2.2	Effective Area Detection	74
4.3	Eye Typing Experiment	77
4.4	Discussion	81
4.5	Summary	86
5	Generic Gaze Estimation with Natural Head Movement	88
5.1	Estimation In Two Spaces	88
5.2	Spaces Alignment	90
5.3	Experiment	91
5.3.1	Data Collection	91
5.3.2	Results	93
5.4	Correlation Between Gaze and Mouse	96
5.5	Summary	100
6	Conclusion and Future Work	101
6.1	Conclusion	101
6.2	Future Work	103
	References	107
	Publications	121

List of Figures

1.1	Applications using eye tracking	2
1.2	Eye-based typing system [1]	3
1.3	The gaze trace of “Hello”	4
2.1	Human eye structure [2]	10
2.2	The curvature of the lens [3]	10
2.3	The relative vision acuity across the visual field [4]. Left: the fovea centralis has the highest acuity, and the periphery vision dramatically falls to 50%. Right: The image vision is radially getting blurred as the acuity decreases.	11
2.4	The visualization of fixations and saccades. Left: the circles indicate fixations, where the size of the circle shows how long the fixation stayed (dwell time), the numbers show order of the fixations; Right: the colored areas represent the relative dwell time across the paint, where red highlights areas with the most fixation activity.	12
2.5	Appearance of the eyes across race.	15
2.6	The direction of pixel intensity around the pupil [5].	16
2.7	Eye movement tracking techniques [6, 7, 8].	18
2.8	Eye Gaze communication board [9].	19
2.9	Gaze-based text entry using discrete gaze gesturing and continuous pointing gesturing [10, 11].	20

2.10	Eye typing systems with alternative keyboard layout and word prediction [12, 13].	22
2.11	Gaze determination with pupil position and head pose	25
2.12	Feature-based gaze estimation. (a) shows the 3D eye structure. The angular offset between visual axis and optical axis is called kappa (k), which is a person-dependent constant. (b) shows the relation between pupil center and the cornea reflection	26
2.13	The manifold learning using locally linear embedding. (a) and (b) illustrate three-dimensional data sampled from a two-dimensional manifold. (c) illustrates the data represented in two-dimensional space [14].	27
3.1	The paradigm of the eye-typing system	30
3.2	The Midas Touch Problem	31
3.3	Low accuracy or Drifting	32
3.4	The gaze heat map of “hello”	33
3.5	Keyboard coordinate	33
3.6	The paradigm of our system	34
3.7	Letter sequence generation	35
3.8	The converting cost calculation	39
3.9	State transition	41
3.10	Preliminary results on neighbour weight	43
3.11	Transition Mapping	44
3.12	Data structure	47
3.13	The comparison of execution time per word	49
3.14	The top-5 rate in extra-letter error	51
3.15	Average position of words in the candidate list	52
3.16	The top-5 rate in two errors	54

3.17	Participant performing experiments	55
3.18	The results of actual users	56
3.19	The average top-5 rate with error bars	56
4.1	The relative relation of images and points. left-hand side indicates the eye-area images that looking at nine different positions, while right-hand side indicates the corresponding points in the high-dimensional space. . .	64
4.2	The mapping from high-dimensional space to gaze coordinate plane. . . .	64
4.3	The rough eye-region image extraction	65
4.4	Eye-appearance feature vector generation.	66
4.5	The 26-letter keyboard layout	67
4.6	The PCA projection. (a) shows the percentage of eigenvalues. (b) plots the 2-D-surface manifold in 3-D space formed by the first three principal components. (c) plots image points distribution with the corresponding key labels in 2-D space formed by the first two principal components. . .	68
4.7	The procedures of the appearance-based approach.	70
4.8	The experiment setup with a desktop computer which displayed 35 cross- hair markers.	71
4.9	Individual images and averaging images	72
4.10	Evaluation results	73
4.11	Cropping operations. The process of two-time horizontal, vertical, and full cropping operations.	75
4.12	Evaluation results. The error of horizontal, vertical, and full cropping operations in different cropping times.	76
4.13	The procedure of eye typing using webcam.	77
4.14	Recognition results.	78

4.15	Estimated gaze points of appearance-based method and eye tracker. they show the estimated points of typing ten words, “guest”, “enterprise”, “seal”, “hike”, “until”, “account”, “normally”, “opportunity”, “charge”, “drag”. The blue dots denote points estimated by the appearance-based method, and the red dots denote points by eye tracker. The upper right box shows top-5 words recommended using estimated points of the method, and the lower right box shows top-5 words using points of the eye tracker.	80
4.16	Eye-appearance manifolds in 3D space.	83
4.17	The effective area	84
4.18	The displacement of head	86
5.1	Distribution of head pose and mouse click	92
5.2	The evaluation results. “Left”, “Right” indicates only left or right eye images is used; “Average” indicates using average estimated points of both left and right eye images; “Concatenate” indicates our proposed method; “Minus”/“Plus” indicates left eye-image vector minus or plus right eye-image vector.	93
5.3	Eye area cropping and the image effect applying the minus/plus operation	94
5.4	The degree of head pose variation	96
5.5	The paradigm of data collection.	97
5.6	The combination of mouse events. LBD is left button down; LBU is left button up; RBD is right button down; RBU is right button up; M is movement.	98
5.7	Some screenshot samples. The blue dot and red circle (the innermost circle) indicate the actual gaze point and mouse click position, respectively; The green and cyan circles show the accuracy scope of the eye tracker. .	99
5.8	The alignment percentage of click events.	99

6.1	An overview of static model	104
6.2	An overview of dynamic model	105
6.3	An overview of system deployment	105

List of Tables

3.1	States Table	42
3.2	Example Table	42
3.3	The summary of four algorithms	61

Chapter 1

Introduction

1.1 Background and Motivations

In modern society, being healthy means more than just not being sick. The quality of health is also influenced by the experience of being active, and having a sense of connection with the community, in which communication is an important tool to staying engaged. With the development of information technology, access to and communication via internet is becoming popular. However, some people with severe motor disabilities have difficulty in communicating using their hands for typing, e.g. Motor neuro disease (MND), such as Locked-in Syndrome or Amyotrophic Lateral Sclerosis, where there are nearly 120,000 new cases diagnosed worldwide annually [15]. Due to complete paralysis of nearly all voluntary muscles in the body, people with these conditions may not be able to move or even use speech properly, and control of the eyes might be the only means to communicate [16]. If their voluntary eye movement can be coded to messages, which is often not affected by the paralysis, it could provide a reliable method of communication.

In our daily life, more than eighty percent of all information received comes from the eyes [17]. Eyes and their movements play an important role in expressing human being's desires, cognitive processes, emotion states, and interpersonal relations [18]. Saccade and fixation are two common status of eyes. Saccade is the rapid eye movement from one location to another, and fixation is stationary positioning over one target. Intuitively, fixation indicates the intent of a person, and has been applied to various applications, such as marketing, advertisement, and human behaviour analysis (figure 1.1). With advances in both eye-tracking technology and computer systems, eye-tracking research

Figure 1.1: Applications using eye tracking

gradually focused on human-computer interaction rather than just cognitive analysis. Eye movements are encoded into commands for controlling the systems, which can now be employed to aid disable people to interact with computer systems. One of outstanding achievements is that eye-tracking devices have helped, in limited manner, some disable people to enter text into their computer by controlling a typing system (figure 1.2), i.e. eye typing [20, 12, 21, 11], which provides an effective and reliable means of communication.

Early eye-typing systems required the person to select each individual letters by fixating at them for a dwell time, which allowed users to type only two or three words per minute, so that speed of text entry is a crucial limitation to practical communication [22, 23, 24]. Most eye typing systems employ a virtual keyboard on the screen, with the gaze point of the user on the virtual keyboard being monitored by an eye tracker. Based on the analysis of gaze behavior, the typing system detects which letters the user is fixating on, and infers whether the letter is actually what the user wants to type. With such an approach, early eye-typing systems had two main reasons for slow text entry:

- (i) *Selection method*; users need to fixate a key to activate it for a minimum interval called dwell time, i.e. if a user wants to select a letter, they have to fixate at the letter on the keyboard for some time (typically 200-600 ms [25]). The dwell time selection method limits the speed.
- (ii) *Text entry method*; early eye-typing systems only supported complete, individual character input, i.e. if a user wanted to input “hello”, they had to select five

Figure 1.2: Eye-based typing system [1]

separate, individual letters. The single character entry limits the maximum entry speed.

In order to improve the speed of text entry, researchers have developed a number of improvements. To increase typing speed when selecting individual characters, the dwell time can be reduced or dynamically adjusted [20], or the keyboard can be modified from the standard QWERTY layout to more efficient forms [12, 21, 11, 26]. However, the user might get rapidly tired when attempting to fixate for every letter of every word. A complementary approach is to predict the next letter(s) and/or words based on previous entries when typing [21, 22], i.e. the users do not have to input all letters of a certain word, and the system would help to predict and complete the intended word. Although speed of text entry is increased, it is still limited by dwell time in which users need to fixate at the correct keys. Also, the user could get tired in a shorter time span when attempting to fixate every key entry. Therefore, it is necessary to innovate and develop a new text entry system so that the user can be relieved from individual key typing. Moreover, if the system allows users to focus on the entire word typing instead of individual letter typing, the speed of text entry would improve.

Recently, touch screen has enabled users to enter text by just dragging their fingers across the screen keyboard, e.g. Swype¹ and SwiftKey² entry system for smart phones.

¹<http://www.swype.com/>

²<http://swiftkey.com/en/>

Figure 1.3: The gaze trace of “Hello”

Inspired by the existing finger swipe typing techniques, the concept of “dwell-free” typing has been proposed [27]. Like the swipe-typing systems, dwell-free typing allows the user to type a word by just gazing at letters sequentially rather than pausing at each individual letter for at least the dwell time. The basic idea is that such a system should be able to recognize the intended word based on the sequential letter detected and extracted from user’s gaze trace. Since each word consists of an ordered sequence of letters, the word can be often be recognized based on the sequential letters detected and extracted from user’s gaze trace. For example, a gaze trace of “Hello” in the system as shown in Figure 1.3. The user will gaze at “H”, and then saccade through “H” $\rightarrow$ “E” $\rightarrow$ “L” $\rightarrow$ “O”. Finally, according to the letter sequence analysis, the system is able to recognize the intended word as “hello”. Note that during the typing period, the user does not need to select the individual letters with dwell time. Such an approach allows users to focus on the entire word while typing instead of the individual letters.

However, without the constraint of dwell time, all letters that the user is looking at will be recorded due to “The Midas Touch” problem. The input sequence contains a lot of redundant and irrelevant letters, which causes three common typing errors in practice, 1). *the extra-letter error*, the input sequence has extra irrelevant letters besides intended letters; 2). *the neighbor-letter error*, some input letters are replaced by their neighboring letters; 3). *the missing-letter error*, some intended letters are totally missing. Without addressing these typing errors, dwell-free typing is vulnerable and infeasible in practice, although it has turned out to be theoretically much faster than existing dwell-based text entry techniques [27].

In addition, the current proposed eye-typing systems always require a commercial eye-tracking device that is providing the estimated gaze point (what the user is looking

at), which is somehow burdensome. As the webcam is becoming a standard component in computers, replacing the eye tracker with a webcam would simplify the setup of eye-typing equipment, and also facilitate the prevalence of eye-tracking applications. Moreover, for generic people who do not have any motor impairment, head movement is a critical issue, where only modelling eye movement will cause the poor performance of gaze estimation. Thus, this problem should be also considered in developing a practical gaze estimation system.

1.2 Research Objectives

This research explores reliable assistive techniques in the eye tracking area, which benefits physically challenged people, as well as the general population. Specially, we would like to propose a reliable typing system for improving the communication ability of target people who can control only eye movement. In particular, the eye-typing system with robust recognition algorithms and gaze-estimation system using a standard webcam are presented in this thesis. The research objectives are detailed as follows:

- Despite the success and surge in interests on the concept of dwell-free typing system, recently proposed dwell-free systems are vulnerable to the common typing errors as mentioned above, i.e. the extra-letter error, the neighbor-letter error, the missing-letter error. The dwell-free system has turned out to be theoretically much faster than existing dwell-based text entry techniques, but if the user has to spend extra time rectifying the wrong words, the entry speed of the system might become much slower, at the same time, the usability is also decreasing. Thus, one objective of this research is to propose a robust dwell-free system that is resilient to the typing errors.
- As discussed, in existing eye-typing systems, a commercial eye-tracking device is indispensable for estimating eye gaze point of the user. With the development of computer vision technology, eye gaze estimation based on original face images might be feasible. Webcam is becoming a standard component in most computers, especially mobile devices. Thus, another objective of this research is to study

the gaze estimation techniques using face images captured by webcam. Besides proposing the eye-typing system for physically challenged people, the objective also explores a broader range of applications for generic users.

1.3 Core Contributions

In this thesis, we have explored three projects. First of all, a dwell-free eye-based typing system along with two robust recognition algorithms is proposed for effective communication of people with severe motor disabilities. Subsequently, feasibility of eye typing using an off-the-shelf webcam is investigated, and some critical issues of appearance-based gaze estimation approach are studied. Finally, a new reconstruction method of gaze estimation is proposed for compensating the impact of head pose variation. The detailed contributions of the thesis are summed up as follows:

- **Robust Recognition Approaches in Eye-Based Dwell-Free Typing**

In this study, the dwell-free typing system, GazeTry, is developed to allow the user to enter text by using eye gaze, where two robust recognition algorithms respectively are proposed for inferring the words which the user intends to type. Instead of requiring the user to gaze at each individual letter, the proposed algorithms determine the intended word based on the pattern of sequential letters of gaze trace, which are more robust to missing letters and when a neighboring letter on the keyboard is incorrectly selected. Meanwhile, an effective tree structure is proposed to accelerate the recommendation. Simulation and experiment results suggest that the algorithms has better accuracy and more resilience to common text entry errors than other currently proposed dwell-free systems.

- **Assistive Text Entry Using Gaze with an Off-the-shelf Webcam**

In this work, a prototype of eye-typing system using an off-the-shelf webcam without the extra eye tracker is proposed, where the appearance-based method shows the feasibility of being used to estimated gaze points of the person. Two critical issues of the appearance-based method are also investigated in this work. Since the captured eye images at calibration session contain invisible artifacts that could

cause large estimation error in subsequent gaze estimation, we propose an averaging filter to multiple captured images, so that the performance in terms of estimation accuracy improves through reducing the impact of artifacts. We study the effective features of the eye area that are highly related to the determination of gaze points, which achieves dimensionality reduction of the appearance space with maintaining the estimation accuracy at decreasing computational cost and time complexity. The performance evaluation show that eye typing with webcam using the proposed method is comparable to the eye tracker under a small degree of head movement.

- **Generic Gaze Estimation with Natural Head Movement**

This work presents a novel reconstruction method for the appearance-based gaze estimation that allows inferring person’s gaze under natural head movement. We first study the locally linear combination in the respective manifolds consisting of stable left and right eye appearances. The local structure of the manifolds is destroyed when there is head movement. This is due to the destruction of the intrinsic relations between the two eyes (left and right) when we do locally linear combinations. We then introduce a new combination of both eye appearances, which aligns the two appearance spaces of both eyes into the same local structure, and maintains the relation embedding into the reconstruction of the training stage. In addition, we also analyze the impact of degree of head pose variation. Through comparison with other well known approaches, we show that the proposed method achieves an optimal performance with head pose variation.

1.4 Thesis Organization

The remainder of this thesis is organized as follows:

- Chapter 2, we present general background knowledge of human eye and eye movement, and introduce related works on eye detection and eye tracking technology. Besides, eye tracking has been applied in various applications, especially being significant for people with severely physical impairment, and the assistive communication applications are also reviewed. Moreover, several gaze estimation approaches using monocular cameras are discussed in detail.

- Chapter 3, we develop an eye-based dwell-free typing system that helps people with motor disabilities enter texts by using eye gaze information. Besides, robust recognition algorithms are proposed to solve practical typing errors, which make the dwell-free typing system have better accuracy and resilience.
- Chapter 4, we investigate the feasibility of eye typing using an off-the-shelf webcam, and then propose a prototype of eye-typing system with the webcam. We show that appearance-based method is an effective method for gaze estimation.
- Chapter 5, we propose a novel reconstruction method for the appearance-based gaze estimation method that allows inferring persons's gaze under natural head movement. Besides, we also investigate the correlation between eye gaze and mouse click, and demonstrate that there is high coordination while clicking menu buttons.
- Chapter 6, we conclude the thesis and discuss several directions on future work.

Chapter 2

Literature Review

In this chapter, we first present an overview of background knowledge of human eye and eye movement. Then, several common eye detection and tracking techniques are reviewed, and we also discuss the advantages and limitations of these techniques. In addition, eye tracking has been applied in various applications, especially people with severely physical impairment, and the assistive communication applications are reviewed. Webcam is becoming a standard component in most computers, especially mobile devices. Thus, we also present several gaze estimation approaches using monocular cameras.

2.1 Basics of Human Eye

In this section, the eye structure and the mechanism of human vision are introduced in details, followed by the common types of eye movements.

2.1.1 Eye Structure

In Gullstrand's eye model [28], the human eye is described as an approximately spherical shape as shown in figure 2.1. The visible parts of the eye in the eye socket include sclera, iris, cornea, pupil, which mainly represent the appearance of the eye. The sclera is commonly known as the white of the eye and forms the outer coating of the eyeball. The iris is colored part that has a thin and circular structure in the eye, and controlling the size and diameter of the pupil. The cornea, a protective transparent membrane, is the eyes outermost layer. It controls and focuses the incoming light through refraction.

The pupil is a dark hole located in the center of the eye (iris) that controls the amount of light rays entering to the retina by changing its size [29].

Figure 2.1: Human eye structure [2]

Behind the visible parts, there is the lens that has a transparent ellipsoid, biconvex structure. As the lens is elastic, it can change shape (focal distance of the eye) to help to refract the incoming light to be focused on the retina, where the curvature is getting smaller to focus distant objects, and larger to near objects (figure 2.2). The middle of the eye is filled with transparent vitreous body. At the back of the eye, there is the retina, the light-sensitive tissue, which covers about 65 percent of its interior surface. The retina receives the light rays passing through the lens, and then converts light to nerve impulses that pass through the optic nerve to the brain, where the visual images are formed.

Figure 2.2: The curvature of the lens [3]

However, visual acuity is not distributed uniformly at the retina of the eye, otherwise human's eye movement does not reveal meaningful information [25]. Instead, there is

Figure 2.3: The relative vision acuity across the visual field [4]. Left: the fovea centralis has the highest acuity, and the periphery vision dramatically falls to 50%. Right: The image vision is radially getting blurred as the acuity decreases.

a highly non-uniform distribution of receptors across the retina, in which the higher density area has the higher visual acuity. The highest density of receptors is a small dimple located near the center of the retina (macula), called fovea, which provides much higher resolution vision than the surrounding areas. The fovea size is less than one percent of total size of the retina (around one degree of the visual field [30]), while it carries half of visual information in the optic nerve [31]. Outside the fovea, acuity of the vision drops to 15% to 50% of the fovea, which is not enough to form the clear visual image (figure 2.3). Thus, for highly clear vision requirement of context, e.g. reading, the area of interest (AOI) should be viewed within the fovea area.

The fovea is not exactly within the optical axis of the eye. The optical axis is the most direct line connecting the center of the pupil and the center of the eyeball, i.e. the line of gaze (LoG). The line defined by the fovea and the center of pupil is known as the visual axis, i.e. the line of sight (LoS), which is determining the real visual attention of a person. Thus, the point of regard, the point at which the eye is looking, is determined as the intersection of the visual axis and the area of interest. The angular offset between the optical axis and the visual axis is called $kappa(k)$, which is a person-dependent constant around 4-5 degrees horizontally, and 1.5 degrees vertically [32, 33].

2.1.2 Eye Movement

Visual acuity decreases rapidly outside the center of the visual field (fovea), and for the clear view of the area of interest, people need to move the eyeball to make the visual

Figure 2.4: The visualization of fixations and saccades. Left: the circles indicate fixations, where the size of the circle shows how long the fixation stayed (dwell time), the numbers show order of the fixations; Right: the colored areas represent the relative dwell time across the paint, where red highlights areas with the most fixation activity.

image directly appear in the fovea area. Thus, such eye movement usually reflects the people's intention. In general, eye movement consists of the voluntary and involuntary eye movement, helping in acquiring, fixating and tracking the target of interest. Compared with the involuntary movement of the eyes, the voluntary movement of the eyes is more informative, which includes the four basic types of movements, saccade, smooth pursuit, vergence shift, and vestibulo-ocular movement [34, 35]. *Saccade* is rapid, ballistic movement of eyes from one location to another for quickly localizing the target area of interest, and the duration is around 30-120 milliseconds. *Smooth pursuit* is much slower movement of the eyes to keep tracking a moving target within the fovea area. *Vergence shift* is aligning the fovea of the both eyes to increase the depth perception and sharpen the visual image of target. Unlike saccade and smooth pursuit, two eyes of vergence shift move in different direction (disconjugate eye movement). *Vestibulo-ocular movement* is keeping the original direction of the eyes so that the image of the target of interest is still on the fovea as head pose varies, which helps to compensate for the impact of head movement.

In the eye-tracking area, the eye movement is simply described as two dominant statuses, *saccades* and *fixations*, because the model of the two statues is efficient enough

to capture people’s attention (figure 2.4). This thesis also focus on the two-stages model of the eye movement:

- **Fixations** Fixations are the periods when the eyes essentially stop moving, keeping the central fovea vision to the point of interest, where a fixation typically stays between 200 and 600ms. During a fixation, the eyes do not hold completely stationary, and there are still minor movements [36, 37, 38, 39], 1). microsaccades, small jerk-like involuntary eye movements which usually occur during a longer fixation (several seconds); 2). tremor, physiological nystagmus with high-frequency (around 90 Hz) and low amplitude (150-2500nm); 3). drifts, slow motions of the eye occur simultaneously with tremor, which compensate for maintaining accurate visual image during microsaccades. Because fixations reflect what the eyes are looking at, they are the most popular feature that researchers focus on to make inferences on people’s cognitive processes or states of interest in probing.
- **Saccades**. As stated above, saccades are a series of sudden jumps from one point of interest to another, which are typically covering 15 to 20 degrees of visual angle, and followed by a fixation. They can be triggered voluntarily and reflexively whenever the eyes are open. Regarding the different intended goals, there are four categories of saccades [40], visually guided saccade, antisaccade, memory guided saccade, and predictive saccade. In *visually guided saccade*, the eyes move to a visual stimulus presented in the periphery. In *antisaccade*, the eyes move away from the visual onset location to its mirror. In *memory guided saccade*, the eyes move to the memorized location. In *predictive saccade*, the eyes move to the predicted location of a moving visual stimulus, which is similar to smooth pursuit movements. In the visual system, the visual information during the saccadic movement is not received in order to maintain perceptual stability, which is known as saccadic suppression [41]. In addition, There is a special type of saccades, the rapid eye movements during a phase of sleep, which is out of the scope of this thesis.

2.2 Eye Tracking

In general, the term “eye tracking” has two definitions, tracking where the eyes are and tracking what the eyes are looking at. They refer to two research areas respectively,

eye detection/localization and *gaze estimation*. 1). Eye detection deals with detecting the presence of eyes, accurately interpreting eye positions in the images, or tracking the eyes from frame to frame for video images, while eye localization refers to individual eye feature areas, i.e. to localize eyebrows, eyelids, eye corners, sclera, iris and pupils [42, 43, 44, 45, 46, 5]. 2). Instead of only localizing where the eyes are in face images, gaze estimation is determining what the eyes are focusing on using detected eyes in images or videos [47, 48, 49, 50].

Nowadays the two terms “eye tracking” and “gaze tracking” are inter-changeable, where gaze tracking is a process of consecutive gaze estimations from frame to frame in a video or real time. In order to disambiguate the two terms, in the remainder of this thesis, “eye tracking” also has the same meaning with gaze tracking, i.e. tracking gaze paths, and the term “eye detection” represents tracking the location of the eyes and eye features in an image. In this section, we mainly review eye detection techniques, as well as commercial eye trackers, the details of gaze estimation using the video-based camera is reviewed in Section 2.4.

2.2.1 Eye Detection

The eyes have distinct characteristics from other features of the face. In eye detection, it is crucial to find a model of the eyes that is robust to complicated environment, under various eye appearances and illumination, and even with occlusion. As stated in Section 2.1.1, the basic structure of the eyes is the same, and the appearance of the anterior of eye socket area also shares commonalities even across race (figure 2.5). 1). The two eyes are symmetrical located at the upper part of the face; 2). the areas of the eyes including eyebrows are darker than periphery; 3). inside the eye socket, the dark part (iris) is surrounded by the white part (sclera), and the center of iris (pupil) is the darkest; 4). the shape of eye socket is approximating the ellipse, and the shape of both the iris and the pupil is circle. In terms of above geometric and photometric properties of the eyes, eye detection techniques are categorised into three main methods, shape-based, feature-based, and appearance-based methods.

The shape-based methods employ a prior shape model of the eye, which is usually consisting of a geometric eye model and a similarity measure. The geometric eye model

Figure 2.5: Appearance of the eyes across race.

is described by the exterior shape of the eye socket (regions between upper and lower eyelids), and the iris and pupil contours. The similarity is measured by the parameters of the geometric model, including the parameters of rigid transformation (rotation, translation, reflection) and nonrigid transformation (deformation). In the simple elliptical shape model, edge detection techniques are widely used to extract the boundaries of eye features with the threshold of image intensity [51, 52, 53, 54], and followed by circle Hough transform or circle/ellipse fitting to localize the iris and the pupil [55, 45, 56, 57]. Since they rely on the intensity threshold to separate the iris/pupil region, they might fail on the low-contrast images. Besides the simple elliptical shape model, there are also complex shape models that interpret the shape of eyelids as two parabolas using deformable template-based methods [58, 59, 60, 61]. The methods usually have more generic and accurate performance, but suffer from some limitation, e.g. time consuming cost, high requirement of initialization.

The feature-based methods utilize well-known eye properties to detect the distinctive features around the eyes. The center of the iris or pupil is commonly used to represent the position of the eyes. The popular techniques are utilizing the spatial information of the iris and the pupil, such as the image gradient field [62, 63], isophote curvature [5, 64, 65]. The gradient-based methods use the eye property that the pupil locates at the center of the darkest area (iris) surrounded by the white area (sclera), and all gradient directions of the pupil center are outward (figure 2.6), and the isophote-curvature-based methods use the edge orientation of the image intensity to gain invariance to linear lighting changes. The pupil center can also be localized by using the symmetry property of the eyes, i.e. looking for the pattern of two dark areas satisfying predefined anthropometric model [66, 67], but the methods are suffering from the false positive for other dark regions (e.g. eyebrows, shadows) due to the limitation of the skin-color model.

Figure 2.6: The direction of pixel intensity around the pupil [5].

The appearance-based methods detect the entire eye directly using the distribution of intensity and surroundings [68, 69, 44]. The performance of the methods is highly dependent on the variety of the image templates, and thus they require a large number of training samples covering various eye templates under different conditions [70, 71]. Being holistic methods, the model trained by the appearance-based methods is usually independent of the specific object. The well-know Viola and Jones face detector can be adapted to eye detection easily by learning informative Haar-like features of the eye areas [72, 73]. Based on the Haar-like features, the method will generate a strong classifier consisting of multiple weak classifiers in cascades through an adaptive boosting (Adaboost). The major limitation of the appearance-based methods is the size of training set, but nowadays, the development of deep learning techniques improve its performance on face detection as well as eye detection [74, 75, 76]. The appearance-based methods used in gaze estimation will be reviewed in Section 2.4.

2.2.2 Eye Movement Tracking

A device used to measure eye movement is commonly known as *eye tracker*, which typically has two categories in terms of measurement, intrusive and non-intrusive techniques. The intrusive techniques include Electro-OculoGraphy (EOG), and sclera contact lens. The non-intrusive techniques include Photo-OculoGraphy(POG), and video-based pupil-corneal reflection [7].

Electro-OculoGraphy (EOG) is measuring skin's electric potentials between the front and the back of the eyes, which indicate eye movement (figure 2.7a). In general, several electrodes are placed around the eyes. The potential is highly related to the position of eye relative to the head, where the range of the recorded potentials is from

15 to 200 μV , and around 20 μV represents one degree of eye movement [77, 78]. The accuracy of EOG is about 2° . Due to the technical simplicity, the EOG technique has been widely used for recording eye movement for clinical applications since half a century before, but the setup requirement is not proper for daily use [2, 79, 7].

Sclera Contact Lens, or search coil, is using a small coil embedded into a contact lens that tightly adhere to the eye, and magnets placed around the eye generate an magnetic field (figure 2.7b) [80, 35]. When moving the eye, i.e. the electrical conductor (the coil) moving through the magnetic field, electric currents are produced in the coil by electromagnetic induction. The direction and degree of the eye movement determine the parameters (polarity and amplitude) of the produced electric currents. Thus, the status of eye movement can be computed through the parameters. The technique is the most accurate method for measuring eye movement (approximately 0.08°), but it is also the most intrusive method [2].

Photo-Oculography (POG), or video-oculography, is a non-intrusive video-based technique of measuring the eye features under eye movement [7, 78]. As shown in (figure 2.7c) The technique usually applies the head-mounted device with miniature cameras to closely track the shape of the pupil and the limbus (the boundary between the iris and the sclera). Measurement of these ocular features do not provide point of regard (gaze point), but only the position (horizontal, vertical and torsional) of the features of the eye movement [81, 82]. Thus, the technique is mainly applied into medical diagnosis applications.

Video-Based Pupil-Corneal Reflection is typically using infra-red light to generate the corneal reflection (the first Purkinje image), and tracking the eye movement through reflection changes relative to the location of the pupil center [83]. In a more accurate method, dual-Purkinje employs the first Purkinje image and the fourth Purkinje image (reflection at the back of the lens) to compute translation and rotation eye movement respectively [84]. The pupil-corneal reflection technique does not require the close attachment with the eye, and it can be table-mounted, which is widely used in most existing commercial eye trackers (figure 2.7d).

Figure 2.7: Eye movement tracking techniques [6, 7, 8].

2.3 Assistive Applications of Eye Tracking

Research on eye tracking dates back to the late 18th century when people's saccadic eye movements were investigated in a reading task with the first eye tracker [85, 86]. For a long time, eye-tracking research was mainly focused on cognitive analysis [87, 88, 89, 90], exploring what eye movements revealed about perceptual and cognitive processes [91, 92, 18]. More recently, with advances in both eye tracking technology and computer systems, eye tracking research has gradually focused on human-computer interactions. Eye movement can be an input source to control electronic devices and applications where eye trackers are used as an interaction device [93, 94]. This role of eye tracking started to become significant for people with physical impairment [95, 96, 97, 98], i.e. assistive technology, and help them conduct a more efficient communication [99, 100]. Among communication techniques, eye typing is a very promising area [101], and achieving text entry through the focus of the gaze can now provide a means of communication for people who are only capable of moving their eyes [102].

2.3.1 Interaction and Communication

In interactive applications of eye tracking, eye movements typically have the similar functions with mouse movements in some ways, where eye gaze is taken as a mouse pointer [103]. Besides two basic statues of eye movements (fixations and saccades), there are other three important metrics that are widely used in eye-tracking applications [104], gaze duration, area of interest, and scan path. Gaze duration is cumulative time duration

Figure 2.8: Eye Gaze communication board [9].

of consecutive fixations on a small area. Depending on the area size, gaze duration might also contain small durations of short saccades. Area of interest is the areas with larger gaze duration which indicate the gaze interests. Scan path is a sequence of fixations in time order, that is connected by saccades.

In human-computer interaction, *pointing* and *selection* are two essential actions (events) for successful activation of an intended target [105], i.e. activate the target by locating first and then confirming. If the eye movement is emulating the manner of a mouse, the only pointing action is registered by the eye gaze, but there is no natural counterpart of the button click, so the selection action is difficult to be identified due to the Midas Touch problem [106], where not all foveated positions reflect people's intention. In [106], several potential activities of the eyes have been investigated, such as eye blinks, gaze duration. Although eye blinks are easy to determine the selection action, they are suffering three main limitations, 1). they are ambiguous with natural eye blinks; 2). eye gaze respond disappears during eye blinks; 3). they might interrupt people's concentration on the target. Due to the the critical drawbacks, the eye blink is rejected to be a signal of selection action. Thus, gaze duration is usually taken as the selection action [107], which is well known as dwell-time selection. The threshold setting of dwell time is able to filter unintended eye gaze. During the dwell time, people always keep focusing on the area of interest without extra effort, e.g. they do not think of blinking.

Early eye-based communication systems usually require manual assistance, where both the users and their communication partner need to be actively involved in the

Figure 2.9: Gaze-based text entry using discrete gaze gesturing and continuous pointing gesturing [10, 11].

interaction, which is also called low-tech communication without involving electricity and electronics [108]. Different eye gestures can be defined as indications of understanding and agreement, e.g. “look up” and “look down” can represent the meanings of “yes” or “no”. These simple commands can be easily adapted to more meaningful communication. For example, the well-known E-Tran frame allows the user to gaze at the letter, symbol, or word that they want to say. According to the proficiency level of the user, each corner can either place one letter/symbol or hierarchical group of letters (figure 2.8). Besides eye gestures, eye blinks are also used to delivery the indication under the help of a partner. The famous example is Jean-Dominique Bauby, the author of *The Diving Bell and the Butterfly* [109]. He wrote the book by blinking his left eyelid to confirm the letter that he heard, while a partner repeatedly recited the frequency-ordered alphabet using partner-assisted scanning.

However, such manual eye-based communication systems are not efficient and convenient. A well-trained partner is required to aid their communication, and the predefined functions (contents) are not always satisfying the user’s needs. Thus, computer-based eye communication systems have been proposed [102], in which the manual communication board is replaced by a computer system with a commercial eye tracker [23, 24]. The systems employ a virtual keyboard on the screen, with the gaze point of the user on the virtual keyboard being monitored by the eye tracker (figure 1.2). Dwell-time selection is used to activate the user’s intended letters, where the user fixates the letter that he

wants to enter for a predefined time duration (threshold), and once the gaze duration exceeds the threshold, the system will take the gazed content as the user's input. The computer-based eye communication system relying on direct gaze pointing is also known as *eye typing*. According to [110, 21], besides eye typing, there are another two types of gaze-based text entry methods, discrete gaze gesturing [10, 111], and continuous pointing gesturing [11, 112]. Discrete gaze gesturing is based on the shape of scan path like "eye strokes", and the gesture is recognized by the relative changes of gaze directions (figure 2.9a). Continuous pointing gesturing is using pursuit movements of the eyes, where the users keep using eye gaze to point at the moving letters (figure 2.9b). Although discrete gaze gesturing and continuous pointing gesturing have certain advantages, i.e. insensitive to the accuracy and calibration drifting problems [113], eye typing with a virtual keyboard is still the most common user interface of text entry, and provides an effective and reliable means of communication for the severely handicapped people.

2.3.2 Eye Typing

Eye typing has been a useful communication modality for physically challenged people, but is also problematic. Although direct gaze pointing is usually similar to a mouse or other pointing modality, gaze pointing is not as precise as with a mouse due to the limitation of fovea accuracy. In addition, the selection action using gaze duration is slower than the button click of a mouse. Thus, dwell-time selection method requires the user to select each individual letters by fixating at them for a while (typically 200-600 ms [25]), which is limiting the speed of text entry [114, 22, 23, 24]. The early eye-typing systems allowed users to type only two or three words per minute, which is impractical for communication. Besides the selection method, complement a word using single letter entry also limits the speed of text entry. In order to increase the speed of text entry, a lot of researcher have worked on improving the above two methods, *selection method* and *text entry method*. For the selection method, there are two main solutions, one is reducing dwell time or setting adjustable dwell time according to the proficiency of the user [20], and the other is to modify the keyboard layout from the standard QWERTY layout to more customized forms [12, 21, 26]. For the text entry method, the main solution is to

Figure 2.10: Eye typing systems with alternative keyboard layout and word prediction [12, 13].

integrate next letter(s) and/or word prediction based on based on previous entries when typing [21, 22].

However, reducing dwell time will cause more typing mistakes, and the user also needs to make extra efforts to focus on dwell time. Although modified keyboard layouts have some advantages on the novices, e.g. alphabetical layout (figure 2.10a), gaze pointing with the QWERTY layout still has the fastest speed of text entry after getting familiar with it [115]. In addition, the user might also get rapidly tired when attempting to fixate for every correct letter of every word. Due to the reduced number of keystrokes, predicting next letters or word based on only a few letters is potential to speed up the text entry, but the it causes additional cost of scanning the prediction list for the user (figure 2.10b). In the finger typing system, people can use eye gaze to receive visual feedbacks while fingers are typing in parallel, but in the eye typing system, eye gaze has been used as controller to type already. The visual feedback of the prediction list will interrupt the typing processing, and adds extra searching time. Thus, the actual benefit of next letters/word prediction is smaller than expectation [116], e.g. the overall communication rate might not increase.

Recently, the concept of “dwell-free” typing has been proposed [27]. In dwell-free typing, there are three main features, using QWERTY keyboard layout, not using dwell-time selection, and not interrupting the user’s attention. Thus, like finger-swipe typing, it allows the user to type a word by just gazing at letters sequentially, instead of pausing at each individual letter for at least the dwell time. The system will recognize the word

that the user wants to enter based on the scan path. This recognition method is different from eye discrete gesturing, where it mainly focus on the sequentially gazed letters over the scan path rather than the scan path itself (e.g. shape). Moreover, dwell-free typing is able to allow the user to focus on the entire word entry instead of the individual letters.

In [27], the researchers investigated the potential of dwell-free eye typing with a QWERTY-based keyboard layout. They demonstrated that dwell-free eye typing is theoretically much faster than existing eye-based text entry techniques. However, their system assumed an ideal case in which the word that the user intends to type is completely known in advance, and only accepts the input letter corresponding to the next letter in the word; if the next letter selected does not fit the known word, it intentionally ignores it. Therefore, words can be written by only accepting correct letters, even if the user looks at extra letters. While useful as a demonstration, clearly in practice it is impossible to know what the user wants to input in advance, and such noisy letter sequences with extra letters must be handled by practical dwell-free techniques in eye typing. For practical text entry, a filter-based dwell-free typing system was proposed, called *Filtaryedping* [117] [118], which is able to handle the “extra letters” error. The system was used to find all possible words formed by filtering extra letters gazed at by the user. The possible words are subsets of the letter sequence, which are sorted by their length and occurrence frequency in the dictionary. In their experiment, their approach enabled participants reach an average of 14.75WPM, comparing with *AltTyping* (10.77WPM), the fastest dwell-based eye typing tool [20, 24]. After two hours of practice, the fastest participant typed at 19.28WPM with *Filtaryedping*. Thus it appears that dwell-free systems have the potential to be much faster than the dwell-based systems in practice.

Although *Filtaryedping* is able to handle the extra letters error, it cannot handle missing letters errors, i.e. the intended word must be a subset of the letter sequence gazed by the user. If the user omits gazing at one or two letters of the intended word, the system fails to recommend the intended word. In practice, with drift and imperfect calibration of the eye tracking apparatus, as well as possible overshoot and/or undershoot of saccades, missing letters must also need to be accounted for. In order to solve the practical issue, In this thesis, we propose a dwell-free system, *GazeTry*, with two recognition algorithms

respectively which are more robust to missing letters and when a neighboring letter on the keyboard is incorrectly selected.

2.4 Eye Gaze Estimation

As discussed in Section 2.2.2, rather than only localizing where the eyes are in face images, gaze estimation is the process of determining what a person is looking at using detected eyes in images or videos [47], and gaze tracking is a process of consecutive gaze estimations from frame to frame in a video or real time. The estimation result can be either a 3-D gaze direction, or a 2-D gaze point/point of regard(PoR) that is the intersection of the gaze direction and the target plane.

The gaze of a person is determined by both head pose (i.e. position, orientation) and eyeball orientation. To change gaze direction, the person can move the head while keeping the eye stationary (also having vestibulo-ocular movement discussed in Section 2.1.2), or only move the eyes. For fast pointing, eye movement and head movement occur simultaneously, where the head is moving to a comfortable position before rotating the eyeball. Head pose determines the coarse-grained gaze area, and the eyeball orientation determines the fine-grained gaze point. If a target is within 15 degrees, a person can rotate eyes alone. If the target exceeds 20 degrees, head movement accounts for around 80% of gaze movement. Therefore, the model of gaze estimation needs to take both head pose and eye position into account, which is very challenging in the research field. In Section 2.2.2, common medical and commercial eye tracking techniques have been discussed in detail already. In this section, we mainly review gaze estimation approaches using the video-based camera, i.e. focusing on the captured face images. In general, there are two typical categories of gaze estimation, feature-based and appearance-based estimation [47].

All gaze estimation approaches will determine several typical parameters through calibration, i.e. camera-specific, geometric, human-specific, and gaze-specific parameters. Camera-specific parameters include intrinsic camera parameters, e.g. focal length. Geometric parameters include locations and orientations of the camera/monitor relative to the person's eyes. Human-specific parameters include *kappa* (angular offset) between

Figure 2.11: Gaze determination with pupil position and head pose

the optical axis and the visual axis. Gaze-specific parameters is the mapping relation between the eye and gaze points. All approaches need to estimate the above parameters explicitly or implicitly with requiring the person to look at predefined reference points on the monitor.

2.4.1 Feature-based Estimation

The feature-based gaze estimation methods are modelling the relation between local eye features and gaze points, e.g. eye corners, iris/pupil centre, and corneal reflections (glints) from the eye image. There are two subtypes in feature-based approaches, 3D-model-based (geometric) methods and interpolation-based (regression-based) methods. the 3D-model-based methods [119, 120, 121] directly estimate the centres of eyeball and iris based on a geometric eye model (figure 2.12a), and then obtain the visual axis (gaze direction) of the person. Since the angular offset of visual and optical axes is invariant to each person, gaze direction is usually determined by the estimated optical axis instead. However, in practice, it is necessary to obtain the 2-D gaze point. To avoid additional calculation of point of regard (PoR), the interpolation-based methods [122, 123, 2, 124] directly find the underlying mapping from the eye image features to the gaze coordinates. The relation of pupil and glints (cornea reflection) is the most popular and widely used for gaze estimation under active light models [125] shown as figure 2.12b. In addition, the pupil centre-eye corners vector is also regarded as a good feature with acceptable accuracy [126].

Due to explicitly determining the optical axis by modelling the eyeball structure, the 3D-model-based approaches are robust to head invariance with the highest accuracy.

Figure 2.12: Feature-based gaze estimation. (a) shows the 3D eye structure. The angular offset between visual axis and optical axis is called kappa (k), which is a person-dependent constant. (b) shows the relation between pupil center and the cornea reflection

The interpolation-based methods are looking for a mapping function between local eye features and gaze coordinates, and thus 2-D gaze points can be directly estimated, but more calibrations to estimate parameters of the mapping function, i.e. these feature vectors determining gaze variance. One main limitation is that the feature-based methods require highly-accurate detection of local features, and thus a high-resolution camera and/or the infra-red light is necessary. Although some approaches are able to use facial landmarks to estimate the head and eyeball orientation, they are prone to errors. In addition, due to the limitation of the infra-red light, the pupil center-cornea reflection vector cannot be used in outdoor environment.

2.4.2 Appearance-based Estimation

Instead of explicitly extracting local eye features, appearance-based approaches directly take the eye-area image as a high-dimensional vector, and then resemble the interpolation-based methods to train a regression function that maps it to 2-D gaze coordinates, such as multi-layer neural network [48], Gaussian process [124], and manifold learning [127, 128]. In the appearance-based estimation, the features to train a regression function are image features (pixel intensity, or other features which can describe the whole image), rather than the position of exact eye features. Since the appearance models have to implicitly estimate more hidden parameters from the images, they usually require a significant number of calibration points in global learning that is directly mapping high-dimensional image points to low-dimensional gaze points. Thus, under invariant head

Figure 2.13: The manifold learning using locally linear embedding. (a) and (b) illustrate three-dimensional data sampled from a two-dimensional manifold. (c) illustrates the data represented in two-dimensional space [14].

pose, or the static model, local learning is widely applied using appearance manifold that is a continuous set of eye-image data points in the high dimensional space.

As shown in figure 2.13, the two-dimensional manifold is embedding in three-dimensional space. The task of manifold learning is to reduce the dimensionality, i.e. extract the low-dimensional manifold from the high dimensional space, meanwhile the relative geometric position in the manifold should be maintained. The manifold formed by gaze points are in two-dimensional space, and the manifold of their corresponding eye image data is in high dimensional space (three-dimensional space in the example). The gaze estimation in manifold learning is that given a set of gaze points (low dimension) and corresponding eye-image data points (high dimension), it estimates the gaze point of a new eye-image data point.

Instead of highly accurate local eye features, appearance-based methods take low-resolution images as inputs, where an off-the-shelf webcam is sufficient which is more flexible, so that the appearance-based methods have been becoming a popular gaze estimation technique. Recently, along with the prevalence of deep learning, some research work also introduce deep neural network into gaze estimation training with millions of eye images [129, 50].

2.5 Summary

In this chapter, we thoroughly review eye-related background knowledge. The anterior parts of the eye, sclera, iris, cornea, and pupil constitute the visible appearance of the eye area. Passing through the pupil, the light of the object reflected by the cornea and the lens will strike the retina which is the light-sensitive tissue at the back of the eye, where only the fovea area has the enough visual acuity to view the object clearly. Therefore, the eyes need to move to view the object foveally, in which saccades and fixations are two dominant eye movements capturing people's attention. According to geometric and photometric properties of the eyes, eye detection techniques include shape-based, feature-based, and appearance-based methods, while eye movements can be measured by intrusive techniques (e.g. electro-oculography, sclera contact lens) and non-intrusive techniques (e.g. photo-oculography, video-based pupil-corneal reflection). With advances in both eye tracking technology and computer systems, eye tracking has been assistive technology that helps to improve the communication ability of people physical impairment, where eye typing is the most efficient and promising communication technique. To increase the speed of text entry, dwell-free typing methods have been proposed to replace dwell-time selection, and meanwhile robustness of typing methods is also a critical issue. With the development of computer vision technology, gaze estimation using face images captured by a monocular camera has been proven to be feasible with feature-based and appearance-based estimation approaches, where the appearance-based approaches have higher potential because of the simple setup without the high-resolution camera.

Chapter 3

Robust Recognition Approaches in Eye-Based Dwell-Free Typing

In this chapter, we provide an overview of the eye-based dwell-free typing system, GazeTry, with the Moving Window String Matching algorithm (MoWing) which is able to handle common eye typing errors. However, although the system out-performs previous systems in the missing-letter case, it sacrifices its performance in more common scenario of extra-letter errors. As the number of extra letters increases, its accuracy decreases dramatically. Therefore, we then propose an “intelligent” dwell-free algorithm, Longest Common States Mapping (LCMapping), which is robust to all of three common errors. Our experimental results from both simulations and field testing with users suggest that such an approach results in improved accuracy.

3.1 Dwell-free Eye-Typing System

In an eye-typing system, there are five steps, calibration, input detection, recognition, prediction, and construction. Compared with the traditional finger typing system, the eye-typing system has two unique steps, calibration and recognition. The paradigm of the eye-typing system is shown in figure 3.1. Although some calibration-free gaze estimation systems have been proposed and perform well on saliency analysis [130, 131, 119, 132, 133], the reported accuracy is relatively low compared to commercial eye trackers, thus they are less suitable for eye typing. The calibration process maps the user’s eye movement to the coordinates on the screen. This necessitates a personal calibration process with

Figure 3.1: The paradigm of the eye-typing system

the eye tracker, since each person has different eye movement characteristics. The eye tracking software models individually-specific eye movements in order to estimate gaze accurately. Since the calibration process is hardware-dependent, researchers investigating eye-typing usually assume *a priori* that calibration is acceptable, or make it more reliable through periodic re-calibration (e.g. every 10-15 minutes). Even so, small drifts in calibration can take place, so any practical pattern recognition system should be robust to this possibility [134].

The recognition step is the key step in an eye-typing system, and is the focus of our work. Compared with a finger typing system, there are additional challenges, e.g. inaccurate eye movement or gaze, as well as small calibration drift, and sometimes what the system detects is not even what the user intends to input. Therefore, the main goal of the recognition step is to correctly recognize the correct intended words based on a noisy input from the user. This process does not rely on the context and semantic analysis which can be applied in subsequent prediction and construction steps.

Due to issues at calibration and detection steps, there are two further problems that make the intended word recognition challenging in a dwell-free system:

- *The Midas Touch problem* [106]; “everywhere you look, the command will be activated”, that means you cannot look anywhere without issuing a command. Things might be selected against the user’s inclination [135].
- *The low accuracy or drift problem*; the fovea, the part of the retina with highest visual acuity, is restricted to the central 1 degree, and yet the accuracy of measured

Figure 3.3: Low accuracy or Drifting

did not solve any of above three types of errors: extra-letter error, neighbor-letter error, and missing-letter error. Recently Pedrosa and Pimentel [117, 118] proposed a filter-based dwell-free typing system, *Filtteryedping*, which solves the *Midas Touch* problem through filtering extra letters. Their approach is good at handling the extra-letter error but fails to tackle the neighbor-letter error and missing-letter error.

3.2 GazeTry System

As discussed above, the extra-letter error, neighbour-letter error, and missing-letter error are three main critical problems at the recognition step in the dwell-free typing system.

For the extra-letter error caused by *The Midas Touch problem*, if we do not rely on other devices (e.g. EEG to detect special event-related potentials [136, 94]), the only way to avoid this problem is using dwell time (dwell-time selection). However, if dwell time is used, the user has to fixate at a key for a minimum time to activate/select it, increasing the overall time taken to type a word. The potential solution is utilizing the pattern of scan path (gaze trace), which reflects the order of fixations. Through analysing the gaze trace, we can extract the letters that the user has gazed at intentionally and unintentionally, and then obtain the letter sequence. In addition, based on the duration of gazed letters, we can find the letters that are dominant for the intended word. Dominant letters are letters where the user spends more time gazing at during the word entry period. According to the features of eye gaze, although there is no fixed dwell time for selecting an intended letter, the user would gaze at an intended letter for a relatively longer time. For example, figure 3.4 shows the heat map of the gaze while typing “hello”, where certain letters have the highest intensity. The letter is called a dominant letter,

Figure 3.4: The gaze heat map of “hello”

Figure 3.5: Keyboard coordinate

because it could be a part of the intended word. Using the letter sequence and dominant letters, it is able to find words in the dictionary with high probability of being correct despite the presence of possibly extraneous letters in the sequence.

For the neighbor-letter error and missing-letter error caused by *the drift problem*, the solution is using a cost function. Once a letter is detected, its neighbors are also regarded as possible input. For example, when “S” is detected, we also take its neighbors, “A”, “W”, “D”, “X” as potential letters. However, the larger set of input letters will lead to many more candidate words, and difficult to find the intended words. In our system with the MoWing algorithm, the neighbor or missing letters will be assigned a cost, and then we use the cost to control the size of possible words. The details of cost calculation in the MoWing algorithm will be interpreted later.

In the eye-typing system, the eye tracker firstly gets the mapping from actual eye movement to its relative position on the screen through a calibration process. Then the system transforms these data into the keyboard area, and gets relative location at the keyboard. During the momentary blink period, we will assume the gaze location remain

the same. So for each time point, we can get a gaze location (coordinate x, y). The time-series data are then processed and a final sorted candidate words are obtained. Figure 3.6 shows the paradigm of our system. The system consists of three modules: letter sequence generation, string matching and candidate ranking.

Figure 3.6: The paradigm of our system

3.2.1 Letter Sequence Generation Module

In this section, we will introduce how to transform keyboard coordinates to letter sequences in our prototype system. Each gaze has a coordinate x, y on the keyboard. Given the coordinates of those letters on the keyboard, each gaze coordinate is corresponding to a nearest letter. The border coordinate of all letters is predefined as shown in Figure 3.5, and the red dotted box is the effective area of the key. If the gaze coordinate is outside the screen “keyboard” area, its nearest letter is labelled as null and will be ignored by the letter sequence generation module. During the period of a word typing, all space and backspace keys also need to be ignored. We will use a real-case example while typing “National” in our system, and the gaze trace of the word is shown in Figure 3.5.

Figure 3.7 indicates the whole steps of letter sequence generation procedure as the following:

Figure 3.7: Letter sequence generation

1). The first step is to find the nearest letter for each time point (coordinate), in which the coordinate and nearest letters are one-to-one mapping. Note that only a small part of the time-series data is shown there.

2). After the first-step transformation, the raw time-series letter sequence contains a lot of non-alphabet characters, e.g. “null”, “space”, and “backspace”. Thus the second step is filtering these characters, and obtaining a time-series letter sequence.

3). The third step is to merge consecutive same letters, and create a frequency table. The number at the right of a letter is the number of consecutive same letters, or frequency of the same letter. Noted that it is also a time-series letter sequence, and the order of different letters is still maintained, though same consecutive letters are merged.

4). The fourth step is to remove outliers. Since the average length of a saccade is around 20 - 40 ms [137, 138], and the sampling frequency of Eye tracker is 60Hz (after initial data processing, the frequency is 50Hz), there will be about 2 time-points due to the saccade caused by eye involuntary movement. We eliminate letters whose frequency is less than or equal to 2, which are outliers. Thus we have filtered some obvious extra

letters and obtained a more compressed letter frequency table, which is generated by Algorithm 1.

After the letter sequence generation module, the letter sequence and corresponding letter frequency table are generated. The letter sequence reflects the gaze trace of user, i.e. how the user gazes at the letters sequentially. The letter frequency table reflects the gaze duration of the letters, i.e. how long time the user gazes at the individual letters. If the frequency of a letter is large, it is a dominant letter as we mentioned before, and it could be a part of the intended word with a high probability. Then the two outputs of this module are input into the string matching module.

Algorithm 1 The Frequency Table Generation Algorithm

```
1: freqTable  $\leftarrow$  empty
2: currentLetter  $\leftarrow$  sequence[0]
3: freq  $\leftarrow$  1
4: i  $\leftarrow$  1
5: while i < sequence.length do
6:   nextLetter  $\leftarrow$  sequence[i]
7:   if currentLetter = nextLetter then
8:     freq  $\leftarrow$  freq + 1
9:   else
10:    if freq > 2 then
11:      freqTable  $\leftarrow$  (currentLetter, freq)
12:    end if
13:    currentLetter  $\leftarrow$  nextLetter
14:    freq  $\leftarrow$  1
15:  end if
16:  i  $\leftarrow$  i + 1
17: end while
```

3.2.2 String Matching Module

In any text entry system, the core part is to find possible words with the highest probability. If the system does not rely on customised behaviors (eg. historical typing records), the only method is to find those words which are the most similar to the input. In the eye typing system, the intended word is recognized based on what the user gazed at. After the letter sequence generation module, a letter sequence and its frequency table are

created. They reflect which letters the user sequentially gazes at, and the gaze duration of each letter. Based on the letter sequence, the system needs to recommend several possible words including the intended word. Therefore, the main task of this module is to find these words with the high similarity (low dissimilarity) to the letter sequence.

Inspired by the approximate string matching method [139], we define the dissimilarity of two strings is measured by the cost of converting one string to the other. Then we propose the **Moving Window String Matching** (MoWing) algorithm to calculate the cost of converting the letter sequence to a word (Algorithm 2).

Algorithm 2 The Moving Window String Matching Algorithm

```

1: Initialization :
2:  $i \leftarrow -1; j, cost \leftarrow 0$ 
3:  $WP1 \leftarrow \text{empty}; WP2 \leftarrow \text{word}[0]$ 
4: repeat
5:    $SP \leftarrow seq[j]$ 
6:   if  $SP = WP1$  or  $WP2$  then
7:      $cost \leftarrow cost + 0$ 
8:   else if  $SP \in neighbor(WP1 \text{ or } WP2)$  then
9:      $cost \leftarrow cost + neighborCost$ 
10:  else
11:     $cost \leftarrow cost + errorCost$ 
12:  end if
13:  if  $SP = WP1$  and  $WP2$  then
14:     $i \leftarrow i + 1$ 
15:  else if  $SP \neq WP1$  then
16:    if  $SP = WP2$  or  $SP \in neighbor(WP2)$  then
17:       $i \leftarrow i + 1$ 
18:    end if
19:  end if
20:  if  $i \neq -1$  and  $i \neq \text{word.length} - 1$  then
21:     $WP1 \leftarrow \text{word}[i]; WP2 \leftarrow \text{word}[i + 1]$ 
22:  end if
23:   $j \leftarrow j + 1$ 
24: until  $j = seq.length$ 
25:  $wordCost \leftarrow (\text{word.length} - i + 1) / \text{word.length}$ 
26:  $totalCost \leftarrow cost / seq.length + wordCost$ 

```

In the MoWing algorithm, we define two windows first, the letter sequence window (LSW , the size is 1), and the word window(WW , the size is 2). The letters of the

windows are to be matched, in which there is one letter in the sequence window (SP), and two letters in word window (the first position as $WP1$, the second position as $WP2$). In order to calculate total cost of converting the letter sequence to the word, we divide it into augmented cost of converting LSW to the WW .

In the approximate string matching method, the cost of converting a letter to the same letter is 0, and the cost of converting to a different letter is 1. But in our method we consider two different cost of converting a letter ($L1$) to a different letter ($L2$). If $L2$ is an adjacent letter (neighbor) of $L1$, the converting cost is smaller than 1 (from the preliminary study, the optimal one is 0.2), and we call the cost as a neighbor cost. Otherwise, the cost is 1, which is called error cost. In addition, we stipulate the two windows converting cost as the following:

- (1). if SP is same as either $WP1$ or $WP2$, the converting cost from LSW to WW is no cost;
- (2). if SP is a neighbor of either $WP1$ or $WP2$, the converting cost is neighbor cost;
- (3). if SP does not satisfy above two, the converting cost is error cost.

Another important part in the algorithm is moving the two windows. In the typing system, due to the extra-letter error, the letter sequence could be much longer than a word, and the words might be a subset of the sequence. Thus, we are trying to convert the letter sequence to the word, and the LSW is keeping moving forward at every loop. For the WW , the moving condition is that all possible letters in the letter sequence have been converted into the current WW , as the following:

- (1). if SP is same as both $WP1$ and $WP2$;
- (2). if SP is not same as $WP1$, but it is same as $WP2$ or a neighbor of $WP2$.

If LSW is the last letter of the letter sequence but WW is not, there is a cost (word cost) for the converting process.

We provide an example to indicate the cost calculation process (Figure 3.8).

The letter sequence and its corresponding frequency table are generated at the letter sequence generation module. If the sequence needs to be converted to the word, “national”, while the converting cost is calculated in the following manner:

Figure 3.8: The converting cost calculation

- (1). Initially, SP is the first letter of the sequence. In order to not ignore the first letter comparison, $WP1$ is set as empty and $WP2$ is 'N' which is the first character of the word. The SP is the same as $WP2$, so the converting cost from LSW to WW is 0. The WW moving condition is satisfied, in which SP is not same as $WP1$, but it is the same as $WP2$. WW is moved forward;
- (2). SP is moved to 'A', which is the same as $WP2$. There is no cost. SP is not same as $WP1$, but it is the same as $WP2$. WW is moved forward, and then $WP1$ is 'A' while $WP2$ is 'T';
- (3). SP is moved to 'I', which is not $WP1$ or $WP1$'s neighbor, and neither it is $WP2$ nor $WP2$'s neighbor. Thus the 'I' is considered as a error cost; At the frequency table, the frequency of 'I' is 13, so a total error cost of 13 added.
- (4). SP is moved to 'T', which is the same as $WP2$. There is no cost. SP is not same as $WP1$, but it is same as $WP2$. WW is moved forward.
- (5). SP is moved to 'O', which is a neighbor of $WP2$. The frequency of this 'O' is 31, so 31 neighbor costs are added. SP is not same as $WP1$, but it is a neighbor of $WP2$. WW is moved forward.
- (6). SP is moved to 'I', which is the same as $WP1$. There is no cost.

- (7). SP is moved to ‘O’, which is the same as $WP2$. There is no cost. SP is not same as $WP1$, but it is a neighbor of $WP2$. WW is moved forward.
- (8). SP is moved to ‘N’, which is the same as $WP2$. There is no cost, and WW is moved forward.
- (9). In the following condition, ‘B’ will be considered to be ‘N’'s neighbor, and ‘K’ will be ‘L’'s neighbor. After all the comparison, cost will be divided by length of sequence (including its frequency) to normalize the cost between 0 and 1. As all letters in the word have been compared, the word cost will be 0, thus the total cost will be the same as the converting cost.

3.2.3 Candidate Ranking Module

After the string matching module, we can get converting cost from the letter sequence to each word in the dictionary. Since the converting cost reflects the dissimilarity of the letter sequence and each word, it is regarded as the ranking metric. The lower the converting cost, the higher is the similarity of the sequence of letters to the word. The letter sequence is what the user input, thus the word is probably what the user want. According to the converting cost, we rank all words in the dictionary, and those words which are at the top positions have high priority to be intended words. The goal of the text entry system is that the intended word is being at the first position or at least the first page, and the user does not need to use page up or page down. In the approach, we do not rely on the word occurrence frequency in the dictionary, and only use the converting cost to sort the candidate words.

In our preliminary experiments (Section 3.4 shows the comprehensive evaluation), the proposed system with the MoWing algorithm outperforms previous systems in the missing-letter case and overall accuracy improved for expert users, but it sacrifices its performance in the more common scenario of extra-letter errors, especially for novices. As the number of extra letters increases, its accuracy decreases dramatically. The main reason is that the MoWing algorithm relies on converting cost of two moving windows, where it is easy to fall into local optimum, and it is greatly affected by a longer letter sequence when there are more extra irrelevant letters. Therefore, in order to obtain

global optimum, the new algorithm, Longest Common States Mapping (LCMapping), is proposed resulting in improved accuracy of all three types of typing errors.

3.3 Longest Common States Mapping

In the eye-typing system, the actual eye movement is mapped to its relative position (coordinate x, y) on the virtual keyboard. Through initial data processing, $\langle x, y, l \rangle$ tuples can be generated, where x and y indicate the location of the gaze and l indicates the nearest letter at the location. According to the saccade-fixation model [140] shown in figure 3.9a, the saccade is the rapid eye movement from one location to another, and the fixation is stationary pointing over one target. Intuitively, the fixations indicate the intent of the user while typing words. Therefore, the tuples of fixations are extracted, in which the saccade-fixation detection algorithm [141] is implemented.

If tuples have the same nearest letter during a period of time, they belong to the same intended fixation location, thus we assign them into the same state. The state is interpreted by a tuple $\langle l, t \rangle$, where l indicates the nearest letter at the state, and t indicates the time interval of the state. Thus, the state is consisting of two elements in the fixation location domain (l) and time domain (t) respectively. The process of typing a word is converted into the states transition shown in figure 3.9b. Since the states directly indicate the user's typing, such as which letters the user gazes at sequentially, and how

Figure 3.9: State transition

Table 3.1: States Table

	S_1	S_2	S_3	$\dots$	S_n
S'_1	a_{11}	a_{12}	a_{13}	$\dots$	a_{1n}
S'_2	a_{21}	a_{22}	a_{23}	$\dots$	a_{2n}
S'_3	a_{31}	a_{32}	a_{33}	$\dots$	a_{3n}
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$
S'_m	a_{m1}	a_{m2}	a_{m3}	$\dots$	a_{mn}

Table 3.2: Example Table

	$\langle S, 10 \rangle$	$\langle C, 20 \rangle$	$\langle X, 10 \rangle$	$\langle A, 15 \rangle$	$\langle R, 15 \rangle$
C	0	20	2	0	0
A	2	0	0	15	0
R	0	0	0	0	15

much time the user spends on each letter, these states are regarded as observable states, and the states transition is an *observable transition* which is taken as (possibly noisy) user input in the recognition step. We then need to recommend the potential words (including the intended word) through the input.

Every word in a dictionary consists of certain ordered letters. If we regard every letter inside as a state, the whole word is consisting of states with certain one-direction transition, and we define potential words as a *potential transition* while processing the input. Potential transitions are generated by words from the dictionary, so each word is corresponding to a potential transition, and each letter indicates a state in the potential transition, in which there is only the location domain, i.e. the letter itself, but no time domain like the observable transition. Consecutive same letters are represented by a state, e.g. “apple” has four states, a, p, l, e.

In the proposed method, drawing on the idea of the longest common subsequence method, our goal is to find the longest common states between *potential transition(s)* and *observable transition(s)* which reflect their mapping similarity. We take the set similarity as a simple example. Given two sets, A and B , according to Sorensen-Dice coefficient, the two sets similarity is measured by $2|A \cap B|/(|A| + |B|)$, and the goal is to find the maximum $|A \cap B|$, which is the size of the common elements. Although set elements are non-ordered and non-repeated, the basic idea is similar.

Figure 3.10: Preliminary results on neighbour weight

We use a 2-D table to describe the method. The row represents the *observable transition*, and the column represents *potential transition* as shown at Table 3.1. There are three conditions as follows to fill in the table:

- If the locations of two transitions are the same, the corresponding cell is filled with value of the time domain of the observable transition.
- if the locations are adjacent, i.e. one is a neighbour (up, down, left, right) of the other, the cell is the value multiplied by a weight (Figure 3.10 shows preliminary results on neighbour weight in which the optimal weight is around 0.4).
- Otherwise, the cell is zero.

Finally, a m-by-n matrix is generated (Algorithm 3). The generated matrix based on a real example of a word entry is shown in Table 3.2, the observable transition is “SCXAR” with the corresponding time interval, and the time interval is processed depending on sampling frequency. the potential transition is a word, “car”.

Instinctively, while the user is typing, each fixation is related to only one intended letter, but several fixations can be related to the same intended letter which is a cause of extra-letter errors. Based on this fact, we define the two transitions mapping as a many-to-one mapping shown in figure 3.11, and the method is called the longest common states

Algorithm 3 Matrix Generation Algorithm

```

1: function BuildMatrix(observable, potential)
2:    $a_{ij}$  is the  $i^{th}$  row,  $j^{th}$  column cell of Matrix
3:    $S_i$  is the  $i^{th}$  state of observable
4:    $S'_j$  is the  $j^{th}$  state of potential
5:   if  $S_i.l = S'_j.l$  then
6:      $a_{ij} \leftarrow S_i.t$ 
7:   else if  $S_i.l \in \text{neighbour}(S'_j.l)$  then
8:      $a_{ij} \leftarrow S_i.t * \text{weight}$ 
9:   else
10:     $a_{ij} \leftarrow 0$ 
11:   end if
12:   return Matrix
13: end function
    
```

mapping method (LCSMapping). As the name suggests, we are trying to find the most common states between the input stream (the observable transition) and each word (the potential transition). One state in the observable transition can only map to one state in the potential transition. Since the transition is ordered, if an observable transition S_1 is mapping S'_2 in potential transition, S_2 cannot map the S'_1 which is earlier than S'_2 . According to the above two constraints, the problem is modelled by the following mathematical formula 3.1, which is a dynamic programming problem. The solution is the longest common states, and we solve it using a recursive formula in algorithm 4.

$$\begin{aligned}
 \text{Maximize : } & \sum_j^n a_{ij}, i \in [1, 2, \dots, m] \\
 \text{Subject to : } & \forall a_{i_1 j_1}, a_{i_2 j_2}; \\
 & \text{if } j_1 < j_2, \text{ then } i_1 \leq i_2
 \end{aligned} \tag{3.1}$$

Figure 3.11: Transition Mapping

3.3.1 Potential Transition Scoring

After common states mapping, we extract the longest common states between *potential transition* and *observable transition*. Then we need to infer the hidden transition which consists of the next letter of the word that the user intends to type. In our method, we use a variant of Sorensen-Dice coefficient in the following formula 3.2.

$$Score = \frac{|A_{(A \cap B)}|}{|A|} + \frac{|B_{(A \cap B)}|}{|B|} \quad (3.2)$$

where $|A_{(A \cap B)}|$ indicates the time interval of the longest common states, and $|A|$ indicates the overall time interval, thus $|A_{(A \cap B)}|/|A|$ is the proportion that the observable transition maps to the potential transition. $|B_{(A \cap B)}|$ indicates the number of states in potential transition that are mapped by the observable transition, and $|B|$ indicates the number of states in potential transition, so $|B_{(A \cap B)}|/|B|$ is the proportion of potential transitions mapped by observable transitions. The score range is same as Sorensen-Dice coefficient, from 0 to 2.

Algorithm 4 Longest Common States Mapping Algorithm

```

1: function LCState(Matrix)
2:    $f(1, j) = \sum_{k=1}^j a_{1k}$ 
3:    $f(i, 1) = \max(a_{i1}, a_{21}, \dots, a_{i1})$ 
4:    $m, n \leftarrow Matrix$ 
5:   if  $i, j > 1$  then
6:      $f(i, j) = \max(f(i, j-1) + a_{ij}, f(i-1, j))$ 
7:   end if
8:   return  $f(m, n)$ 
9: end function

```

The proof of LCSMapping algorithm: We define $f(i, j)$ as the longest common states of first i states in *observable transition* and first j states in *potential transition*.

First given a 1-by-n matrix, in which the *potential transition* contains only one state, the longest common states in formula 3.1 is sum of this row:

$$f(1, j) = \sum_{k=1}^j a_{1k}, j \in [1, 2, \dots, n] \quad (3.3)$$

Then given a m-by-1 matrix, in which the *observable transition* contains only one state, the longest common states in formula 3.1 is the row with the maximum value:

$$f(i, 1) = \max(a_{i1}, a_{21}, \dots, a_{i1}), i \in [1, 2, \dots m] \quad (3.4)$$

Given a m-by-n matrix ($m, n > 1$), we verify the following equation 3.5:

$$f(i, j) = \max(f(i, j-1) + a_{ij}, f(i-1, j)) \quad (3.5)$$

We use the mathematical induction proof as follows.

- 1) When $m = 2$ and $n = 2$, $f(2, 2)$ has three combinations with constraints in formula 1, $a_{11} + a_{12}$, $a_{11} + a_{22}$, $a_{21} + a_{22}$, and $f(2, 2) = \max(a_{11} + a_{12}, a_{11} + a_{22}, a_{21} + a_{22})$, in which $a_{11} + a_{12} = f(1, 2)$ (eq.3.3). According to eq.3.4, $\max(a_{11}, a_{21}) = f(2, 1)$, so $\max(a_{11} + a_{22}, a_{21} + a_{22}) = f(2, 1) + a_{22}$, and then $f(2, 2) = \max(f(2, 1) + a_{22}, f(1, 2))$ satisfying eq.3.5
- 2) when $m = 2$ and $n = 3$, $f(2, 3) = \max(a_{11} + a_{12} + a_{13}, a_{11} + a_{12} + a_{23}, a_{11} + a_{22} + a_{23}, a_{21} + a_{22} + a_{23})$, in which $a_{11} + a_{12} + a_{13} = f(1, 3)$ (eq.3.3), and $\max(a_{11} + a_{12}, a_{11} + a_{22}, a_{21} + a_{22}) = f(2, 2)$ at the first step. So $f(2, 3) = \max(f(2, 2) + a_{23}, f(1, 3))$ satisfying eq.3.5.
- 3) when $m = 3$ and $n = 2$, $f(3, 2) = \max(a_{11} + a_{12}, a_{11} + a_{22}, a_{11} + a_{32}, a_{21} + a_{22}, a_{21} + a_{32}, a_{31} + a_{32})$, in which $\max(a_{11}, a_{21}, a_{31}) = f(3, 1)$ (eq.3.4), so $\max(a_{11} + a_{32}, a_{21} + a_{32}, a_{31} + a_{32}) = f(3, 1) + a_{32}$. According the proof at the first step, $\max(a_{11} + a_{12}, a_{11} + a_{22}, a_{21} + a_{22}) = f(2, 2)$, so $f(3, 2) = \max(f(3, 1) + a_{32}, f(2, 2))$ satisfying eq.3.5.
- 4) when $m = 3$ and $n = 3$, $f(3, 3) = \max(C_{32} + a_{33}, C_{23})$, in which C_{ij} indicates the possible combinations with constraints at the first i rows and j columns. According to the second and third steps, $\max(C_{32}) = f(3, 2)$, $\max(C_{23}) = f(2, 3)$, so $f(3, 3) = \max(f(3, 2) + a_{33}, f(2, 3))$ satisfying eq.3.5.
- 5) Given $f(m, n-1)$, $f(m-1, n)$ satisfying eq.3.5, $f(m, n-1) = \max(C_{m, n-1})$, $f(m-1, n) = \max(C_{m-1, n})$. Due to $f(m, n) = \max(C_{m, n-1} + a_{mn}, C_{m-1, n})$, then $f(m, n) = \max(f(m, n-1) + a_{mn}, f(m-1, n))$ satisfying eq.3.5.

Figure 3.12: Data structure

3.3.2 Candidate Ranking

After deriving the potential transition scoring, the score of the potential transition created by each word in the dictionary is computed. Since the score reflects the similarity of the input and each word, it is regarded as a ranking metric. The larger the score, the higher is the similarity of the input to the word. The *observable transition* is what the user inputs, thus *potential transition* is the word which the user probably intends to type. According to the score, we rank all words in the dictionary, in the descending order. The higher the score of a word, the closer to the beginning of the candidate list it will be. Those words which are at the top positions with the highest scores, are the most likely word that the user intend to enter. The goal of the text entry system is that the intended word should be displayed in the first page of the list of possible words, and the user does not need to scroll the list. Note that in the current approach, we do not rely on the word occurrence frequency in the dictionary, and only use the score to sort the candidate words.

3.3.3 Data Structure

Individually mapping the potential transitions created by each word in the dictionary is consuming in both time and space, because every time we need to create a new matrix for each newly-selected word from the dictionary for comparison. The comparison time is proportional to the size of the total number of characters in the dictionary. From the Algorithm 4, if we add a new state after the last state of potential transition, the longest common states of original matrix does not change, and the longest common states of new matrix only relates to the last state, which is a 1st-order system. Therefore, it allows us to create an augmented matrix.

In our work, we apply a prefix tree structure (Trie) to maintain the words of the dictionary. As shown in figure 3.12, each node contains a letter, its next node, and a flag that indicates whether it is able to form a word through the current node and previous nodes. From the tree, if words have the same prefix, we only operate on them once. In addition, for one branch of the tree, we only need to build one augmented matrix instead of building matrices for each word, i.e. only leaf nodes generate matrices.

From our preliminary experiment results, using the prefix tree structure cuts operation time by nearly half compared to brute force method mentioned earlier. Compared with the recent dwell-free approaches, Filterypedping (the execution time of FdpRevised is the same as FdpRevised's) and MoWing, the execution time of our approach using the tree structure is reduced by 11% and 46% respectively (figure 3.13).

3.4 Evaluation

In this section, we evaluate four standard dwell-free eye typing approaches, the proposed LCSMapping approach, Filterypedping [117], a revised version of Filterypedping (FdpRevised), and MoWing.

3.4.1 Filterypedping, FdpRevised and MoWing approaches

The approach in Filterypedping only relies on the sequential gazed letters. The basic method is to find all possible words which consist of subsequences of the input letter sequence. If the input sequence contains all letters of the intended word, the word will

Figure 3.13: The comparison of execution time per word

be short-listed by the system. Then these possible words in the candidate list are sorted based on the length and frequency in dictionary using the following formula:

$$score(word) = \log(freq(word)) + w * length(word) \quad (3.6)$$

where $freq(word)$ is the number of times the word appears in the Corpus of Contemporary American English (COCA) [142], $length(word)$ is the number of letter in the word, and w is a weight. The higher the score of a word, the closer to the beginning of the candidate list it will be. In their study, the optimal value of the weight w was found to be 1.09.

Since the original Fliteryedping algorithm fails to solve the neighbour-letter error and missing-letter error, we simply revise the matching algorithm part in order to handle the neighbour-letter error (Algorithm 5). The revised one is FpdRevised. The ranking algorithm is the same as the original Fliteryepding using formula 3.6.

The MoWing in GazeTry relies on a cost function which calculates the converting cost between the input letter sequence and each word, and then sorts the candidates according to the cost. In this approach, there are two moving windows, one is for the input letter sequence and the other is for the candidate word, and it calculates the converting cost between the two windows while keeping moving the two windows forward according to

Algorithm 5 The FdpRevised's Matching Algorithm

```
1: function match(word, sequence)
2:    $i, j \leftarrow 0$ 
3:   while  $i \leq \text{word.length}$  do
4:      $\text{validity} \leftarrow \text{false}$ 
5:     while  $j \leq \text{sequence.length}$  do
6:       if  $\text{isNeighbour}(\text{word}[i], \text{sequence}[j])$  then
7:          $\text{validity} \leftarrow \text{true}$ 
8:         break
9:       end if
10:       $j \leftarrow j + 1$ 
11:    end while
12:     $i \leftarrow i + 1$ 
13:  end while
14:  return  $\text{validity}$ 
15: end function
```

corresponding rules, and the summation of the each window converting cost is taken as the final cost. This approach is greatly affected by extra letters. Although it focuses on local optimization, as the number of extra letters increases, the total error rate also increases.

3.4.2 Experiments

Two experiments were carried out. In experiment 1, a simulator was built to simulate the user input with different types of errors, i.e. extra-letter errors, neighbour-letter errors, and missing-letter errors. In experiment 2, ten users taking part in the experiment were instructed to type words into the system.

3.4.2.1 Experiment 1: Simulation

We built a word generator which generates sets of words simulating different text entry errors to compare the resilience of the different algorithms. In the extra-letter error, the simulator created a random noisy input, e.g. irrelevant letters, into a word which simulated that the user gazed at extra letters accidentally. In addition, for the simulated input, the time interval of intended letter was around 10% to 50% longer than extra irrelevant letters, which was to simulate that the user spent relatively longer time on the

Figure 3.14: The top-5 rate in extra-letter error

intended letters. In the neighbour-letter error, the simulator included some neighbour of letters in a word, which simulated that the user gazed at a neighbour of an intended letter rather than the letter itself. In the missing-letter error, the input lacked some letters of a word, simulating that the user does not gaze at the intended locations nor neighbours.

For each type of error, the simulator randomly selects one thousand words from the dictionary to generate one thousand letter sequences, and this operation was repeated 100 times. Thus, a total of 100 thousand letter sequences were generated for each type of error. The set of simulated letter sequences were used as input to the four algorithms to establish the recommended words based on the letter sequences.

Figure 3.14 shows the top-5 rate results with the extra-letter error. The top-5 rate is the percentage of the intended words existing at the first five positions of the list of recommended words, which reflects the accuracy of a typing system. The X axis is the number of simulated extra letters, which indicates how many irrelevant letters the simulator generates into the word. From the results, when there was no extra letter (i.e. the number of the extra letter is 0) the top-5 rates of Filteryedping, MoWing and LCSMapping were almost the same (approaching 100%), however the top-5 rate of FdpRevised was only 62.8%. As the number of the extra letter increased, the top-5 rate of MoWing decreased dramatically. When the number of the extra letters was 1 or 2, the top-5 rate of Filteryedping was 0.2% higher than LCSMapping, likely because the

Figure 3.15: Average position of words in the candidate list

LCSMapping algorithm had a bigger search space for possible words. If the score of the intended word was not ranked in the first five positions in the candidate list, it is flagged as unsuccessful detection of intended word. When the number of the extra letters was larger than 2, LCSMapping gradually performed better than Filteryedping, with LCSMapping performing 20% higher with 10 extra letters. From our preliminary experiment of actual users, the typical number of extra letters is around 5, and LCSMapping accuracy at this point is around 4% higher than other algorithms. Statistical analysis showed significant differences between LCSMapping and Filteryedping ($p = 0.0174 < 0.05$). However, since FdpRevised allowed a word to consist of neighbours of the gazed letters, the search space was much larger than the original Filteryedping approach. When the number of the extra letters increased, the top-5 rate of FdpRevised exponentially decreased to nearly 0%.

To further understand the accuracy of the different approaches, we were also interested in the average position of the intended word in the candidate list. Figure 3.15 shows the comparison of the average position of the different method, in which the X axis is the number of simulated extra letters, and the Y axis is the average position. This result also supports the top-5 rate in figure 3.14. When the number of the extra letters is 0 to 4, Filteryedping and LCSMapping have the similar results, i.e. the average position is around 1 which means the intended words are always listed in the first position. As

the number of extra letters increases, the average positions of Filterypedping gradually increase and exceed that of LCSMapping, indicating deteriorating performance. The average position of FdpRevised is high even when there is a single extra letter error indicating clearly that it is unable to cope with the extra letter error. In the case of MoWing, the average positions increase as the number of extra letter is increasing again showing its inability to cope with the extra letter error problem.

Figure 3.16 shows the performance of four algorithms when there are both neighbour-letter and missing-letter errors. In figure 3.16a, the X axis is the number of letter which is replaced by its neighbour. In figure 3.16b, the X axis is the number of missing letter in the word. Due to the word length limitation, only parts of the word can be changed to neighbors or missing, otherwise it is not meaningful if the whole word information is lost. Thus from our preliminary investigation, we set the neighbor-letter error from 1 to 5, which means that, at most, five letters of the word were changed to their neighbors. The missing-letter error was set to 1 to 3, which means at most three letters of the word can be missing.

The results show that LCSMapping is able to handle both neighbor-letter and missing-letter errors, in which the top-5 rate is still higher than 80% even though the neighbour-letter error is 5 and missing-letter error is 3, which is better than MoWing (40.7% and 52.5% respectively). However, although the neighbour-letter and missing-letter errors are 1, The top-5 rate of Filterypedping is around 12%. Thus Filterypedping is not suitable for handling text entry with these types of errors. Although the top-5 rate of FdpRevised (37.85%) is much higher than Filterypedping (11.89%), its performance decreases dramatically once the error increases due to its poor search-space control.

The results of simulation show that LCSMapping outperforms other approaches in both neighbour-letter and missing-letter errors without sacrificing performance in extra-letter errors. The Filterypedping approach had a good performance in the extra-letter error case, while in both neighbour-letter and missing-letter errors cases it had very poor performance. Although the FdpRevised approach improves the accuracy in the neighbour-letter error case, its overall performance is relatively low due to its poor search-space control. While the MoWing approach is able to solve the problem of both neighbour-letter and missing-letter errors cases, it sacrifices the accuracy in the extra-letter error case, which, in fact, is the most common error case in practical eye typing environment.

Figure 3.16: The top-5 rate in two errors

3.4.2.2 Experiment 2: Actual User

We evaluated the performance of the proposed approach via actual user experiments. The interface was presented on a Dell P2314H 23-inch LED-lit monitor with a resolution of 1920×1080 at a frame rate of 60Hz, and were run on a Dell-PC with a Xeon(R) processor at 3.5GHz and 16GB DDR-RAM under 64-bit WindowsTM 7. We used a low-cost eye tracker (TheEyeTribe¹) put under the monitor shown in figure 3.17. The eye tracker had a sampling rate of 60Hz. Nine-point calibration was used in the study. The tracked gaze coordinates data were mapped onto the mouse cursor using moving average smoothing algorithms provided by the device. A chin rest, used to help participants understand that doing eye typing they only need to move their eyes and also help to keep calibration, was placed 50 cm in front of the monitor. The participants did not need to re-calibrate the tracker unless they were uncomfortable with the accuracy of the eye tracker, or there is a big drift.

We recruited ten volunteers (7 male, 3 female; 23-35 years) from the local university. All of them had 4 to 8 hours per day computer usage, and had normal vision. Three of them had rich prior experience with eye tracking (expert), and seven of them had no such eye-tracking experience before (novice).

All participants had at least half an hour training to control the cursor using eye movements, and type letters using gaze. In the experiment, there were five sessions, but from results the users did not demonstrate significant different performance among

¹<http://theeyetribe.com/>

Figure 3.17: Participant performing experiments

the five sessions. Between sessions, the participants had a five-minute break and then underwent re-calibration. In each session, each participant needed to type 200 words using the proposed eye typing system. These words were randomly selected one by one from the dictionary which has 5000 commonly-used words in the Corpus of Contemporary American English (COCA) [142].

In the experiment, a random word was selected and displayed at the top of the screen. The user was required to type the word using eye movement and gaze. Since the proposed system is a dwell-free typing system, they did not have to spend dwell time typing individual letters of the word. Like the finger swipe-based typing, they only needed to “swipe” the letters of the word using gaze. Upon completing the word entry, they had to gaze at the “SPACE” key to indicate the end of entry. The system then displayed the candidate words at the bottom of the screen. At the bottom of the screen, there were four rows of the top-5 ranked words recommended by each of the four algorithms, Filteryedping, FdpReivsed, MoWing, LCSMapping, respectively. The system simultaneously showed up the top 5 words at the four lists. Since we are evaluating the performance of four algorithms in typing errors rather the typing performance of the participants, they did not need to choose the word from candidate words, but they were encouraged to look at the recommended results, and kept typing the next given random word. Then a new random word was displayed at the top of the screen.

Figure 3.18: The results of actual users

Figure 3.19: The average top-5 rate with error bars

Figure 3.18 demonstrate the top-5 rate and the error rate of the ten participants. The error rate is the percentage of the intended words being out of top 30 in the candidate set, which reflects the robustness of the typing system. As can be seen, the performance of the three experts was better than seven novices. Figure 3.19 shows the average top-5 rate with error bars, where the results of MoWing and Filteryedping were almost same. This result fully supports our preliminary observations that the extra-letter error is the most common error, with new users trying to input all letters of the intended words resulting in more extra letters in their input. Therefore, although the MoWing approach improves its performance in neighbour-letter and missing-letter errors cases, its overall performance does not improve with sacrificing the performance in the most common, extra-letter error case. Among the seven novices, the top-5 rate of LCSMapping was around 20% higher than Filteryedping and MoWing, and the error rate of LCSMapping was only half of Filteryedping and MoWing. From the results of the three experts, the top-5 rate of LCSMapping was around 20% higher than Filteryedping, and the error rate of LCSMapping was nearly 0%, but the error rate of Filteryedping was higher than 10%.

3.5 Discussion

Through running experiments with simulation and actual users, we have proven that the proposed LCSMapping is a robust approach in the eye-typing system that is able to handle extra letters, neighbouring and even missing letters, both in terms of accuracy metrics and error metrics. In the simulation study, we developed a word error generator to simulate the different types of errors as well as the number of such errors. This is essential for us to determine the performance of the different algorithms. It helps us collect an amount of data that could be difficult to obtain with actual users, because in eye-tracking area, we commonly experience difficulties in finding users for our eye tracking studies [143]. Therefore, the simulation experiment is able to not only provide a theoretical testing of the approach, but also allow us to control specific (error) environment so that we optimize the system before testing on actual users. It is important for the eye-typing system design or even other HCI systems in some cases, for example, finding actual users is difficult, requiring repeated practical experiment is expensive, and quickly testing prototype is necessary.

During our study, we identified several steps in a typical eye-typing system, i.e. calibration, detection, recognition, prediction, and construction. While calibration and recognition are unique steps in the eye-typing system compared with a traditional finger typing system. Specifying the unique parts helps us define the clear purpose/scope of developing the system. The quality of calibration is hardware-dependent, e.g. the accuracy of the eye tracker. In general, almost all eye-typing work assumed the perfect calibration, i.e. the user most probably fixates on the correct letter directly. In order to guarantee that, usually the high-quality eye trackers are applied, or users are required to re-calibrate frequently. However, the high-quality eye trackers usually are very expensive, which could be one main reason limiting the prevalence of eye-tracking applications requiring the high tracking accuracy. Nowadays as human-computer interaction tends to be seamless and natural, repeating the calibration process would be burdensome, even though the eye-tracking devices are non-invasive for users. Therefore, reducing the calibration is a challenging issue that cannot be ignored when developing the eye-typing. Although some automatic re-calibration works have been proposed [134, 144], they are not suitable for the real-time eye-typing system, and still contain the error range. Therefore, handling poor calibration or calibration drift is necessary in the rest of steps, and our system addresses this problem in the recognition step. The detection step exists in any typing system, which is basically that the system receives the “operation command” from the human with some initial processing.

In our study, we distinguished recognition and prediction, even though sometimes the two terms are interchangeable in typing systems. In this context, recognition is to process noisy inputs of the user, while prediction is a further process relying on syntactic/semantic analysis. This work is focusing on the recognition step instead of prediction. The purpose of our work is to efficiently recognize the potential words with high probabilities only based on what the user inputs, i.e. a robust approach handling noisy input. In recognition approaches, the key algorithm is to define the similarity/dissimilarity between candidate words in the dictionary and input of the user. In general, they are categorized into shape-based matching approaches and string-based matching approaches.

Shape-based approaches compare the shape of the path covered by the eye gaze with shapes stored in a word list [111, 145, 146, 147], and these approaches have been widely

used in finger-based typing systems with the touch screen, such as Swype² and SwiftKey³ entry system for smart phones. The touch screen has enabled users to enter text by just dragging their fingers across the screen keyboard. While in eye-typing system, the shapes of the eye-gaze paths are always sharp (not smooth as dragging by fingers), and the shape-based approaches are less helpful. Pedrosa et al. [117] investigated the shape-based approach in which the error rate is very high in the eye-typing system, and it is not suitable for practical eye typing.

While string-based approaches usually apply string matching techniques finding the similarity/dissimilarity of candidate words and converted letter sequence of input stream of the user. Edit distance and longest common subsequence are widely used as the similarity metrics, and more comprehensive surveys can be found in [148]. However, the existing string similarity metrics are location-independent (or keyboard-layout-independent), for example in the edit distance method, the distance between “a” and “s” is the same as the distance between “a” and “p”, but as we know, they cannot be regarded as the same distance according to the QWERTY layout. In practice the location information is very important when low accuracy or calibration draft exists, and the neighbouring letters of gaze fixations (or gazed letters) should be also taken into account. In addition, another important factor in the eye typing we mentioned before is the time duration of gazed letters. The duration factor is able to help us identify the dominant letters, which improve the recognition accuracy and speed up the system. Therefore, the proposed LCSMapping is a string-based matching approach with location and time duration information, i.e. space and time dimensions, which are two intuitive factors of eye typing.

Although our work is not focusing on prediction and construction steps, we still discuss the potential research that can be integrated with our system. Our system is able to allow the user to focus on the entire word entry instead of individual letters. Furthermore, with syntactic/semantic analysis and even typing behaviours of the user, it is possible to allow the user to focus on the entire sentence, especially daily sentences. The recommended words in our system can also be a “coarse” input feeding forward into the prediction engine, and then the better qualified words would be recommended. In

²<http://www.swype.com/>

³<http://swiftkey.com/en/>

eye-typing systems, the real-time prediction is seldom used, because it would disturb the attention of the user, and then the user usually types all letters of the intended word at once [149]. In traditional finger typing systems, input controller (finger) and visual feedback (eye) receiver are different. While using eye movement as the input controller, the visual feedback cannot be received in parallel, and then the audio feedback can be a good alternative [150]. Although candidate words are ranked highly with their corresponding probabilities, the users still need to select the intended word that requires cognitive effort and takes some time. Thus if the intended words are always ranked in the first position, it could be more portable for practical typing. In our experiment, using LCSMapping, 87.6% intended words of the three expert users were ranked in the first position, compared with Filteryedping (61.4%), FdpReivsed(2.5%) and MoWing(76.5%).

We summarize the differences of four algorithms in Table 3.3. On processing the gazed letters of a user, Filteryedping and FdpReivsed only extract the ordered letters that the user has gazed at. MoWing and LCSMapping extract time duration of gazed letters, besides the letter sequence. The gaze duration provided us with the dominant letters in the sequence of letters entered. Thus enabling a better decision making by our algorithm. As for the similarity calculation (string matching), Filteryedping finds the possible words consisting of the letter sequence. The intended word must be a subset of the sequence of letters entered, which eliminates many possible words. Thus it is not able to handle missing-letter errors as well as neighbour-letter errors. FdpRevised takes letter sequence' neighbour into account that makes its search space much larger, which make it fail in controlling the size of candidate words. Therefore, although the ranking method in Filteryedping and FdpRevised is reasonable, it still cannot recommend words efficiently. While MoWing uses the a cost function to calculate the converting cost of the entered letters and the words in the dictionary, which controls the size of the candidate-word space, but this method searches for the local optimal. As the number of extra letters increases, the performance deteriorates gradually. LCSMapping finds the longest mapping states between the letter sequence and each word, and this method is able to reach the global optimal. From the Simulation and actual users experiments, Filteryedping is only able to handle the extra-letter error, and FdpRevised has the very limited ability to handle the neighbour-letter error, while MoWing is able to handle the

Table 3.3: The summary of four algorithms

	Filteryedping	FdpRevised	MoWing	LCSMapping
Letter Sequence	Only gazed letter sequence		Gazed letter sequence with corresponding time duration	
String Matching	Find words consisting letters of the sequence	Find words consisting letters the sequence, or neighbors of these letters	Calculate the cost from the sequence to words	Find the longest common states mapping between sequence and words
Candidate Ranking	$\log_{10}(\text{freq}(\text{word})) + w * \text{length}(\text{word})$		Sorting by the cost	Sorting by the mapping score
Types of Error Solved	Extra-letter error	Neighbour-letter error (little)	Extra-letter error (little) Neighbour-letter error Missing-letter error	Extra-letter error Neighbour-letter error Missing-letter error

small number of the extra-letter error. The proposed LCSMapping is resilient to all of the three common errors.

In the experiment, the users were not required to select the intended word, but they were encouraged to look at the candidate list (top-5 words). The purpose of our work is to understand common text entry errors using gaze and improve accuracy of the work, rather than the entry speed, because the speed is not only depending on the proposed approach, but also other factors, e.g. user proficiency, UI design, even the eye tracker. It is promising that in the future thorough experiments are carried to control these factor, and also investigate the usability of the system. From the experiment results, novices show relatively poor performance where even using our proposed algorithm, LCSMapping, the top-5 rate is less than 70%. From their experiment records and feedbacks, there are two main reasons, the first one is that they are not good at controlling the cursor using gaze, and the other one is that they forget some letters of the word while gazing sequentially. It can be a guide of subsequent UI and experiment design, such as replacing the cursor, and allowing the users to memorize the given word carefully first before starting to type the word.

3.6 Summary

In this work, we first propose a dwell-free system, GazeTry, with two robust recognition algorithms (MoWing and LCSMapping) respectively for inferring the words which the user intends to type. The proposed algorithms determine the intended word based on the sequential letters of gaze trace instead of pausing at each individual letter, which are more robust to missing letters and when a neighboring letter on the keyboard is incorrectly selected. Unlike simple filtering and string matching methods, the proposed approach

takes the letter location and gaze duration into account besides the gazed letters, which has made dwell-free typing even more accurate and more resilient to text entry error, much akin to finger-swipe typing, even using a low-cost eye-tracking device. Furthermore, implementing the tree structure to process the recommendation, the operation time of LCSMapping is reduced by 11% and 46% respectively, compared with Filteredping (FdpRevised) and MoWing.

Chapter 4

Assistive Text Entry Using Gaze with an Off-the-shelf Webcam

In this chapter, we propose a prototype of eye-typing system using an off-the-shelf webcam without the extra eye tracker, in which the appearance-based method is proposed to estimate people’s gaze coordinates on the screen based on the frontal face images captured by the webcam. We also investigate two critical issues of the appearance-based method, which help to improve the estimation accuracy and reduce computing complexity in practice. Five users participated in the eye-typing experiment, and the performance evaluation shows that eye typing with webcam using the proposed method is comparable to the eye tracker under a small degree of head movement.

4.1 Gaze Estimation From Eye Appearance

As introduced in Chapter 2, appearance-based gaze estimation does not require the high-quality input source, but usually utilizes a low-resolution eye-area image as the input, without highly-accurate features detection. Each pixel of the image is described as one dimension in the appearance space, so that the entire image can be described as a feature point in the high-dimensional space, where the dimensionality is the same as the number of pixels in the image. The basic assumption of the appearance-based methods is that although these image points are in the high-dimensional space, they still follow the distance rule in Euclidean space. The two similar images would be two corresponding points close to each other in the high-dimensional space, and vice versa.

Based on the assumption, the relative similarity among images will be represented by the relative distance of corresponding points in the high-dimensional space, as shown in figure 4.1. In fact, due to the rotation of eyeball, there is small distortion of the iris area in 2-D image, or the minor variance of iris and pupil size at different directions, which is ignored in our work.

Figure 4.1: The relative relation of images and points. left-hand side indicates the eye-area images that looking at nine different positions, while right-hand side indicates the corresponding points in the high-dimensional space.

Figure 4.2: The mapping from high-dimensional space to gaze coordinate plane.

The “movement” of iris area is the main factor of variation of the eye appearance. The eye movement is rotating around the eyeball centre, so that iris area is moving in the 2-D surface which is embedded into the 3-D space. According to the properties of the Euclidean space above, the corresponding points of the eye images will form a manifold in the eye appearance space that has an approximately 2-D surface, so that the relative similarity would be maintained inside, and the 2D gaze points of eye images are sharing the same local interpolation with corresponding image points in the appearance space (figure 4.2).

Figure 4.3: The rough eye-region image extraction

4.1.1 Eye-Appearance Feature Vector Extraction

Given a face image captured by webcam, the eye-appearance feature vector is extracted in the following steps.

- (i) *Eye-region image cropping* (figure 4.3). The first step is to extract the eye-region images. For the efficiency of computing and image processing, it is common to convert RGB scale image to gray scale. Initial eye areas (left and right) are detected by using Haar-cascade classifier [72]. In order to guarantee the correct eye-area extraction, we use two methods to validate the detection result. 1). Eye should be contained in the face area. The whole face area is detected using Haar-cascade classifier, and the margin between eye area and face area should be in the predefined interval. 2). The average intensity of eye area should be lower than the average of the face area. The valid detection result is taken as the rough eye region.
- (ii) *Feature vector generation* (figure 4.4). The extracted rough eye-region image usually contains the eyebrow, which is irrelevant to the gaze. Eyeball and eyebrow areas are two darkest parts in the image. For eyebrow removal, we apply the horizontal integrated projection of the image, which is adding up the pixel intensity of each row, and removing the upper-peak area. Then, we further crop the eye area in terms of the outer and inner eye corners detected by Canny edge filter [128], and the eye image is resized at the fixed aspect ratio, 60×36 pixels as suggested in [129].

Figure 4.4: Eye-appearance feature vector generation.

Since the pixels of an image is represented by numerical values in computer system, the image can also be considered as a matrix of pixel intensity. Through a raster scan of the matrix, we obtain the final feature vector of the eye appearance.

Each element in the feature vector corresponds to a pixel of the cropped eye-area image, i.e. the image is fully described by a 2160-dimensional vector, and the image can also be represented by a point in the appearance space. Due to two degrees of freedom of iris-area movement in the images, the corresponding points would move across a 2-D surface embedded into the high-dimensional space, which constitutes a manifold, called eye-appearance manifold.

4.1.2 Eye-Appearance Manifold Verification

As we mentioned above, although the “image points” are in a high dimensional space, the constituted manifold of these points has an approximately 2D surface. Each point of an eye appearance on the 2D-surface will correspond to a gaze point that a person is looking at on the screen, and these points in the respective spaces are one-to-one mapping. Therefore, if the person looks at distinct letters on a virtual keyboard, appearance points of eye images should be also differentiable, which is a prerequisite of recognizing which letters the person intends to enter using the eye image captured by the webcam.

To verify the practicability, we conducted a simple experiment. A participant was asked to sit in front of a monitor with webcam, where the monitor displays an on-screen virtual keyboard as shown in figure 4.5. The keyboard is the same keyboard in our eye-typing system. To guarantee limited head movement, the participant was asked to

Figure 4.5: The 26-letter keyboard layout

move the head to a comfortable position first and then only rotate eyeball to gaze at every letter one by one on the virtual keyboard. while looking at the intended key, he clicked a “confirm” button, and then the webcam captured fifty consecutive frontal face images within a short time. All images were be processed by using the above extraction steps, to generate the eye-appearance feature vectors. For a better visualization, we then used principal component analysis (PCA) to project these high-dimensional vectors into 2-D/3-D space.

Figure 4.6a shows the percentage of eigenvalues of the transformed data using PCA. The first three components contain around 90% of accumulated eigenvalues, which means only using the three dimensions will well represent the data distribution in original 2160-dimensional space with little information loss. This observation verified the previous statement that the image points are embedding into a low-dimensional manifold, though they are in the high-dimensional space. Figure 4.6b plots the data distribution in first three principal components (3-D space), and the low-dimensional manifold of points has an approximately 2-D-surface shape. Figure 4.6c plots the first two principal components, where there are 26 clearly-separated clusters. The eye-appearance points looking at different letters are differentiable in 2-D space, which can be taken as a classification problem. In addition, the more important observation is that the relative positions of the clusters are the same as the keyboard layout, which means the linearity is existing in the manifold. The classification problem can be further converted to a regression problem, and gaze estimation becomes feasible in practice.

Instead of explicitly extracting the mapping function between local features (e.g. pupil centre, eye corners, or glints) and gaze points, the appearance-based methods implicitly

Figure 4.6: The PCA projection. (a) shows the percentage of eigenvalues. (b) plots the 2-D-surface manifold in 3-D space formed by the first three principal components. (c) plots image points distribution with the corresponding key labels in 2-D space formed by the first two principal components.

extract the hidden mapping relations between the eye images and gaze coordinates. As we can see, the pattern that the eye appearance changes in high dimensional space is like the variation of gaze points on the 2-D screen, and the local linearity of the two spaces shows the relevance.

4.1.3 Manifold Learning Method

Appearance manifold is a continuous set of eye-appearance feature points embedding in the high dimensional space, and any point in the manifold can be linearly interpolated by its local neighbours. The task of manifold learning is reducing the dimensionality, i.e. extracting the low-dimensional manifold from the high-dimensional space, and the relative local geometric position should be maintained. In the practical learning problem, training samples are a set of high-dimensional data (eye appearance vectors) with

corresponding low-dimensional data (2-D gaze coordinates), and estimating the gaze coordinate of a new eye appearance vector. Basically, there are two types of the learning, *global* and *local* learning.

Global learning is directly learning the global parameter between eye appearance vectors and corresponding gaze coordinates, and using the same parameter to determine the gaze coordinate of a new eye appearance. This method only estimates the global parameter using training samples once, but it cannot guarantee the invariance of relative position of data points in the two spaces. Therefore, we work with the local learning in appearance-based gaze estimation. In order to maintain the relative positions of gaze points in the eye-appearance manifold, the interpolation-based method with locally linear embedding [14, 127] is used to find the mapping relation. The method finds locally linear mapping instead of global mapping directly from high-dimensional data to low-dimensional data. The basic idea is estimating the mapping parameters (weights) of a new high-dimensional data point from its neighbouring data, and assuming the low-dimensional space also shares the same locally linear mapping, and then applying the same parameters to the corresponding low-dimensional data points to obtain the new low-dimensional data point (eq.5.1).

$$\begin{aligned} Xw &= \hat{x} \\ Pw &= \hat{p} \end{aligned} \tag{4.1}$$

where X is a matrix consisting of eye-appearance feature vector, P is the matrix consisting of corresponding gaze points, and w is the parameter that needs to be estimated. We do not estimate the mapping parameters between X and P . Given a new eye-image data point $\hat{x}$, we find the local mapping parameter w among its neighbors, and then estimate the gaze point $\hat{p}$, by assuming it shares the same neighboring mapping with the new eye-image data. However, it contains errors between the locally linear combination Xw and the new eye-image point $\hat{x}$, because the equation is overdetermined. The objective is to minimize the error by tuning parameter w . Therefore, we have the gaze estimation function with optimization constrains in eq.5.2

$$\begin{aligned} \tilde{w} &= \arg \min \left| \hat{x} - \sum_i^k w_i x_i \right|, s.t. \sum_i^k w_i = 1 \\ \hat{p} &= \sum_i^k w_i p_i \end{aligned} \tag{4.2}$$

Figure 4.7: The procedures of the appearance-based approach.

where $\tilde{w}$ is optimal weight vector formed by the scalars w_i . $p_1 \dots p_k$ are the corresponding gaze points in 2D space. Once the optimal weights $\tilde{w}$ are obtained, the gaze point $\hat{p}$ is estimated using the same weights. The entire procedures of the method is shown in figure 4.7.

4.2 Practical Issues

Training data of interpolation-based methods are the most important to infer a new sample. In the gaze estimation method, training data are obtained from the calibration phase. Thus, guaranteeing a high-quality calibration is very crucial for the accuracy of gaze estimation. In addition, dimensionality reduction is also important part for a learning method to avoid the “curse of dimensionality”. Therefore, in this work, we will study the above two issues: 1). improving the gaze estimation by reducing artefacts of calibration images. 2). reducing the image size by detecting the effective areas of eye image in determining the gaze point.

(a). Experiment setup

(b). The layout of 35 cross-hair markers

Figure 4.8: The experiment setup with a desktop computer which displayed 35 cross-hair markers.

We developed the system on a desktop computer with a 23-inch LED-lit monitor attaching an off-the-shelf webcam (30fps) as shown in figure 4.8. The eye tracker (TheEyeTribe, 30Hz) was placed under the monitor. The chin rest, placed 50 cm in front of the monitor, is used to minimize the head movement of the participants. The experiment procedure is as follows: the participants first did the nine-point calibration with the eye tracker. Thirty-five cross-hair markers are individually displayed on the screen and the participants are asked to look at the central of the marker. To help the participants fixate at the centre, they were instructed to move the mouse cursor to align with the marker, and shape of the mouse cursor is the same as the marker [127]. When the mouse cursor overlaps with the marker, the participants are required to click on the mouse key. The webcam will simultaneously capture 10 images of the participants. Both the marker's position and the gaze coordinate estimation of the eye tracker are also recorded.

Three volunteers (3 male, 25-33 years) from the local university participated in the experiment. All of them had normal vision. One of the participants had prior experience with the use of eye tracking input software/devices while the remaining two volunteers are novice. The leave-one-out cross validation is employed in which one eye image is used as the test image and the rest for training. To measure the accuracy, the mean estimated

Figure 4.9: Individual images and averaging images

angular error is calculated as eq.5.6:

$$error = \frac{1}{n} \sum_{i=1}^n \arctan\left(\frac{\|\hat{p}_i - p_i\|_2}{d}\right) \quad (4.3)$$

where $\|\hat{p}_i - p_i\|$ is the Euclidean distance between estimated gaze position $\hat{p}_i$ and actual gaze position p_i . d is the distance between participant's eyes and the screen. We assume that the line of sight of the participant is perpendicular to the screen when the participant fixates at the centre.

4.2.1 Artefacts Reduction

In the appearance-based method, the images of calibration are determining gaze points the subsequent new images. If the calibration images have the bias caused by artefacts while doing the calibration, it would lead to large estimation error in gaze estimation. In previous work [127, 128], the system captured only one face image of participants while looking at each calibration point. Although the images are captured in a well-controlled environment, there are inevitable factors that degrade the quality of the calibration, such as instant fine-grain eye motion and illumination variance. The artefacts might cause a bias of captured images while looking at the same calibration point. In order to improve

Figure 4.10: Evaluation results

the quality of calibration, reducing the impact of artefact is necessary at the calibration stage.

In our experiment, multiple images (10 images) were captured once the participant clicked the reference point while doing the calibration. Since 35 reference points are used for calibration, there are 35 sets of images where each set contains 10 images (figure 4.9a). In order to compare with the performance of capturing one image of calibration, we select only one image from each set. This selection method would generate 10^{35} image combinations which is too large, and for computing efficiency, we choose 10,000 image combinations of them, i.e. randomly selecting 35 images from 35 sets respectively (figure 4.9b), and repeating 10,000 times. For reducing the impact of artefacts, we proposed an averaging filter to each set of images (figure 4.9c).

To evaluate the performance of the two methods, we used the leave-one-out cross validation. The average estimation error (angular) was taken as the performance metric in equation 5.6. Figure 4.10 shows the evaluation results of three participants. The first and second bars indicate the average error of 10,000 image combinations and minimum error of 10,000 image combinations (using best individual images), and the third bar indicates applying the averaging filter. We also calculated the error of gaze points estimated by the eye tracker. As we can see, the average estimation error of using individual image is

0.2 degree higher than using the proposed averaging filter. The minimal error of using individual images is still 0.1 degree higher. After applying the averaging filter, in subject 1 and 3, the proposed appearance-based method even outperforms the commercial eye tracker. The observation also verified that the artefacts of calibration images have a great impact in the gaze estimation. In the experiment, we only evaluated a subset of 10^{35} image combinations. Although there is no guarantee that the averaging filter is better than any “best” one of 10^{35} image combinations, the proposed filter is reducing the error by 20% from the average combination, and thus it is still a good method to minimize the impact of artifacts, and improve the estimation accuracy.

4.2.2 Effective Area Detection

In the appearance-based gaze estimation methods, the gaze status is determined by the appearance of eye area, rather than other face features, e.g. nose, mouth. In general, the eye-area image is cropped at a predefined aspect ratio in terms of outer and inner eye corners. However, there is a fundamental issue that has not been studied, which is how the different sizes of eye images affect gaze estimation. Without addressing this issue, the current cropping method might not be efficient. To further crop the eye area, we define three cropping operations as shown in figure 4.11. A single horizontal cropping operation is cropping a row of both top and bottom pixels of the eye image. A single vertical cropping operation is cropping one column of both right and left pixels. A single full cropping operation is a combination of horizontal and vertical cropping. The original size of the eye image is 60×36 pixels.

In the experiment, we still used the leave-one-out cross validation to evaluate the performance of each cropping. In figure 4.12, the x-coordinate indicates the number of cropping times. For example, the number of horizontal cropping is 1, which means removing a row of both top and bottom pixels of the image respectively. The y-coordinate denotes the angular estimated error. There are two observations: 1). the estimation errors remain almost constant as the image is cropped horizontally until the “knee” point, above which the estimating error increases drastically. It was observed that the “knee” point occurs when the upper eyelid is cropped; 2). the effect of vertical cropping is less than horizontal cropping. Although both outer and inner eye corners are cropped, the

Figure 4.11: Cropping operations. The process of two-time horizontal, vertical, and full cropping operations.

error does not increase drastically. This suggests that eye corners as used in feature based methods are less critical. If the image can contain the whole iris, the gaze estimation error does not increase drastically.

The observations indicate the different effective features determining the gaze in appearance-based methods. In feature-based gaze estimation methods, eye corners are taken as the second important features besides the pupil centre. However, in appearance-based methods, eye corners are not important features, and the upper eyelid plays an important role that affects the gaze estimation. The importance of different eye features also reveals the mechanism of feature-based and appearance-based methods. Feature-based methods mainly focus on the relation of pupil centre and eye corners, and the eye corner is taken as a reference. However, appearance-based methods make use of the appearance variance of the eye area. Besides the movement of the iris part, the eyelid openness also affects the appearance, but the appearance of eye corner is relatively invariant. These observations also help to improve the current cropping method that the inner and outer eye corners are not necessary to be contained, which is to further reduce the size of eye-area image. In figure 4.12c, despite ten-time full cropping, i.e. the size of eye image becomes 40×16 pixels from 60×36 pixels, the estimation error does not increase comparing with the original image. Meanwhile, the dimension of the appearance

(a). Horizontal cropping

(b). Vertical cropping

(c). Full cropping

Figure 4.12: Evaluation results. The error of horizontal, vertical, and full cropping operations in different cropping times.

feature vector is also reduced from 2160 to 640, which decreases computational cost and time complexity.

4.3 Eye Typing Experiment

In the experiment, we developed an eye-typing system using webcam in desktop computer, which is configured with a 23-inch monitor. The placement of equipments is as follows: 1). The off-the-shelf webcam was attached on the top of the monitor; 2). the eye tracker was put under the monitor; 3). the chin rest was mounted on the desktop, and the distance between the chin rest and the monitor is 50cm. The layout of on-screen virtual keyboard is the same as previous work. Five volunteers (4 male, 1 female, 25 - 28 years old) were invited in the experiment, where all of them participated our previous eye-typing experiments, and thus they had rich prior experience with eye typing.

The experiment procedures are as follows: 1). the participants put the head upon the chin rest and moved to a comfortable position. They were asked to try to keep the head stationary, and only rotate the eyeball. 2). Then they were instructed to do the nine-point calibration with the eye tracker. 3). They were required to type 120 words using gaze with the eye-typing system. The words were randomly selected one by one from 5000 commonly-used words [142], and displayed in the top of keyboard which is the same as the previous work [151]. The 120-words typing was divided into six consecutive

Figure 4.13: The procedure of eye typing using webcam.

Figure 4.14: Recognition results.

sessions. In each session, the participants first did the calibration with the 26-letter keys, where the system recorded the pairs of the face images captured by the webcam while they were looking at each key, and the corresponding keys' coordinates. After doing the calibration, the participants would type 20 random words by sequentially gazing at the letter keys of the given word. while typing each word, the webcam captured the images at 30 frames per second, which is the same as the sampling frequency of the eye tracker. After completing all six sessions, there are six sets of calibration images, and 120 sets of images typing words.

All gaze points of word-typing images were estimated by the calibration images. For the purpose of comparison, we applied two scenarios to estimate the gaze points of the images, 1). Only the calibration images of the first session were used to estimate all 120 sets of word-typing images; 2). the calibration images of each session were used to estimate word-typing images in the same session (20 sets), respectively. These estimated points were collected in accordance with the time series. To evaluate the performance of eye typing using the webcam, we take recognition accuracy of intended words as the metric. A robust eye-typing algorithm, LCSMapping [151], was used to recognize the word in terms of the sequential gaze points, where it would rank all possible words in the dictionary by the probability. In the eye typing system, the top-k rate is usually used as the accuracy metric, i.e. the intended word is ranked in the top-k words. In our experiment, we used the top-5 rate metric as same as our previous evaluation, i.e. If the intended word is listed in the top-5 candidate words, it is regarded as correct recognition.

Figure 4.14 shows the recognition results of 120 words that five subjects type. In figure 4.14a, there are the results when only the calibration images of the first session are used to estimate gaze coordinates of images in subsequent five sessions. The x-axis indicates the five subjects, where in each subject, the six bars indicate the six sessions of typing in order. The y-axis indicates the number of words that are successfully recognized. As we can see, in the first session, almost all twenty words can be recognized, the number of recognized words decreases over time (sessions). While using the calibration images of each session to estimate gaze coordinates of images in the same session, almost all words are successfully recognized. In the experiment, using the gaze coordinates estimated by the eye tracker, all words can be successfully recognized. For the better understanding of the quality of gaze estimation, we plot the estimated gaze points of typing several typical words with the virtual keyboard in figure 4.15. The blue dots and red dots represent the estimated points of our proposed method and the eye tracker, respectively. Along with the virtual keyboard, the upper and lower right box list the top five words that are recommended by the LCSMapping algorithm using “blue” points and “red” points respectively, and the intended words are highlighted.

Due to the smoothing algorithm of the eye tracker, the neighbouring points within a predefined threshold would be integrated into the same coordinates, and thus the red dots look concentrated and fewer. As we can see, the estimated points of the eye tracker is more accurate as the red dots cover all letters of the ten words which are ranked at the first place. The point distribution (blue dots) of the ten words represents four typical cases using the appearance-based method, 1). the nearest letters of these points contain all letters of the intended word; 2). there is a small drift between the points and the intended letters, i.e. some nearest letters of the points are neighbours of the intended letters. Since the LCSMapping algorithm can handle the neighbour-letter error well, the intended word is still ranked at the first place; 3). along with the neighbour-letter error caused by the drift, many other irrelevant letters are also included which cause the extra-letter error, and thus the priority of the intended word decreases, but it is still in the top five words. 4), as the points keep drifting, the missing-error is also existing which might make the recognition algorithm fail to recommend the intended word.

The above observations help us understand that calibration drift is the critical factor of word recognition in eye typing using the appearance-based method. When we only use

Figure 4.15: Estimated gaze points of appearance-based method and eye tracker. they show the estimated points of typing ten words, “guest”, “enterprise”, “seal”, “hike”, “until”, “account”, “normally”, “opportunity”, “charge”, “drag”. The blue dots denote points estimated by the appearance-based method, and the red dots denote points by eye tracker. The upper right box shows top-5 words recommended using estimated points of the method, and the lower right box shows top-5 words using points of the eye tracker.

the first calibration, the performance of recognition gets poorer over time, in which the main reason is the calibration drift, i.e. the variation of the status between typing words and the calibration phase is becoming larger. While doing the calibration every once in a while (recalibration), the drift can be controlled within limits, and the performance of recognition approximates the eye tracker as shown in figure 4.14a. Therefore, eye typing with the webcam is becoming feasible even under some degree of the calibration drift.

4.4 Discussion

The basic idea of appearance-based gaze estimation method is to infer the gaze coordinate of a new face image based on the calibration images and their corresponding gaze coordinates. In machine learning, it is a common supervised learning algorithm to solve the regression problem. In general, the class (continuous value) of a new sample is usually predicted in terms of the global mapping parameter from the sample space to the target space, in which the dimension of the sample space depends on the number of attributes, and the target space is usually one or two dimensions. The global mapping emphasizes the inter-relation of two spaces, but misses the intra-correlation within the sample space.

In the appearance-based gaze estimation method, all attributes are generated from pixels of eye image (figure 4.4), and they have a strong spatial correlation. As we mentioned before, these “image” points constitutes a 2D-surface manifold in the sample space (figure 4.6b), where the local linear combination is existing within the manifold. Moreover, the relative positions of points are consistent with the degree of eyeball rotation. The points in the sample space (high-dimensional space) might share the same local relation with the target space (low-dimension space, or 2-D gaze coordinate), which has been verified. In order to guarantee the same local relation, recently, Yu et al. [152] have further unified the structures of sample space and target space using a metric learning. Therefore, given a new eye image, the core step is to obtain the local combination parameter of its neighbouring training images, and then the optimal local parameter is employed to infer the new gaze coordinate using the combination of corresponding gaze coordinates of training images.

In general, the training images are collected through a calibration session. Since the subsequent gaze estimation relies on the training set, the quality of calibration is

very crucial for estimation. The different number of calibration points has been already studied [128, 17], in which the more calibration points (training images) usually improve the accuracy in subsequent gaze estimation. From the view of the sample space, the structure of manifold formed by training samples determines the gaze process. If we have enough training samples including all possible degrees of eyeball rotation, there would be no estimation error. In practice, it is not possible for a person to do the long-term calibration, which is burdensome. Since the range of the target space is usually finite, e.g. a screen, the boundary of the manifold is definite. Thus, the reference points of calibration are uniformly selected across the plane (figure 4.8b), which is a common method in calibration including existing commercial eye trackers.

Although the reference points in the target plane define the size and shape of manifold, the final manifold is formed by the appearance feature vectors of captured eye images. Thus, the quality of the eye images might also change the structure of manifold and then to affect the accuracy of gaze estimation. The traditional method is capturing the only one face image while the person is looking at each reference point, which might be deteriorated by uncontrollable artefacts. In order to reduce the impact of the artefacts, the averaging filter is employed to the multiple images captured at each reference point, which can improve the accuracy of estimation using the individual image (figure 4.10). As we mentioned above, the accuracy of the new sample estimation is the performance of the interpolation using training samples, which depends on the structure of the manifold. In figure 4.16, we plot the 3-D projection of the manifolds formed by individual images and average images using PCA. Although the structure of both manifolds is homogeneous, local texture is different. The manifold of the average images show a better approximation of the 2-D surface, and the interpolation becomes more accurate. Thus, the averaging filter is improving the estimation performance through smoothing the manifold, so that the linear combination of neighbouring sample is better supported. While the reference points in the target plane determine the coarse-grained shape of manifold, the eye images determine the fine-grained shape.

As we mentioned before, each dimension (attribute) of the sample space is described by the pixels of the eye image. In fact, since the eye image has the specific pattern, e.g. the relative distance between eye features, these dimensions have a strong spatial

4.16.a: all individual images

4.16.b: the average images

Figure 4.16: Eye-appearance manifolds in 3D space.

correlation. Lu et al. [128] and Yu et al. [152] proposed the method of dimensionality reduction that is dividing the eye image into several small blocks, and taking the sum of pixel intensities in each block, in which the dimension of the target space decreases to the number of blocks. Although the method might require a shorter calibration session, the accuracy of estimation decreases because of information loss. Therefore, to achieve dimensionality reduction without information loss, i.e. reducing the dimensions that are irrelevant to determine the gaze point, the intuition of appearance-based method need to be further studied. In figure 4.17, we plot the weight of directly mapping from the

Figure 4.17: The effective area

sample space (eye image) to the target space (gaze point). Although we do not use the global mapping to estimate gaze points, it might help us understand the rough relation between the appearance and gaze point. We randomly select 10 to 500 eye images ($\mathbf{X}$) of word typing with their corresponding gaze coordinates ($\mathbf{P}$, estimated points using the proposed method above). Since the dimension (2160) of the sample space is much larger than the number (k) of selected eye images, the global mapping function $\mathbf{wX}=\mathbf{P}$ is an underdetermined system, and L1 norm is used to obtain the weight $\mathbf{w}$ with k non-zero components which is highly related to determine gaze points $\mathbf{P}$. We will find that the area of eye orbit is contributing to the mapping, especially the edge area, e.g. eye lid and the boundary of iris. Thus, cropping other irrelevant pixels might not cause information loss, and also maintain the accuracy.

When we use the calibration images to estimate a new image, the condition is that these images should be in the same manifold, i.e. sharing the same local structure. However, if the head pose changes, this condition is not satisfied. The same reference point on the plane will correspond to different eye appearance images, and vice versa. Thus, under different head pose, the eye images will constitute a new manifold, in which the head pose is determining the location of manifold in the sample space, and the

eye image is determining the position of the corresponding appearance vector in the manifold [153].

While doing practical eye typing using the webcam, the critical issue is the calibration drift. Although the six sessions were consecutive and participants were required to keep the head stationary all the time, there might be also slight head motion. For the better understanding of head movement, We use the head tracker in OpenFace toolkit [154] to estimate the head pose of each face image, which outputs 6-dimensional head pose vectors. After processing the head pose vector, we can obtain the average head displacement of participants while typing each word in order as shown in figure 4.18. The displacement is increasing over the time, which causes the variation of eye images looking at the same coordinate. Since the relative position of the eye images in the manifold is still maintained, the new estimated points would demonstrate a drift of the actual points (figure 4.15).

In the eye-typing system, there are three common typing errors, extra-letter, neighbour-letter, and missing-letter errors. As the points are keeping drifting, more missing-letter errors will deteriorate the accuracy of the recognition algorithm. The purpose of recalibration (calibrate in each session) is to control the drift of the points updating the manifold under a new head pose. Thus, eye typing using webcam is feasible under a small degree of head movement.

However, there is still a challenging issue of eye typing using the webcam. To control the degree of the calibration drift, the system requires the person to do periodic recalibration, which brings extra burden. Zhang et al. [134] and Vadillo et al. [144] proposed the automatic recalibration methods based on the eye tracking data, but the methods are not suitable for the real-time system. Recently, along with the prevalence of deep learning, some research work also attempted to introduce deep neural network into gaze estimation training with millions of face images [129, 155]. Since the underlying head pose information has been automatically extracted from the big data set, it is possible that people do not even have to do calibration any more. However, it requires the high-performance machine, and our daily computer is not practical to train the model.

Besides the gaze estimation method, the recognition algorithm still need to improve to reduce the impact of the points drift. Since the relative position of points is maintained,

Figure 4.18: The displacement of head

the shape of gaze trace is homogeneous. Thus, it is possible to use shape-based approach which is comparing the shape of gaze trace of typing with the actual word shape [145, 111]. Moreover, the recognition algorithm we used in our work is only focusing on the input of the person. If a prediction engine can be integrated into the eye-typing system with semantic analysis and user historical typing behaviour, it would also improve the accuracy of word recommendation in practice.

4.5 Summary

In this work, we have proposed an eye-typing system using an off-the-shelf webcam to replace the eye tracker. First, we compared two types of gaze estimation methods using the webcam, in which the appearance-based method is more promising, because of its advantages in processing low-resolution images. We then found that the eye-appearance feature vectors would constitute a 2-D manifold embedded into the appearance space, and these feature vectors are clearly differentiated in the manifold. We further studied practical issues of the appearance-based method, where an averaging filter was proposed to improve the gaze estimation accuracy through reducing the impact of uncontrollable

artefacts. Through detecting the effective eye area determining the gaze variance, we achieved the purpose of dimensionality reduction without information loss. In the eye-typing experiment of actual users, although the proposed method is not as accurate as the commercial eye tracker, the robust typing algorithm still successfully recognized intended words according the estimated gaze points. With periodic recalibration controlling a small degree of head movement, the performance of word recognition using the webcam is approximating the eye tracker.

Chapter 5

Generic Gaze Estimation with Natural Head Movement

In this chapter, we present a novel reconstruction method for the appearance-based gaze estimation that allows inferring persons' gaze under natural head movement. We first study the locally linear combination in the respective manifolds consisting of stable left and right eye appearances is efficient. This local structure of the manifolds is destroyed when there is head movement. This is due to the destruction of the intrinsic relations between the two eyes (left and right) when we do locally linear combinations. We then introduce a new combination of both eye appearances, which maintains the intrinsic relation. Through comparison with other well known methods, we show that the proposed method achieves an optimal performance with head pose variation. In addition, we also investigate the correlation between eye gaze and mouse click, and demonstrate that there is high coordination while clicking menu buttons.

5.1 Estimation In Two Spaces

As discussed in Chapter 4, appearance manifold is a continuous set of appearance feature points embedding in the high dimensional space, and any point in the manifold can be linearly interpolated by its local neighbours. The assumption of appearance-based approaches is that eye-appearance feature points constitute a manifold in the high dimensional space (the appearance space), and its relative local interpolation is maintained in the low dimensional space, which has been well verified already.

Thus rather than global mapping directly from high-dimensional appearance data to the low-dimensional gaze coordinates, appearance-based gaze estimation is reconstructing the query appearance vector $\hat{\mathbf{x}}$ by the local linear combination of its neighbouring appearance vectors $\{\mathbf{x}_i\}$ in the appearance space. Then it infers the corresponding gaze point $\hat{\mathbf{p}}$ of $\hat{\mathbf{x}}$ by using the same linear combination of corresponding gaze points $\{\mathbf{p}_i\}$ of $\{\mathbf{x}_i\}$ in the gaze-coordinate space:

$$\begin{aligned} \mathbf{X}\mathbf{w} &= \hat{\mathbf{x}} \\ \mathbf{P}\mathbf{w} &= \hat{\mathbf{p}} \end{aligned} \quad (5.1)$$

where $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k\}$, $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_k\}$, and $\mathbf{w} = \{w_1, w_2, \dots, w_k\}$ denotes the locally linear combination, which avoids directly estimating the mapping between $\mathbf{X}$ and $\mathbf{P}$. However, since the equation is overdetermined, it contains estimation errors between the locally linear combination $\mathbf{X}\mathbf{w}$ and the query appearance vector $\hat{\mathbf{x}}$. Therefore, we have the optimization function with minimizing the estimation error:

$$\begin{aligned} \tilde{\mathbf{w}} &= \arg \min \left\| \hat{\mathbf{x}} - \sum_i^k w_i \mathbf{x}_i \right\| \quad s.t. \sum_i^k w_i = 1 \\ \hat{\mathbf{p}} &= \sum_i^k w_i \mathbf{p}_i \end{aligned} \quad (5.2)$$

where $\tilde{\mathbf{w}}$ is optimal locally linear combination parameter minimizing the linear combination error.

Then without loss of generality, let $\hat{\mathbf{x}}^L, \hat{\mathbf{x}}^R$ denote the left and right eye appearance vectors of the query face image; $\mathbf{X}^L, \mathbf{X}^R$ denote the neighbouring appearance vectors of $\hat{\mathbf{x}}^L$ and $\hat{\mathbf{x}}^R$; $\mathbf{w}^L, \mathbf{w}^R$ denote corresponding linear combination parameters. Thus, the actual reconstruction should be given by:

$$\begin{aligned} \tilde{\mathbf{w}}^L &= \arg \min \left\| \hat{\mathbf{x}}^L - \mathbf{X}^L \mathbf{w}^L \right\|, \quad s.t. \mathbf{1}^T \mathbf{w}^L = 1 \\ \tilde{\mathbf{w}}^R &= \arg \min \left\| \hat{\mathbf{x}}^R - \mathbf{X}^R \mathbf{w}^R \right\|, \quad s.t. \mathbf{1}^T \mathbf{w}^R = 1 \\ \hat{\mathbf{p}}^L &= \mathbf{P}^L \tilde{\mathbf{w}}^L, \hat{\mathbf{p}}^R = \mathbf{P}^R \tilde{\mathbf{w}}^R, \hat{\mathbf{p}} = (\hat{\mathbf{p}}^L + \hat{\mathbf{p}}^R)/2 \end{aligned} \quad (5.3)$$

where $\hat{\mathbf{p}}$ is averaging estimated gaze points of both left and right linear combination. Note that $\mathbf{X}^L$ and $\mathbf{X}^R$ are in different sets, so the functions are optimized in respective

spaces. Thus, let “Left”/“Right” denote the method using $\hat{\mathbf{p}}^L/\hat{\mathbf{p}}^R$ as the estimated point, and “Average” denote the method using average point $\hat{\mathbf{p}}$.

In the stable environment, e.g. no head movement, invariant illumination, the appearances of both eyes are relatively stable, and the two manifolds constituted by respective appearances would be homogeneous in their respective spaces. If any left appearance is a neighbour of the left query appearance, its corresponding right appearance should be also the neighbour of the right query appearance, and vice versa. That means that the pair of neighbours $(\{\mathbf{x}_i^L\}, \{\mathbf{x}_i^R\})$ are from the same face image $(\{\mathbf{f}_i\})$, i.e. $\forall i, \mathbf{x}_i^L \in \mathbf{f}_i$, then $\mathbf{x}_i^R \in \mathbf{f}_i$. Thus $\mathbf{P}^L = \mathbf{P}^R$, and then linear combination parameters $\mathbf{w}^L$ and $\mathbf{w}^R$ are approximate to each other. The relation of left and right appearance vector is not necessary to be modelled.

However, under natural head movement, the appearance images would be very different while looking at the same position under different head poses. Likewise, the two similar eye appearances might correspond to two different gaze points which are far away from each other. The left and right neighbouring appearances might not be from the same face images, which causes $\mathbf{P}^L \neq \mathbf{P}^R$. Thus potential head pose information described by the relation between the pair of left and right eyes is missing. The reconstruction error would be large using individual linear combination in respective spaces with head pose variation.

5.2 Spaces Alignment

In order to maintain the relationship between the pair of eyes, we should ensure that the neighboring appearances should be pairs of eyes from the same face image ($\mathbf{P}^L = \mathbf{P}^R$). In addition, the same local structure should be guaranteed, i.e. the same locally linear combination in the two appearance spaces:

$$\begin{aligned} \tilde{\mathbf{w}} &= \arg \min \left\| \hat{\mathbf{x}}^L - \mathbf{X}^L \mathbf{w} \right\| + \arg \min \left\| \hat{\mathbf{x}}^R - \mathbf{X}^R \mathbf{w} \right\| \\ \text{s.t. } \quad & \mathbf{1}^T \mathbf{w} = 1; \forall i, \mathbf{x}_i^L \in \mathbf{f}_i, \text{ then } \mathbf{x}_i^R \in \mathbf{f}_i \\ & \hat{\mathbf{p}} = \mathbf{P} \tilde{\mathbf{w}} \end{aligned} \tag{5.4}$$

where $\mathbf{X}^L$ and $\mathbf{X}^R$ contain left and right eye-pair appearances of the same face images respectively, and the number of the neighboring appearances are the same. Thus the

above equation(5.4) can be simplified to:

$$\begin{aligned}\tilde{\mathbf{w}} &= \arg \min \left\| \hat{\mathbf{x}}^E - \mathbf{X}^E \mathbf{w} \right\| \quad s.t. \mathbf{1}^T \mathbf{w} = 1; \\ \hat{\mathbf{p}} &= \mathbf{P} \tilde{\mathbf{w}}\end{aligned}\tag{5.5}$$

where $\hat{\mathbf{x}}^E = \begin{pmatrix} \hat{\mathbf{x}}^L \\ \hat{\mathbf{x}}^R \end{pmatrix}$, $\mathbf{X}^E = \begin{pmatrix} \mathbf{X}^L \\ \mathbf{X}^R \end{pmatrix}$. As we can see, while solving the optimization equation, this combination is the same with concatenating left and right appearance vectors together, which unifies left and right appearance spaces into the same space, so that the relation of both eye images are implicitly modelled in the same locally linear combination. Thus, let “Concatenate” denote the proposed method.

We also study another two methods applying the basic mathematical operation to the left and right vectors. In equation 5.5, $\hat{\mathbf{x}}^E = (\hat{\mathbf{x}}^L - \hat{\mathbf{x}}^R)$, $\mathbf{X}^E = (\mathbf{X}^L - \mathbf{X}^R)$, which is denoted as “Minus”. $\hat{\mathbf{x}}^E = (\hat{\mathbf{x}}^L + \hat{\mathbf{x}}^R)$, $\mathbf{X}^E = (\mathbf{X}^L + \mathbf{X}^R)$, which is denoted as “Plus”. The methods can be regarded as the dimensionality reduction of “Concatenate” where it reduces half reconstruction space.

5.3 Experiment

In order to evaluate the gaze estimation performance of our method with head movement, we conducted an experiment where all the subjects are working in their normal working environment, and collected valid samples with natural head movement. In the experiment, our objectives are two fold: 1). Performance comparison of our proposed method with other methods. 2). Impact of the degree of head pose variation on performance.

5.3.1 Data Collection

We developed the data collection application to run on a desktop computer with a 21-inch LED-lit monitor attaching an off-the-shelf webcam (30fps), and the eye tracker (Tobii EyeX, 60Hz) was put on the bottom of the monitor. Five right-handed volunteers (2 male, 3 female) from the local university participated in the experiment. To guarantee natural head movement, the system was installed on their own computer in the office, so they worked on their daily work as usual with free movement of head and body, and

Figure 5.1: Distribution of head pose and mouse click

they were also allowed to leave and come back. The data collection experiment of each participant covers a period of two days.

The experiment procedure is as follows: the participants first carried out the calibration with the eye tracker. Later, the gaze points estimated by the eye tracker were taken as validation data. The mouse clicks were taken as the trigger event. Once the participant clicked the left mouse button, the system simultaneously recorded the frontal face image captured by the webcam, the current gaze point estimated by the eye tracker, the mouse click position, and the screenshot of a window (200×200 pixels) centered on the click point (the screenshots are discussed in next section). If the distance of mouse click position and estimated gaze point is within 80 pixels (the eye tracker accuracy is around $2^\circ \sim 3^\circ$ with head movement), the pair of face image and the mouse click position was selected as a valid sample. In total, we collected one thousand valid samples for each participant.

To demonstrate the head variability of the participants, we used the method from [154] to estimate the head pose based on the face images. Figure 5.1.a shows the distribution of head pose that indicates natural head movement; figure 5.1.b shows that mouse clicks distribute among the large area of the screen.

Then the eye-area images were cropped at the same fixed aspect ratio as shown in figure 5.3.a. Finally the feature vector of the eye appearance is generated by raster scan

Figure 5.2: The evaluation results. “Left”, “Right” indicates only left or right eye images is used; “Average” indicates using average estimated points of both left and right eye images; “Concatenate” indicates our proposed method; “Minus”/“Plus” indicates left eye-image vector minus or plus right eye-image vector.

of the eye intensity images. The leave-one-out cross validation was employed, and the performance was evaluated by the mean estimated angular error as:

$$error = \frac{1}{n} \sum_{i=1}^n \arctan\left(\frac{\|\hat{p}_i - p_i\|_2}{d}\right) \quad (5.6)$$

where $\|\hat{p}_i - p_i\|$ is the Euclidean distance between estimated gaze position $\hat{p}_i$ and actual gaze position p_i . d is the distance between participant’s eyes and the screen. In the experiment, we take the average distance as 60cm).

5.3.2 Results

Figure 5.2 shows the evaluation results, where x-axis indicates different methods in each subject, and y-axis indicates the estimated angular error. There are some observations: 1). Although the performance of using the only left or right eye images has big gap on subject 1, it does not show significant difference among subjects ($p\text{-value} = 0.3981 > 0.05$), but averaging is always better than using the only one eye images. 2). for applying the mathematical operation on eye-image vector, the plus operation could slightly decrease

Figure 5.3: Eye area cropping and the image effect applying the minus/plus operation

the error, but the minus operation leads to larger error. 3). our proposed method always shows the lowest error, which outperforms averaging method by 12.97% to 45.28% ($p\text{-value} = 0.0082 < 0.05$).

These observations help us understand the intuition of the appearance-base method. “Average” can be regarded as a post-combination method, where it does not process the original image vectors, but deal with the estimation results of both eye images. The method is capable of decreasing the error by reducing the bias of the respective reconstruction spaces, but it is losing the local relation between left and right eyes spaces. Thus, the pre-combination method is restricting the local relation of both eye images by processing the original image vectors, i.e. “Minus” and “Plus”, in which the two individual spaces are integrated to the same space. However, such mathematical operations could affect the pixel-level information of both eye images, which causes the different performance. In order to better understand the impact of the operators, we plot the resultant images in figure 5.3.b (normalized to 0-255). As we can see, after doing the “minus” operation, the image becomes blurred and not clearly recognized. The “plus” operation enhances the same and reduces the difference, so that the pupil area which is related to gaze becomes more prominent. Our proposed method is not pre-combination (original vector processing) or post-combination (estimated gaze points averaging), but an embedding combination which is to learn the “hidden” relation between left and right eye images better matching the space mapping with head movement.

In order to further analyze the performance of different combinations, we investigated the impact of degree of head poses variation. we used a head tracker in the toolkit [154].

The tracker estimated the head pose of each face image, and output 6-dimensional head pose vector, $\mathbf{h} = \{t_x, t_y, t_z, r_x, r_y, r_z\}$, which indicate the location of head with respect to camera and rotation around X,Y,Z axes with the convention $r = r_x * r_y * r_z$. Thus, let $\mathbf{g} = \{\mathbf{h}, \mathbf{x}\}$ denote the vector of gaze by adding head pose information. As we discussed before, the head pose variation would destroy the structure of manifold formed by eye appearance feature points. Also, head appearance points constitute a manifold in head space which shows the local linearity [14]. Under similar head poses, i.e. $\hat{\mathbf{h}} \approx \{\mathbf{h}_i\}$, the corresponding eye manifolds would have more similar structure. i.e. better reconstruction. Thus, given a new eye image, the first step is searching the nearest cluster in which head poses are similar, and then using the gaze estimation method in the cluster. The head-pose similarity is calculated using Euclidian distance of corresponding vectors, and the K-means algorithm is used to do the clustering task (Algorithm 6).

Algorithm 6 Gaze Estimation with Similar Head Pose

```

1: function GetEstimatedGazePoint( $G, k, \hat{g}$ )
2:    $\{h_i, x_i\} \leftarrow G; (\hat{h}, \hat{x}) \leftarrow \hat{g};$ 
3:   Let  $C_1, C_2, \dots, C_k$  denote centroids of the  $k$  clusters;
4:   Initialize  $C_1, C_2, \dots, C_k$  using random  $k$  head pose vectors  $\{h_i\}$ ;
5:   repeat
6:     for  $j$  in size( $G$ )
7:        $n = \arg \min_n \|h_j - C_n\|;$ 
8:       Add  $h_j$  into Cluster  $n$ ;
9:     end for
10:    Update  $C_1, C_2, \dots, C_k$ ;
11:  until  $C_1, C_2, \dots, C_k$  do not change.
12:   $m = \arg \min_m \|\hat{h} - C_m\|;$ 
13:  select all  $\{x_m\}$  from  $C_m$ ;
14:   $\hat{p} = \text{gazeEstimation}(\{x_m\}, \hat{x});$ 
15: end function
    
```

In the experiment, we iteratively set k from 1 to 10. Given a new face image, the gaze point is estimated in the closest cluster. Figure 5.4 shows the evaluation results of subject 4. When $k = 1$, it means all samples are used in estimation, and the result is the same as figure 5.2. As k increases, i.e. controlling the head pose gradually, the error first drops dramatically, and then taper off. As we can see, the relative performance

Figure 5.4: The degree of head pose variation

of these combinations does not change, but “Average” is approximating “Concatenate”, which verified the above statement that with relatively stable appearance, the two manifolds constituted by left and right eye appearances would be homogeneous, so that the “Average” method is able to handle it. While head pose variation is large ($K < 3$), our proposed method is still better. These observations verify the above statement that gaze estimation under similar head poses has better performance, because the local structure of manifold is approximating. When k is larger, the estimation error will be smaller in theory, but the number of samples in each cluster is becoming too small to reconstruct the new eye image location.

5.4 Correlation Between Gaze and Mouse

In this section, we will investigate the correlation between eye gaze and interaction cues, such as mouse cursor. In general, there are two opinions on the coordination of gaze fixations and mouse clicks, 1). the user is looking at the cursor position while clicking [153, 156]; 2). the cursor lags behind gaze [157]. Based on the first opinion, the pairs of frontal face images and mouse-click position are always recorded. Supporting the second opinion, it needs to record the video that verify the best-match time point between gaze and click event, e.g. around 100ms to 300ms preceding each event. Although the method is reasonable, the video record is too heavy for daily work. In addition, extracting and

Figure 5.5: The paradigm of data collection.

verifying 3-second data preceding every click is inefficient in practice. Moreover, the both opinions make the same assumption that the gaze and cursor align perfectly during the click event, and only difference is time gap. However, the prior assumption might be wrong in real-use cases, where some clicks do not require the user's gaze, i.e. the user is not looking at the cursor position. Therefore, in the above experiment, the screenshots of a window (200×200 pixels) centered on the click point, as well as mouse events were also recorded simultaneously that helps to investigate the correlation of gaze and click event.

The standard mouse interaction typically include six basic events, movement (M), left button down (LBD), left button up (LBU), right button down (RBD), right button up (RBU), and wheel (W). Once one of above mouse events occurred, the data were recorded using the format (gaze position, mouse position, mouse event name, time stamp), where gaze position is provided by Tobii EyeX. In addition, once the left-button-down event occurred, the frontal face image captured by the webcam and the screenshot were collected, which is consistent with the Chapter 5.3.1. The paradigm of the whole data collection is shown in figure 5.5. In order for more meaningful analysis, we summarize the six basic mouse events to six common-use interaction events (figure 5.6). Except movement and wheel events, other four events cannot be an individual event, and they always occur with others sequentially. For example, the button-down event is followed by the button-up event. In figure 5.6, there are three interaction events containing left button down/up,

Figure 5.6: The combination of mouse events. LBD is left button down; LBU is left button up; RBD is right button down; RBU is right button up; M is movement.

left click, double click, drag, which are commonly used to study the coordination of eye gaze and mouse click.

Moreover, besides the mouse interaction events, we also analyze the event context further based on the screenshots, and define another type of click event, i.e. menu-like events. Since the context categorization is subjective, we try to abstract them. These events have a similar pattern that people should pay more attention to them, e.g. close/minimize/maximize a window, add/close a tab in browser, click a shortcut in the desktop or taskbar, confirm pop-up warning dialogue (figure 5.7). while the gaze point is within the accuracy scope of the eye tracker, the corresponding event is labelled as alignment. Figure 5.8 shows the comparison of these events. There are several observations verifying our statement that the coordination assumption of gaze and mouse could be wrong. The LBD events have low alignment percentage, e.g. the percentage of Subject 5 is even lower than 50%. Thus, taking all pairs of gaze and click as the sample data of auto calibration might cause large estimation error in practice. In this work, we briefly investigated the click context, where the alignment of menu-like clicks is 10-20% higher than other LBD events. Although the investigation results do not show which specific click content has the highest alignment percentage, it can be a potential research direction using the correlation of gaze and mouse click in the auto-calibration area. In

Figure 5.7: Some screenshot samples. The blue dot and red circle (the innermost circle) indicate the actual gaze point and mouse click position, respectively; The green and cyan circles show the accuracy scope of the eye tracker.

addition, there is one more observation that we found in the experiment. The correlation is highly depending on subject's proficiency, concentration, or even habit. Thus, in order to obtain more accurate samples, customization might be required in practice.

Figure 5.8: The alignment percentage of click events.

5.5 Summary

In this chapter, we have presented a new reconstruction method that is able to robust to head movement in appearance-based gaze estimation. The proposed method is embedding into the reconstruction stage, aligning the left and right appearance spaces into the same local structure, which is able to maintain the relation of both eyes appearance under head movement. In addition, we conducted an experiment to investigate the performance of different combinations of eye images. The experiment results demonstrate the proposed method outperforms others significantly. The observations of the results also provide new sights of intuition of the appearance-based method, which could help to extract efficient eye appearance features even in deep learning.

Chapter 6

Conclusion and Future Work

In this chapter, we conclude the thesis and propose several directions to continue our research.

6.1 Conclusion

In order to increase the communication ability of physically challenged people, a number of eye-based typing systems have been proposed, but most have been plagued by slow text entry speed. Recently, dwell-free systems have been proposed, but they still are vulnerable to common text entry errors. The user may gaze at extra irrelevant letters accidentally, and the Filtered typing method has overcome this problem by filtering out the extra letters. However, in practice, the low accuracy or drift problem caused by inaccurate calibration is another critical issue. The user might only gaze at an adjacent letter (neighbour-letter error), and even might not gaze at all letters of the word (missing-letter error). Then we proposed the MoWing method to improve its performance in neighbour-letter and missing-letter errors cases while sacrificing some degree of performance in extra-letter error case. However, due to the Midas Touch Problem, the extra-letter error is the most common error encountered in eye typing, and the overall performance of MoWing is not superior in practice.

Therefore, we then proposed a robust dwell-free recognition approach, LCSMapping, to handle the neighbour-letter and missing-letter errors without sacrificing the performance in the extra-letter error case. Unlike simple filtering and string matching methods, the proposed approach takes the letter location and gaze duration into account besides

the gazed letters, which has made dwell-free typing even more accurate and more resilient to text entry error, much akin to finger-swipe typing, even using a low-cost eye-tracking device. Furthermore, implementing the tree structure to process the recommendation, the operation time of LCSMapping is reduced by 11% and 46% respectively, compared with Filteredping (FdpRevised) and MoWing.

However, the additional eye tracker is still a necessary device of the eye-typing system, which might be burdensome in some case. Since webcam is becoming a standard component in computers, especially in mobile devices, replacing the eye tracker with a webcam would simplify the setup of eye-typing equipment, but also facilitate the prevalence of eye-tracking applications. Therefore, in our work, we have proposed an eye-typing system using an off-the-shelf webcam to replace the eye tracker. First, we compared two types of gaze estimation methods using the webcam, in which the appearance-based method is more promising, because of its advantages in processing low-resolution images. We then found that the eye-appearance feature vectors would constitute a 2-D manifold embedded into the appearance space, and these feature vectors are clearly differentiated in the manifold. We further studied practical issues of the appearance-based method, where an averaging filter was proposed to improve the gaze estimation accuracy through reducing the impact of uncontrollable artefacts. Through detecting the effective eye area determining the gaze variance, we achieved the purpose of dimensionality reduction without information loss. In the eye-typing experiment of actual users, although the proposed method is not as accurate as the commercial eye tracker, the robust typing algorithm still successfully recognized intended words according the estimated gaze points. With periodic recalibration controlling a small degree of head movement, the performance of word recognition using the webcam is approximating the eye tracker.

Eye tracking has broader and more generic users, not only disable people when using webcam. In this scope, eye gaze can be an additional useful input for them to use applications. Thus, head movement is a critical issue that should be also considered in eye gaze estimation. We presented a novel reconstruction method for the appearance-based gaze estimation that allows inferring persons' gaze under natural head movement. We first studied that the locally linear combination in the respective manifolds consisting of stable left and right eye appearances is efficient. The local structure of the manifolds is

destroyed when there is head movement. This is due to the destruction of the intrinsic relations between the left and right eyes while doing locally linear combinations. We then introduced a new combination of both eye appearances, which maintains the relation embedding into the reconstruction of the training stage. Through comparison with other well known methods, we showed that the estimation accuracy of proposed method outperforms other methods by 12.97% to 45.28%.

6.2 Future Work

In future work, there are three potential areas of eye tracking research, data collection, model generation, and system deployment. Currently the most challenging issue of gaze estimation is lacking standard training dataset. Although there are several open-source data sets [158, 159, 160, 161, 129], they are suffering the following drawbacks, e.g. small number of data, under a controlled indoor laboratory, limited variation of head pose, etc. In addition, the quality of collected data is also critical, because it is difficult (or impossible) to validate the correctness of recording data manually, i.e. whether a captured face image exactly corresponds the given target point. It is different with other data sets, e.g. for object detection/recognition, labels of all recording images can be annotated manually any time after the data collection session. However, the pair of gaze point and face image can be only recorded together during the collection session, and post-annotation is impossible. The existing works do not have a validation step in the data collection session so that there might be large bias in the training samples. Thus, it is necessary to develop a reliable data-collection tool to guarantee a high-quality of dataset.

Although we have done preliminary work on head-pose variation, it is still a challenging issue, and the modelling of both head and eye poses requires the much larger search space. Recently, there are few attempts to handle the head-pose variations [129, 155], where both of them apply convolution neural network (CNN) to take eyes image as input. However, [129] can be regarded as a hybrid model of feature-based and appearance-based approaches, where head pose vectors that are estimated by face alignment are added to the output of the fully connected layer. The approach has no difference with previous

Figure 6.1: An overview of static model

learning-based methods except applying a deeper neural network. [155] is the end-to-end model using pure appearance-based method with AlexNet, but there are two problems, 1). The model is lacking solid interpretation; 2). The model and dataset have not been released yet. More importantly, all those works do not have fine-grained analysis of the appearance which is determining the gaze points, i.e. which is effective area. Although with CNN and enough samples, the intrinsic features would be extracted automatically, the analysis of effective area might benefit the trained model, e.g. reduce the training cost, avoid overfitting, etc. Therefore, the effective eye areas that contribute the eye-appearance various and finally determine the gaze estimation still need to be investigated, and generating different models to solve fixed and various head pose respectively depending on accuracy requirements of different applications. Therefore, in the future, we will integrate the eye tracking system with two models, static model and dynamic model. The two kinds of models will benefit different demands of target users, in which the static model (fixed head pose) is providing fine-grained estimation for a subset of people (e.g. motor impairment), and the dynamical model (free head pose) is for general people.

Static model. As discussed in Chapter 4, we use locally linear embedding [14] method to build the static model as shown in figure 6.1. Each pixel of the image is described as one dimension in the appearance space, so that the entire image can be regarded as a feature point in the space. When the head pose is fixed, the corresponding points of eye images will form a manifold embedding in the appearance space that has an approximately 2-D surface, and the relative local interpolation is maintained inside. The proposed approach is reconstructing the query appearance vector (eye image) by the

Figure 6.2: An overview of dynamic model

local linear combination of its neighboring vectors, and then inferring the corresponding gaze point by using the same linear combination of gaze points of neighboring vectors. Since this approach does not directly model the hidden relation between eye images and gaze coordinates, small number of samples is enough to train the model.

Dynamic model. Instead of local learning in the static model, we propose a global learning approach (end-to-end learning) using multimodal convolutional neural networks (CNN) as shown in figure 6.2. In the model, the input is only face images which is different with [129, 155], i.e. we do not need to extract eye-area images. Besides traditional convolutional layers and final mapping layer, we add an extra layer called spatial mask layer [162]. In this layer, the effective area of face image will be extracted which is determining the gaze variation. With the spatial mask, this model can suppress irrelevant regions, and enhance subtle regions that represent gaze status, as well as head pose information.

The high accuracy and real-time capability are two main requirements of a practical gaze-estimation system. Most existing works mainly focus on the accuracy issue, and evaluate the trained model testing off-line images, but usually ignore the second requirement so that their models cannot be applied in real time directly, especially for mobile

Figure 6.3: An overview of system deployment

devices. Thus, to satisfy the requirement of real-time interaction applications, deploying a light-weight model in the client is necessary. In addition, extensibility should be also considered. Since the user of the eye-tracking application is relatively fixed, e.g. personal smart phone and computer, the system should be able to allow the user to personalize the pre-trained model, and keep tuning the parameters of the model. Thus, the pre-trained model can be a generic model, and then the end-users keep training a personalized light-weight model in the client as shown in figure 6.3.

References

- [1] U. of Maine System, “What is computing for the disabled? available at: <https://sites.google.com/a/maine.edu/computers-for-the-disabled/>.” Accessed: 2017-03-01.
- [2] C. H. Morimoto and M. R. Mimica, “Eye gaze tracking techniques for interactive applications,” *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 4–24, 2005.
- [3] E. Silversmith, “Light from a single point of an object. available at: [https://en.wikipedia.org/wiki/lens_\(anatomy\)](https://en.wikipedia.org/wiki/lens_(anatomy)).” Accessed: 2017-04-01.
- [4] M. F. Land, “Eye movements and the control of actions in everyday life,” *Progress in retinal and eye research*, vol. 25, no. 3, pp. 296–324, 2006.
- [5] R. Valenti and T. Gevers, “Accurate eye center location through invariant isocentric patterns,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 9, pp. 1785–1798, 2012.
- [6] J. Gips and P. Olivieri, “Eagleeyes: An eye control system for persons with disabilities,” in *The Eleventh International Conference on Technology and Persons with Disabilities*, pp. 1–15, 1996.
- [7] A. Duchowski, *Eye tracking methodology: Theory and practice*, vol. 373. Springer Science & Business Media, 2007.
- [8] A. Penkar, *Hypertext Navigation with an Eye Gaze Tracker*. PhD thesis, ResearchSpace@ Auckland, 2014.
- [9] C. Matters, “E-tran frames. available at: <http://www.communicationmatters.org.uk/page/e-tran-frames>.” Accessed: 2017-04-01.
- [10] 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*, pp. 11–18, ACM, 2008.

REFERENCES

- [11] D. J. Ward and D. J. MacKay, “Artificial intelligence: Fast hands-free writing by gaze direction,” *Nature*, 2002.
- [12] S. Sarcar, P. Panwar, and T. Chakraborty, “Eyek: an efficient dwell-free eye gaze-based text entry system,” in *Proceedings of the 11th Asia Pacific Conference on Computer Human Interaction*, pp. 215–220, ACM, 2013.
- [13] K. Sengupta, R. Menges, C. Kumar, and S. Staab, “Gazethekey: Interactive keys to integrate word predictions for gaze-based text entry,” in *Proceedings of the 22nd International Conference on Intelligent User Interfaces Companion*, pp. 121–124, ACM, 2017.
- [14] S. T. Roweis and L. K. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
- [15] P. Majaranta, H. Aoki, M. Donegan, D. W. Hansen, and J. P. Hansen, *Gaze Interaction and Applications of Eye Tracking: Advances in Assistive Technologies*. Hershey, PA: Information Science Reference - Imprint of: IGI Publishing, 1st ed., 2011.
- [16] J. R. Duffy, *Motor speech disorders: Substrates, differential diagnosis, and management*. Elsevier Health Sciences, 2013.
- [17] F. Lu, Y. Sugano, T. Okabe, and Y. Sato, “Adaptive linear regression for appearance-based gaze estimation,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 10, pp. 2033–2046, 2014.
- [18] G. Underwood, *Cognitive processes in eye guidance*. Oxford University Press, 2005.
- [19] P. Org, “Consumer product giants’ eye-trackers size up shoppers. available at: <https://phys.org/news/2012-07-consumer-product-giants-eye-trackers-size.html>.” Accessed: 2017-03-01.
- [20] P. Majaranta, U.-K. Ahola, and O. Špakov, “Fast gaze typing with an adjustable dwell time,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 357–360, ACM, 2009.
- [21] M. H. Urbina and A. Huckauf, “Alternatives to single character entry and dwell time selection on eye typing,” in *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, pp. 315–322, ACM, 2010.
- [22] I. S. MacKenzie and X. Zhang, “Eye typing using word and letter prediction and a fixation algorithm,” in *Proceedings of the 2008 symposium on Eye tracking research & applications*, pp. 55–58, ACM, 2008.

REFERENCES

- [23] K. Kotani, Y. Yamaguchi, T. Asao, and K. Horii, “Design of eye-typing interface using saccadic latency of eye movement,” *International Journal of Human-Computer Interaction*, vol. 26, no. 4, pp. 361–376, 2010.
- [24] K.-J. Rih and S. Ovaska, “An exploratory study of eye typing fundamentals: dwell time, text entry rate, errors, and workload,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 3001–3010, ACM, 2012.
- [25] R. J. Jacob, “Eye tracking in advanced interface design,” *Virtual environments and advanced interface design*, pp. 258–288, 1995.
- [26] J. P. Hansen, K. Trning, A. S. Johansen, K. Itoh, and H. Aoki, “Gaze typing compared with input by head and hand,” in *Proceedings of the 2004 symposium on Eye tracking research & applications*, pp. 131–138, ACM, 2004.
- [27] P. O. Kristensson and K. Vertanen, “The potential of dwell-free eye-typing for fast assistive gaze communication,” in *Proceedings of the Symposium on Eye Tracking Research and Applications*, pp. 241–244, ACM, 2012.
- [28] R. S. Longhurst, *Geometrical and physical optics*. Orient BlackSwan, 1970.
- [29] B. Cassin, S. Solomon, and M. L. Rubin, *Dictionary of eye terminology*. Triad Pub. Co., 1984.
- [30] M. D. Fairchild, *Color appearance models*. John Wiley & Sons, 2013.
- [31] J. H. Krantz, “Chapter 3: The stimulus and anatomy of the visual system,” *Experiencing Sensation and Perception*. Pearson Education. ISBN, pp. 978–0, 2012.
- [32] R. H. Carpenter, *Movements of the Eyes, 2nd Rev.* Pion Limited, 1988.
- [33] E. D. Guestrin and M. Eizenman, “General theory of remote gaze estimation using the pupil center and corneal reflections,” *IEEE Transactions on biomedical engineering*, vol. 53, no. 6, pp. 1124–1133, 2006.
- [34] D. Purves, G. Augustine, and D. Fitzpatrick, *Types of Eye Movements and Their Functions, Neuroscience. 2nd edition*. Sunderland (MA), Sinauer Associates, 2001.
- [35] L. R. Young and D. Sheena, “Survey of eye movement recording methods,” *Behavior Research Methods*, vol. 7, no. 5, pp. 397–429, 1975.
- [36] S. Martinez-Conde, S. L. Macknik, and D. H. Hubel, “The role of fixational eye movements in visual perception,” *Nature Reviews Neuroscience*, vol. 5, no. 3, pp. 229–240, 2004.

REFERENCES

- [37] G. D. Horwitz and T. D. Albright, “Short-latency fixational saccades induced by luminance increments,” *Journal of Neurophysiology*, vol. 90, no. 2, pp. 1333–1339, 2003.
- [38] N. Sheahan, D. Coakley, F. Hegarty, C. Bolger, and J. Malone, “Ocular microtremor measurement system: design and performance,” *Medical and Biological Engineering and Computing*, vol. 31, no. 3, pp. 205–212, 1993.
- [39] R. Ditchburn and B. Ginsborg, “Involuntary eye movements during fixation,” *The Journal of physiology*, vol. 119, no. 1, p. 1, 1953.
- [40] N. N. Rommelse, S. Van der Stigchel, and J. A. Sergeant, “A review on eye movement studies in childhood and adolescent psychiatry,” *Brain and cognition*, vol. 68, no. 3, pp. 391–414, 2008.
- [41] S. Klingenhoefer and F. Bremmer, “Saccadic suppression of displacement in face of saccade adaptation,” *Vision research*, vol. 51, no. 8, pp. 881–889, 2011.
- [42] P. Wang, M. B. Green, Q. Ji, and J. Wayman, “Automatic eye detection and its validation,” in *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, pp. 164–164, IEEE, 2005.
- [43] Z.-H. Zhou and X. Geng, “Projection functions for eye detection,” *Pattern recognition*, vol. 37, no. 5, pp. 1049–1056, 2004.
- [44] Z. Zhu and Q. Ji, “Robust real-time eye detection and tracking under variable lighting conditions and various face orientations,” *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 124–154, 2005.
- [45] S.-T. Kim, K.-A. Choi, Y.-G. Shin, and S.-J. Ko, “A novel iris center localization based on circle fitting using radially sampled features,” in *Consumer Electronics (ISCE), 2015 IEEE International Symposium on*, pp. 1–2, IEEE, 2015.
- [46] E. Skodras and N. Fakotakis, “Precise localization of eye centers in low resolution color images,” *Image and Vision Computing*, vol. 36, pp. 51–60, 2015.
- [47] D. W. Hansen and Q. Ji, “In the eye of the beholder: A survey of models for eyes and gaze,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 3, pp. 478–500, 2010.
- [48] S. B. and D. P., “Non-intrusive gaze tracking using artificial neural networks,” in *NIPS*, vol. 6, 1994.

REFERENCES

- [49] Y. Cheung and Q. Peng, “Eye gaze tracking with a web camera in a desktop environment,” *THMS*, vol. 45, no. 4, pp. 419–430, 2015.
- [50] K. Krafska, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, “Eye tracking for everyone,” in *CVPR*, 2016.
- [51] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.
- [52] A. Pérez, M. L. Córdoba, A. Garcia, R. Méndez, M. Munoz, J. L. Pedraza, and F. Sanchez, “A precise eye-gaze detection and tracking system,” 2003.
- [53] K.-N. Kim and R. Ramakrishna, “Vision-based eye-gaze tracking for human computer interface,” in *Systems, Man, and Cybernetics, 1999. IEEE SMC’99 Conference Proceedings. 1999 IEEE International Conference on*, vol. 2, pp. 324–329, IEEE, 1999.
- [54] W. Fuhl, T. Kübler, K. Sippel, W. Rosenstiel, and E. Kasneci, “Excuse: Robust pupil detection in real-world scenarios,” in *International Conference on Computer Analysis of Images and Patterns*, pp. 39–51, Springer, 2015.
- [55] D. Young, H. Tunley, and R. Samuels, *Specialised hough transform and active contour methods for real-time eye tracking*. University of Sussex, Cognitive & Computing Science, 1995.
- [56] H.-I. Kim, J.-B. Kim, J.-E. Lee, T.-Y. Lee, and R.-H. Park, “Gaze estimation using a webcam for region of interest detection,” *Signal, Image and Video Processing*, vol. 10, no. 5, pp. 895–902, 2016.
- [57] W. Fuhl, T. C. Santini, T. Kübler, and E. Kasneci, “Else: Ellipse selection for robust pupil detection in real-world environments,” in *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, pp. 123–130, ACM, 2016.
- [58] A. L. Yuille, D. S. Cohen, and P. W. Hallinan, “Feature extraction from faces using deformable templates,” in *Computer Vision and Pattern Recognition, 1989. Proceedings CVPR’89., IEEE Computer Society Conference on*, pp. 104–109, IEEE, 1989.
- [59] K.-M. Lam and H. Yan, “Locating and extracting the eye in human face images,” *Pattern recognition*, vol. 29, no. 5, pp. 771–779, 1996.

REFERENCES

- [60] F. J. S. Carvalho and J. M. R. da Silva Tavares, “Eye detection using a deformable template in static images,” *VIPimage 2007*, 2008.
- [61] J. Alabort-i Medina, B. Qu, and S. Zafeiriou, “Statistically learned deformable eye models,” in *Workshop at the European Conference on Computer Vision*, pp. 285–295, Springer, 2014.
- [62] R. Kothari and J. L. Mitchell, “Detection of eye locations in unconstrained visual images,” in *Image Processing, 1996. Proceedings., International Conference on*, vol. 3, pp. 519–522, IEEE, 1996.
- [63] F. Timm and E. Barth, “Accurate eye centre localisation by means of gradients,” *VISAPP*, vol. 11, pp. 125–130, 2011.
- [64] R. Valenti and T. Gevers, “Accurate eye center location and tracking using isophote curvature,” in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, IEEE, 2008.
- [65] J. Lichtenauer, E. Hendriks, and M. Reinders, “Isophote properties as features for object detection,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2, pp. 649–654, IEEE, 2005.
- [66] J. Yang, R. Stiefelhagen, U. Meier, and A. Waibel, “Real-time face and facial feature tracking and applications,” in *International Conference on Auditory-Visual Speech Processing*, 1998.
- [67] S. Kawato and J. Ohya, “Two-step approach for real-time eye tracking with a new filtering technique,” in *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, vol. 2, pp. 1366–1371, IEEE, 2000.
- [68] P. W. Hallinan, “Recognizing human eyes,” in *San Diego, '91, San Diego, CA*, pp. 214–226, International Society for Optics and Photonics, 1991.
- [69] K. Grauman, M. Betke, J. Gips, and G. R. Bradski, “Communication via eye blinks-detection and duration analysis in real time,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 1010–1017, IEEE, 2001.
- [70] J. Huang, D. Ii, X. Shao, and H. Wechsler, “Pose discrimination and eye detection using support vector machines (svm),” in *Face Recognition*, pp. 528–536, Springer, 1998.

- [71] Z. Zhu, K. Fujimura, and Q. Ji, “Real-time eye detection and tracking under various light conditions,” in *Proceedings of the 2002 symposium on Eye tracking research & applications*, pp. 139–144, ACM, 2002.
- [72] P. Viola and M. J. Jones, “Robust real-time face detection,” *International journal of computer vision*, vol. 57, no. 2, pp. 137–154, 2004.
- [73] I. Fasel, B. Fortenberry, and J. Movellan, “A generative framework for real time object detection and classification,” *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 182–210, 2005.
- [74] Y. Sun, X. Wang, and X. Tang, “Deep convolutional network cascade for facial point detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3476–3483, 2013.
- [75] Z. Zhang, P. Luo, C. C. Loy, and X. Tang, “Facial landmark detection by deep multi-task learning,” in *European Conference on Computer Vision*, pp. 94–108, Springer, 2014.
- [76] S. Yang, P. Luo, C.-C. Loy, and X. Tang, “From facial parts responses to face detection: A deep learning approach,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3676–3684, 2015.
- [77] R. W. Baloh and K. Kerber, *Baloh and Honrubia’s clinical neurophysiology of the vestibular system*, vol. 77. Oxford university press, 2010.
- [78] W. Heide, E. Koenig, P. Trillenber, D. Kömpf, and D. Zee, “Electrooculography: technical standards and applications,” *Electroencephalogr Clin Neurophysiol Suppl*, vol. 52, pp. 223–240, 1999.
- [79] A. E. Kaufman, A. Bandopadhyay, and B. D. Shaviv, “An eye tracking computer user interface,” in *Virtual Reality, 1993. Proceedings., IEEE 1993 Symposium on Research Frontiers in*, pp. 120–121, IEEE, 1993.
- [80] D. A. Robinson, “A method of measuring eye movement using a scleral search coil in a magnetic field,” *IEEE Transactions on bio-medical electronics*, vol. 10, no. 4, pp. 137–145, 1963.
- [81] T. Eggert, “Eye movement recordings: methods,” in *Neuro-Ophthalmology*, vol. 40, pp. 15–34, Karger Publishers, 2007.
- [82] T. Haslwanter and A. H. Clarke, “Eye movement measurement: electro-oculography and video-oculography,” *Handbook of Clinical Neurophysiology*, vol. 9, pp. 61–79, 2010.

REFERENCES

- [83] H. D. Crane, “The purkinje image eyetracker, image stabilization, and related forms of stimulus manipulation,” *Visual science and engineering: Models and applications*, pp. 15–89, 1994.
- [84] H. D. Crane and C. M. Steele, “Generation-v dual-purkinje-image eyetracker,” *Applied Optics*, vol. 24, no. 4, pp. 527–537, 1985.
- [85] E. B. Huey, *The psychology and pedagogy of reading*. The Macmillan Company, 1908.
- [86] P. Biswas and P. Langdon, “Multimodal intelligent eye-gaze tracking system,” *International Journal of Human-Computer Interaction*, vol. 31, no. 4, pp. 277–294, 2015.
- [87] R. A. Monty and J. W. Senders, “Eye movements and psychological processes,” tech. rep., DTIC Document, 1976.
- [88] M. A. Just and P. A. Carpenter, “Eye fixations and cognitive processes,” *Cognitive psychology*, vol. 8, no. 4, pp. 441–480, 1976.
- [89] K. Rayner, “Visual attention in reading: Eye movements reflect cognitive processes,” *Memory & Cognition*, vol. 5, no. 4, pp. 443–448, 1977.
- [90] K. Rayner, “Eye movements in reading and information processing: 20 years of research.,” *Psychological bulletin*, vol. 124, no. 3, p. 372, 1998.
- [91] R. Jacob and K. S. Karn, “Eye tracking in human-computer interaction and usability research: Ready to deliver the promises,” *Mind*, vol. 2, no. 3, p. 4, 2003.
- [92] T. Ohno, “Eyeprint: Using passive eye trace from reading to enhance document access and comprehension,” *International Journal of Human-Computer Interaction*, vol. 23, no. 1-2, pp. 71–94, 2007.
- [93] A. Murata, “Eye-gaze input versus mouse: Cursor control as a function of age,” *International Journal of Human-Computer Interaction*, vol. 21, no. 1, pp. 1–14, 2006.
- [94] T. O. Zander, M. Gaertner, C. Kothe, and R. Vilimek, “Combining eye gaze input with a brain-computer interface for touchless human-computer interaction,” *International Journal of Human-Computer Interaction*, vol. 27, no. 1, pp. 38–51, 2010.

REFERENCES

- [95] M. Adjouadi, A. Sestin, M. Ayala, and M. Cabrerizo, *Remote eye gaze tracking system as a computer interface for persons with severe motor disability*. Springer Berlin Heidelberg, 2004.
- [96] M.-C. Su, K.-C. Wang, and G.-D. Chen, “An eye tracking system and its application in aids for people with severe disabilities,” *Biomedical Engineering: Applications, Basis and Communications*, vol. 18, no. 06, pp. 319–327, 2006.
- [97] T. Koceljko, A. Bujnowski, and J. Wtorek, “Eye-mouse for disabled,” in *Human-Computer Systems Interaction*, pp. 109–122, Springer, 2009.
- [98] M. Donegan, J. D. Morris, F. Corno, I. Signorile, A. Chió, V. Pasian, A. Vignola, M. Buchholz, and E. Holmqvist, “Understanding users and their needs,” *Universal Access in the Information Society*, vol. 8, no. 4, pp. 259–275, 2009.
- [99] M. Caligari, M. Godi, S. Guglielmetti, F. Franchignoni, and A. Nardone, “Eye tracking communication devices in amyotrophic lateral sclerosis: Impact on disability and quality of life,” *Amyotrophic Lateral Sclerosis and Frontotemporal Degeneration*, vol. 14, no. 7-8, pp. 546–552, 2013.
- [100] R. Spataro, M. Ciriaco, C. Manno, and V. La Bella, “The eye-tracking computer device for communication in amyotrophic lateral sclerosis,” *Acta Neurologica Scandinavica*, vol. 130, no. 1, pp. 40–45, 2014.
- [101] J. San Agustin, H. Skovsgaard, J. P. Hansen, and D. W. Hansen, “Low-cost gaze interaction: ready to deliver the promises,” in *CHI’09 Extended Abstracts on Human Factors in Computing Systems*, pp. 4453–4458, ACM, 2009.
- [102] P. Majaranta and K.-J. Rih, “Twenty years of eye typing: Systems and design issues,” in *Proceedings of the 2002 Symposium on Eye Tracking Research & Applications*, ETRA ’02, (New York, NY, USA), pp. 15–22, ACM, 2002.
- [103] A. T. Duchowski, “A breadth-first survey of eye-tracking applications,” *Behavior Research Methods, Instruments, & Computers*, vol. 34, no. 4, pp. 455–470, 2002.
- [104] R. G. Lupu and F. Ungureanu, “A survey of eye tracking methods and applications,” *Bul Inst Polit Iasi*, pp. 71–86, 2013.
- [105] A. Al-Rahayfeh and M. Faezipour, “Eye tracking and head movement detection: A state-of-art survey,” *IEEE journal of translational engineering in health and medicine*, vol. 1, pp. 2100212–2100212, 2013.

REFERENCES

- [106] 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*, pp. 11–18, ACM, 1990.
- [107] R. A. Bolt, “A gaze-responsive self-disclosing display,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 3–10, ACM, 1990.
- [108] J. Scott, “Low tech methods of augmentative communication,” *Augmentative Communication in Practice 2*, 1998.
- [109] J.-D. Bauby, “The diving bell and the butterfly,” *London: Founh Estate*, 1997.
- [110] N. Bee and E. André, “Writing with your eye: A dwell time free writing system adapted to the nature of human eye gaze,” in *International Tutorial and Research Workshop on Perception and Interactive Technologies for Speech-Based Systems*, pp. 111–122, Springer, 2008.
- [111] P.-O. Kristensson and S. Zhai, “Shark 2: a large vocabulary shorthand writing system for pen-based computers,” in *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pp. 43–52, ACM, 2004.
- [112] D. W. Hansen, H. H. Skovsgaard, J. P. Hansen, and E. Møllenbach, “Noise tolerant selection by gaze-controlled pan and zoom in 3d,” in *Proceedings of the 2008 symposium on Eye tracking research & applications*, pp. 205–212, ACM, 2008.
- [113] H. Drewes and A. Schmidt, “Interacting with the computer using gaze gestures,” *Human-Computer Interaction–INTERACT 2007*, pp. 475–488, 2007.
- [114] D. W. Hansen, J. P. Hansen, M. Nielsen, A. S. Johansen, and M. B. Stegmann, “Eye typing using markov and active appearance models,” in *Applications of Computer Vision, 2002.(WACV 2002). Proceedings. Sixth IEEE Workshop on*, pp. 132–136, IEEE, 2002.
- [115] M. H. Urbina and A. Huckauf, “Dwell time free eye typing approaches,” in *Proceedings of the 3rd Conference on Communication by Gaze Interaction (COGAIN 2007)*, pp. 65–70, 2007.
- [116] H. H. Koester and S. P. Levine, “Modeling the speed of text entry with a word prediction interface,” *IEEE transactions on rehabilitation engineering*, vol. 2, no. 3, pp. 177–187, 1994.
- [117] D. Pedrosa, M. D. G. Pimentel, A. Wright, and K. N. Truong, “Filteryedping: Design challenges and user performance of dwell-free eye typing,” *ACM Transactions on Accessible Computing (TACCESS)*, vol. 6, no. 1, p. 3, 2015.

- [118] D. Pedrosa, M. d. G. Pimentel, and K. N. Truong, “Filteryedping: A dwell-free eye typing technique,” in *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, pp. 303–306, ACM, 2015.
- [119] J. Chen and Q. Ji, “Probabilistic gaze estimation without active personal calibration,” in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 609–616, IEEE, 2011.
- [120] J.-G. Wang, E. Sung, and R. Venkateswarlu, “Estimating the eye gaze from one eye,” *Computer Vision and Image Understanding*, vol. 98, no. 1, pp. 83–103, 2005.
- [121] A. Villanueva, R. Cabeza, and S. Porta, “Gaze tracking system model based on physical parameters,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 21, no. 05, pp. 855–877, 2007.
- [122] X. L. Brolly and J. B. Mulligan, “Implicit calibration of a remote gaze tracker,” in *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW’04. Conference on*, pp. 134–134, IEEE, 2004.
- [123] Y. Ebisawa and S.-I. Satoh, “Effectiveness of pupil area detection technique using two light sources and image difference method,” in *Engineering in Medicine and Biology Society, 1993. Proceedings of the 15th Annual International Conference of the IEEE*, pp. 1268–1269, IEEE, 1993.
- [124] O. Williams, A. Blake, and R. Cipolla, “Sparse and semi-supervised visual mapping with the s^3 gp,” in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06)*, vol. 1, pp. 230–237, IEEE, 2006.
- [125] J. J. Cerrolaza, A. Villanueva, and R. Cabeza, “Taxonomic study of polynomial regressions applied to the calibration of video-oculographic systems,” in *Proceedings of the 2008 symposium on Eye tracking research & applications*, pp. 259–266, ACM, 2008.
- [126] L. Sesma, A. Villanueva, and R. Cabeza, “Evaluation of pupil center-eye corner vector for gaze estimation using a web cam,” in *Proceedings of the Symposium on Eye Tracking Research and Applications, ETRA ’12*, (New York, NY, USA), pp. 217–220, ACM, 2012.
- [127] K.-H. Tan, D. J. Kriegman, and N. Ahuja, “Appearance-based eye gaze estimation,” in *Applications of Computer Vision, 2002. (WACV 2002). Proceedings. Sixth IEEE Workshop on*, pp. 191–195, IEEE, 2002.

REFERENCES

- [128] F. Lu, Y. Sugano, T. Okabe, and Y. Sato, “Inferring human gaze from appearance via adaptive linear regression,” in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 153–160, IEEE, 2011.
- [129] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, “Appearance-based gaze estimation in the wild,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4511–4520, 2015.
- [130] B. Wolfe and D. Eichmann, “A neural network approach to tracking eye position,” *International Journal of Human-Computer Interaction*, vol. 9, no. 1, pp. 59–79, 1997.
- [131] Y. Sugano, Y. Matsushita, and Y. Sato, “Calibration-free gaze sensing using saliency maps,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2667–2674, IEEE, 2010.
- [132] F. Alnajar, T. Gevers, R. Valenti, and S. Ghebreab, “Calibration-free gaze estimation using human gaze patterns,” in *Computer Vision (ICCV), 2013 IEEE International Conference on*, pp. 137–144, IEEE, 2013.
- [133] J. Chen and Q. Ji, “A probabilistic approach to online eye gaze tracking without explicit personal calibration,” *Image Processing, IEEE Transactions on*, vol. 24, no. 3, pp. 1076–1086, 2015.
- [134] Y. Zhang and A. J. Hornof, “Easy post-hoc spatial recalibration of eye tracking data,” in *Proceedings of the symposium on eye tracking research and applications*, pp. 95–98, ACM, 2014.
- [135] R. J. Jacob, “The use of eye movements in human-computer interaction techniques: what you look at is what you get,” *ACM Transactions on Information Systems (TOIS)*, vol. 9, no. 2, pp. 152–169, 1991.
- [136] C. Guan, M. Thulasidas, and J. Wu, “High performance p300 speller for brain-computer interface,” in *Biomedical Circuits and Systems, 2004 IEEE International Workshop on*, pp. S3–5, IEEE, 2004.
- [137] T. T. Whitepaper, “Eye tracking as a tool in package and shelf testing,” 2008.
- [138] T. T. Whitepaper, “Tobii eye tracking. an introduction to eye tracking and tobii eye trackers,” 2010.
- [139] P. A. Hall and G. R. Dowling, “Approximate string matching,” *ACM computing surveys (CSUR)*, vol. 12, no. 4, pp. 381–402, 1980.

REFERENCES

- [140] D. D. Salvucci, “Inferring intent in eye-based interfaces: tracing eye movements with process models,” in *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 254–261, ACM, 1999.
- [141] M. Kumar, *Gaze-enhanced user interface design*. PhD thesis, Stanford University, 2007.
- [142] M. Davies, “Word frequency data from the corpus of contemporary american english (coca),” 2011.
- [143] S. Eraslan, Y. Yesilada, and S. Harper, “Eye tracking scanpath analysis on web pages: how many users?,” in *Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications*, pp. 103–110, ACM, 2016.
- [144] M. A. Vadillo, C. N. Street, T. Beesley, and D. R. Shanks, “A simple algorithm for the offline recalibration of eye-tracking data through best-fitting linear transformation,” *Behavior research methods*, vol. 47, no. 4, pp. 1365–1376, 2015.
- [145] S. Hoppe, M. Löchtefeld, and F. Daiber, “Eypeusing eye-traces for eye-typing,” in *Workshop on Grand Challenges in Text Entry (CHI 2013)*, 2013.
- [146] S. Zhai and P.-O. Kristensson, “Shorthand writing on stylus keyboard,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 97–104, ACM, 2003.
- [147] 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*, pp. 61–70, ACM, 2003.
- [148] G. Navarro, “A guided tour to approximate string matching,” *ACM computing surveys (CSUR)*, vol. 33, no. 1, pp. 31–88, 2001.
- [149] K. Trnka, J. McCaw, D. Yarrington, K. F. McCoy, and C. Pennington, “Word prediction and communication rate in aac,” *Telehealth and Assistive Technologies (Telehealth/AT)*, pp. 19–24, 2008.
- [150] P. Majaranta, I. S. MacKenzie, A. Aula, and K.-J. Rähä, “Auditory and visual feedback during eye typing,” in *CHI’03 Extended Abstracts on Human Factors in Computing Systems*, pp. 766–767, ACM, 2003.
- [151] Y. Liu, B.-S. Lee, and M. J. McKeown, “Robust eye-based dwell-free typing,” *International Journal of Human-Computer Interaction*, vol. 32, no. 9, pp. 682–694, 2016.

- [152] P. Yu, J. Zhou, and Y. Wu, “Learning reconstruction-based remote gaze estimation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3447–3455, 2016.
- [153] Y. Sugano, Y. Matsushita, Y. Sato, and H. Koike, “An incremental learning method for unconstrained gaze estimation,” in *European Conference on Computer Vision*, pp. 656–667, Springer, 2008.
- [154] T. Baltru, P. Robinson, L.-P. Morency, *et al.*, “Openface: an open source facial behavior analysis toolkit,” in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1–10, IEEE, 2016.
- [155] K. Krafska, A. Khosla, P. Kellnhofer, H. Kannan, S. Bhandarkar, W. Matusik, and A. Torralba, “Eye tracking for everyone,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2176–2184, 2016.
- [156] A. Papoutsaki, N. Daskalova, P. Sangkloy, J. Huang, J. Laskey, and J. Hays, “Webgazer: Scalable webcam eye tracking using user interactions,” in *International Joint Conference on Artificial Intelligence*, 2016.
- [157] M. X. Huang, T. C. Kwok, G. Ngai, S. C. Chan, and H. V. Leong, “Building a personalized, auto-calibrating eye tracker from user interactions,” in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 5169–5179, 2016.
- [158] C. D. McMurrough, V. Metsis, J. Rich, and F. Makedon, “An eye tracking dataset for point of gaze detection,” in *Proceedings of the Symposium on Eye Tracking Research and Applications*, pp. 305–308, ACM, 2012.
- [159] U. Weidenbacher, G. Layher, P.-M. Strauss, and H. Neumann, “A comprehensive head pose and gaze database,” 2007.
- [160] K. A. F. Mora, F. Monay, and J.-M. Odobez, “Eyediap: A database for the development and evaluation of gaze estimation algorithms from rgb and rgb-d cameras,” in *Proceedings of the Symposium on Eye Tracking Research and Applications*, pp. 255–258, ACM, 2014.
- [161] Y. Sugano, Y. Matsushita, and Y. Sato, “Learning-by-synthesis for appearance-based 3d gaze estimation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1821–1828, 2014.
- [162] X. Zhang, Y. Sugano, M. Fritz, and A. Bulling, “It’s written all over your face: Full-face appearance-based gaze estimation,” *arXiv preprint arXiv:1611.08860*, 2016.

Publications

Journal

- (i) **Yi Liu**, Bu-Sung Lee, Deepu Rajan, Andrzej Sluzek, and Martin J. McKeown, “CamType: Assistive Text Entry Using Gaze with an Off-the-shelf Webcam” *Machine Vision and Applications (MVAP)*, Under review.
- (ii) **Yi Liu**, Bu-Sung Lee, and Martin J. McKeown, “Robust Eye-Based Dwell-Free Typing” *International Journal of Human-Computer Interaction (IJHCI)*, 2016.

Conference

- (i) **Yi Liu**, Bu-Sung Lee, and Martin McKeown, “A New Reconstruction Method in Gaze Estimation with Natural Head Movement” In *IAPR Conference on Machine Vision Applications*, Nagoya, Japan, May 2017.
- (ii) **Yi Liu**, Bu-Sung Lee, Andrzej Sluzek, Deepu Rajan, and Martin McKeown, “Feasibility Analysis of Eye Typing with a Standard Webcam” In *ECCV Workshop on Assistive Computer Vision and Robotics*, Amsterdam, The Netherlands, Oct 2016.
- (iii) **Yi Liu**, Bu-Sung Lee, Martin J. McKeown and Chonho Lee, “A Robust Recognition Approach in Eye-Based Dwell-Free Typing,” In *2015 International Conference on Progress in Informatics and Computing (PCI)*, Nanjing, China, Dec 2015.
- (iv) **Yi Liu**, Chi Zhang, Chonho Lee, Bu-Sung Lee and Alex Q. Chen, “GazeTry: Swipe Text Typing Using Gaze,” In *The 27th Australian Conference on Human-Computer Interaction (OzCHI)*, Melbourne, Australia, Dec 2015.

Co-Authored Publications

- (i) Chonho Lee, Jinyao Chin, **Yi Liu**, and Bu-Sung Lee, “Feasibility analysis and adaptive thresholding for mobile applications controlled by EEG signals,” In *23rd European Signal Processing Conference (EUSIPCO)*, Nice, France, Aug 2015.

REFERENCES

- (ii) Chonho Lee, **Yi Liu**, Li-Hau Tan, Weihan Goh, Bu-Sung Lee and Chai-Kiat Yeo, “A Wavelet Entropy-Based Change Point Detection on Network Traffic: A Case Study of Heartbleed Vulnerability,” In *IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom)*, Singapore, Dec 2014.
- (iii) Chonho Lee, Yu Terada, **Yi Liu**, and Bu-Sung Lee, “A motion accuracy evaluator based on body parts movement by MapReduce video processing,” *IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, Valencia, Spain, June 2014.