

# Applying Touch Gesture to Improve Application Accessing Speed on Mobile Devices



Chi Zhang

A thesis submitted in partial fulfilment of the requirements of  
Bournemouth University for the degree of Doctor of Philosophy

Jan. 2019

# Abstract

The touch gesture shortcut is one of the most significant contributions to Human-Computer Interaction (HCI). It is used in many fields: e.g., performing web browsing tasks (i.e., moving to the next page, adding bookmarks, etc.) on a smartphone, manipulating a virtual object on a tabletop device and communicating between two touch screen devices. Compared with the traditional Graphic User Interface (GUI), the touch gesture shortcut is more efficient, more natural, it is intuitive and easier to use. With the rapid development of smartphone technology, an increasing number of data items are showing up in users' mobile devices, such as contacts, installed apps and photos. As a result, it has become troublesome to find a target item on a mobile device with traditional GUI. For example, to find a target app, sliding and browsing through several screens is a necessity. This thesis addresses this challenge by proposing two alternative methods of using a touch gesture shortcut to find a target item (an app, as an example) in a mobile device.

Current touch gesture shortcut methods either employ a universal built-in system-defined shortcut template or a gesture-item set, which is defined by users before using the device. In either case, the users need to learn/define first and then recall and draw the gesture to reach the target item according to the template/predefined set. Evidence has shown that compared with GUI, the touch gesture shortcut has an advantage when performing several types of tasks e.g., text editing, picture drawing, audio control, etc. but it is unknown whether it is quicker or more effective than the traditional GUI for finding target apps. This thesis first conducts an exploratory study to understand user memorisation of their Personalized Gesture Shortcuts (PGS) for 15 frequently used mobile apps. An experiment will then be conducted to investigate (1) the users' recall accuracy on the PGS for finding both frequently and infrequently used target apps, (2) and the speed by

which users are able to access the target apps relative to GUI. The results show that the PGS produced a clear speed advantage (1.3s faster on average) over the traditional GUI, while there was an approximate 20% failure rate due to unsuccessful recall on the PGS.

To address the unsuccessful recall problem, this thesis explores ways of developing a new interactive approach based on the touch gesture shortcut but without requiring recall or having to be predefined before use. It has been named the Intelligent Launcher in this thesis, and it predicts and launches any intended target app from an unconstrained gesture drawn by the user. To explore how to achieve this, this thesis conducted a third experiment to investigate the relationship between the reasons underlying the user's gesture creation and the gesture shape (handwriting, non-handwriting or abstract) they used as their shortcut. According to the results, unlike the existing approaches, the thesis proposes that the launcher should predict the users' intended app from three types of gestures. First, the non-handwriting gestures via the visual similarity between it and the app's icon; second, the handwriting gestures via the app's library name plus functionality; and third, the abstract gestures via the app's usage history.

In light of these findings mentioned above, we designed and developed the Intelligent Launcher, which is based on the assumptions drawn from the empirical data. This thesis introduces the interaction, the architecture and the technical details of the launcher. How to use the data from the third experiment to improve the predictions based on a machine learning method, i.e., the Markov Model, is described in this thesis. An evaluation experiment, shows that the Intelligent Launcher has achieved user satisfaction with a prediction accuracy of 96%.

As of now, it is still difficult to know which type of gesture a user tends to use. Therefore, a fourth experiment, which focused on exploring the factors that influence the choice of touch gesture shortcut type for accessing a target app is also conducted in this thesis. The results of the experiment show that (1) those who preferred a name-based method used it more consistently and used more letter gestures compared with those who preferred the other three methods; (2) those who preferred the keyword app search method created more letter gestures than other types; (3) those who preferred an iOS system created more drawing gestures than other types; (4) letter gestures were more often used for the apps that were used frequently, whereas drawing gestures were more often used for

the apps that were used infrequently; (5) the participants tended to use the same creation method as the preferred method on different days of the experiment.

This thesis contributes to the body of Human-Computer Interaction knowledge. It proposes two alternative methods which are more efficient and flexible for finding a target item among a large number of items. The PGS method has been confirmed as being effective and has a clear speed advantage. The Intelligent Launcher has been developed and it demonstrates a novel way of predicting a target item via the gesture user's drawing. The findings concerning the relationship between the user's choice of gesture for the shortcut and some of the individual factors have informed the design of a more flexible touch gesture shortcut interface for "target item finding" tasks. When searching for different types of data items, the Intelligent Launcher is a prototype for finding target apps since the variety in visual appearance of an app and its functionality make it more difficult to predict than other targets, such as a standard phone setting, a contact or a website. However, we believe that the ideas that have been presented in this thesis can be further extended to other types of items, such as videos or photos in a Photo Library, places on a map or clothes in an online store. What is more, this study also leads the way in tackling the advantage of a machine learning method in touch gesture shortcut interactions.

# Dissertation Declaration

I agree that, should the University wish to retain it for reference purposes, a copy of my dissertation may be held by Bournemouth University normally for a period of 3 academic years. I understand that once the retention period has expired my dissertation will be destroyed.

## Confidentiality

I confirm that this dissertation does not contain information of a commercial or confidential nature or include personal information other than that which would normally be in the public domain unless the relevant permissions have been obtained. In particular any information which identifies a particular individual's religious or political beliefs, information relating to their health, ethnicity, criminal history or sex life has been anonymised unless permission has been granted for its publication from the person to whom it relates.

## Copyright

The copyright for this dissertation remains with me.

## Requests for Information

I agree that this dissertation may be made available as the result of a request for information under the Freedom of Information Act.

**Signed:**

Name:

# Acknowledgements

The doctoral research is coming to an end. Looking back over my doctoral study, my greatest achievement is not limited to the experience of scientific research and academic outcomes, more importantly, the good friendship with my supervisors and classmates. Their excellence and hard working make me deeply aware of my own shortcomings and inspires me never slack off. Their kindness and sincerity give me endless support and power, making me feel fearless and courageous. I would like to take this opportunity to express my great thanks to all those who have helped me.

Firstly, I would like to express my sincere gratitude to my advisor Prof. Feng Tian for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D study.

I would like to thank my second supervisor Prof. Chang Hong Liu. He's supports and advices have guided me through the most difficult moments of my research. Prof. Liu has given a lot of valuable suggestions on my research and helped me correcting my paper over and over again even for the details of changes. His serious attitude of research inspires my enthusiasm for my future research.

I would like to thank Dr. Nan Jiang. We have communicated, discussed and exchanged ideas in Human-Computer Interaction for two years. He have given me guidance for my research, so that I can get deeper understanding of the research topic.

Besides my supervisors, I would like to thank Dr. Jing Wang and Dr. Wenjuan Wang. We always have discussion, inspire each other and progress together. Their high motivated research attitude influences me greatly. I am lucky to have all the lovely colleagues

and friends who shared P521 in Pool House with me and I appreciate all the fun times we spent together. Thank all my friends in BU, we often have meals, do excises and chat together. Thanks for their accompanies.

My sincere thanks also goes to the anonymous volunteers for their invaluable data and feedback of the experiments in this thesis.

Finally, I would like to thank my family and friends for their love and support. This thesis is dedicated to my parents, Feng Zhang and Ying Liu.

# Contents

- 1 Introduction 19**
  - 1.1 Background . . . . . 19
  - 1.2 Motivation and Research Aims . . . . . 22
    - 1.2.1 PGS Method . . . . . 23
    - 1.2.2 Intelligent Launcher . . . . . 24
    - 1.2.3 Improve the prediction of Intelligent Launcher . . . . . 25
  - 1.3 Contributions . . . . . 27
  - 1.4 Thesis Outline . . . . . 30
  
- 2 Literature Review 32**
  - 2.1 Improvement on the App finding task . . . . . 32
    - 2.1.1 Organize apps by Categorise . . . . . 32
    - 2.1.2 Keyword Search Method . . . . . 33
    - 2.1.3 Voice Search Method . . . . . 33
    - 2.1.4 Touch Gesture Shortcut . . . . . 34
  - 2.2 Gesture definition and history . . . . . 35
  - 2.3 The advantages of Touch Gesture and its usage scenario . . . . . 39
    - 2.3.1 Naturalness and gesture based NUI . . . . . 40
    - 2.3.2 Intuitiveness and gesture control . . . . . 41
    - 2.3.3 Estate free and usage on smart-watch . . . . . 43
    - 2.3.4 Rich vocabulary and usage for inputting . . . . . 43
    - 2.3.5 Eyes-free and usage in moving environments . . . . . 45



---

2.3.6	Privacy and usage for authentication . . . . .	45
<b>3</b>	<b>Personalized Gesture Shortcuts for Frequent Used App Access</b>	<b>47</b>
3.1	Introduction . . . . .	47
3.2	Methods . . . . .	48
3.2.1	Gesture shortcut definition strategy . . . . .	48
3.2.2	The Agile Search App . . . . .	49
3.2.3	App sampling . . . . .	51
3.2.4	Measurement . . . . .	51
3.2.5	Experiment design . . . . .	52
3.3	Results . . . . .	54
3.3.1	Gesture shortcut recall . . . . .	54
3.3.2	Definition strategy recall . . . . .	54
3.3.3	Recall failures over apps . . . . .	54
3.4	Discussion . . . . .	56
3.5	Summary . . . . .	59
<b>4</b>	<b>Personalized Gesture Shortcuts for App Access on Smartphones</b>	<b>60</b>
4.1	Introduction . . . . .	60
4.2	Methods . . . . .	61
4.2.1	GUI . . . . .	62
4.2.2	Agile Gesture Interface . . . . .	62
4.2.3	Apparatus . . . . .	64
4.2.4	Participants . . . . .	64
4.2.5	App sample set . . . . .	65
4.2.6	Experimental design . . . . .	66
4.2.7	Procedure . . . . .	67
4.3	Results . . . . .	71
4.3.1	App access speed . . . . .	71
4.3.2	Relationship between access speed and the number of installed apps	72
4.3.3	Gesture recall accuracy . . . . .	73

4.3.4	Sample set size and recall accuracy . . . . .	74
4.3.5	Unsuccessfully recalled gestures . . . . .	74
4.3.6	Perceived effectiveness (Usability) . . . . .	75
4.4	Discussion . . . . .	76
4.4.1	App access speed . . . . .	76
4.4.2	Accuracy . . . . .	77
4.5	Summary . . . . .	78
<b>5</b>	<b>Intelligent Launcher: An Interface for App Search on Smartphones</b>	<b>79</b>
5.1	Introduction . . . . .	79
5.2	Exploring Gesture Shortcut for App Access . . . . .	80
5.2.1	Data Collection . . . . .	81
5.2.2	Types of Gesture Shortcut and the Reasons for their Creation . . . . .	82
5.2.3	Results and Analyses . . . . .	83
5.2.4	Creating an AI Gesture Shortcut Interface. . . . .	87
5.3	Intelligent Launcher . . . . .	88
5.3.1	Interaction with Intelligent Launcher . . . . .	88
5.3.2	Architecture of the Intelligent Launcher . . . . .	89
5.3.3	Algorithm of Intelligent Launcher . . . . .	90
5.3.4	Implementation . . . . .	94
5.4	Evaluations . . . . .	94
5.4.1	Participants . . . . .	94
5.4.2	Data Collection . . . . .	95
5.4.3	Results . . . . .	95
5.5	Discussion and Conclusions . . . . .	99
<b>6</b>	<b>Factors that Influence the Choice of a Touch Gesture for App Access</b>	<b>101</b>
6.1	Introduction . . . . .	101
6.2	Method . . . . .	103
6.2.1	Factors . . . . .	103
6.2.2	Participants . . . . .	106

---

6.2.3	Procedure . . . . .	106
6.2.4	Experiment Design . . . . .	108
6.3	Results . . . . .	108
6.3.1	Gesture creation reason consistency . . . . .	109
6.3.2	User's Profile and Gesture they choose . . . . .	111
6.4	Discussion . . . . .	119
6.4.1	Gesture creation reason . . . . .	119
6.4.2	User's Profile and Gesture Type . . . . .	119
6.4.3	Gesture recognition method . . . . .	122
6.5	Conclusion . . . . .	125
<b>7</b>	<b>Conclusion and Future Direction</b>	<b>128</b>
7.1	Conclusion . . . . .	128
7.2	Future direction . . . . .	130

# List of Tables

3.1	Examples of Definition Strategy. The gesture shortcut examples listed in this table were provided by users. . . . .	49
3.2	Apps selected for the experiment. . . . .	51
4.1	App's amount allocation in the experiment. . . . .	66
4.2	An example of an app sample set used in the experiment. . . . .	66
4.3	Pre-experiment Questionnaire . . . . .	68
4.4	7-point Likert scale for post-experiment rating . . . . .	70
4.5	Mean task completion time (s) as a function of <i>Access Method</i> , <i>App Usage Frequency</i> , <i>Retention Interval</i> . Values in parentheses represent the standard deviation. . . . .	72
4.6	Mean proportion recall accuracy as a function of <i>App Usage Frequency</i> and <i>Retention Interval</i> . Values in parentheses represent the standard deviation. . . . .	73
4.7	Summary of common recall errors and examples in each category . . . . .	75
5.1	Sample gestures from five creation reasons. . . . .	84
5.2	Sample of <i>function</i> gestures that were successfully mapped to the target app and those unsuccessfully (uns.) mapped. . . . .	86
6.1	Gesture samples of four gesture creation reasons in the experiment. . . . .	104
6.2	Gesture samples of three gesture shapes in the experiment. . . . .	106
6.3	Pre-experiment Questionnaire . . . . .	106

6.4	App allocations and participant information in the experiment. (Note "SD" in the table means the Standard Deviation.) . . . . .	107
6.5	Gesture creation reason question . . . . .	107
6.6	ANOVA of the <b>consistency</b> on Gesture Creation Reason for <i>frequent</i> and <i>infrequent</i> used apps with <i>0-day</i> , <i>1-day</i> and <i>7-days</i> Retention Interval of Preferred Gesture Creation Reason. . . . .	109
6.7	ANOVA of the <i>Letter</i> gesture <b>rate</b> on four Preferred Gesture Creation Reasons for <i>frequent</i> and <i>infrequent</i> used apps with three Retention Intervals. .	113
6.8	ANOVA of the <i>Drawing</i> gesture <b>rate</b> on four Preferred Gesture Creation Reasons for <i>frequent</i> and <i>infrequent</i> used apps with three Retention Intervals. . . . .	113
6.9	ANOVA of the <i>Abstract</i> gesture <b>rate</b> on four Preferred Gesture Creation Reasons for <i>frequent</i> and <i>infrequent</i> used apps with three Retention Intervals. . . . .	114
6.10	Mean reason <b>consistency</b> value as a function of Preferred Creation Reason, App Usage Frequency and Retention Interval. . . . .	123
6.11	App selected principal for "app classifier". . . . .	124

# List of Figures

1.1	An example of using GUI for app access. The actions circled by the red dotted line need to be repeated several times. . . . .	20
1.2	Overall prediction process of improved Intelligent Launcher. The process added as improvement is shown in the red rectangles. From left to right, the first red rectangle shows the process of using the factors which are extracted from the user profile to predict the type of gesture. The second red rectangle shows the process of improving the candidate's target prediction via learning from the relationship between the matching set and the app's Name, Function, and the Icon set. For more details, please refer to Chapter 6. . . . .	26
3.1	Workflow of Agile Search app. <b>Setting</b> and <b>Using</b> are two working modes and can be toggled by a button at the top right of the app. . . . .	50
3.2	The sequence of the experiment. . . . .	53
3.3	Overview of participants' gesture shortcut recall rate. . . . .	54
3.4	Overview of participants' gesture shortcut definition strategy recall rate. . .	55
3.5	Detailed breakdown of gesture shortcut recall failures over apps. . . . .	55
3.6	Detailed breakdowns of incorrect gesture shortcut pairs based on different definition strategies. . . . .	56
3.7	Detailed breakdowns of incorrect gesture shortcut pairs based on same definition strategy. . . . .	57

4.1	Agile Gesture for app access. Top row: procedure for app personalization. Bottom row: procedure for testing app access. . . . .	63
4.2	Agile Gesture records the task completion time in the background. The two example trials show access times by the PGS and GUI, respectively. . .	64
4.3	Nexus 5 used in the experiment for testing PGS. . . . .	64
4.4	Experiment procedure. The order of the two tasks (Task 1 and Task 2) in Day 2 and Day 7 was counterbalanced. . . . .	68
4.5	Mean task completion time for accessing <i>frequently</i> and <i>infrequently</i> used app via <i>Agile Gesture</i> and <i>GUI</i> . The error bars stand for one standard error about the mean. . . . .	72
4.6	Task completion time by <i>Agile Gesture</i> and <i>GUI</i> as a function of number of apps on participants' phones. . . . .	73
4.7	Mean proportion recall accuracy as a function of test sample set size. . . .	74
5.1	The Intelligent Launcher interface on Android. (a) The Intelligent Launcher is initially hidden with a floating icon (in the blue circle). (b) Pressing the icon opens the Launcher Screen. (c) When a user draws a gesture on the Launch Screen, (d) The Intelligent Launcher predicts the gesture's eight best matched target apps and presents them to the user. (e) The gesture is interpreted based on the app's name, function, icon and app usage history.	79
5.2	The data collection tool. . . . .	82
5.3	Gesture examples for <i>Letter</i> , <i>Drawing</i> and <i>Abstract</i> types. . . . .	83
5.4	The percentages of three types of gestures and their creation reasons. . . .	84
5.5	The percentage of <i>Single Letter</i> (initial), <i>Single Letter</i> (non-initial) and <i>Mul- tiple Letter</i> gestures created from an app's <i>name</i> and <i>function</i> . . . . .	85
5.6	The percentage of <i>Abstract</i> gestures for the top 3 and below the top 3 most frequently used apps. . . . .	87
5.7	The architecture of Intelligent Launcher. . . . .	90
5.8	The Intelligent Launcher optimised the prediction based on a Hidden Markov Model (HMM). . . . .	91

5.9	An example of how a "house" gesture matches the target app by learning its visual similarity to the icons in the template. The value on the black line is the visual similarity score: the value on the red line is the probability value of matching to that app. . . . .	93
5.10	The mean percentage of (1) correct match by the three gesture recognisers and the Intelligent Launcher (I.L.) when considering the top 8 matches, and (2) the frequency of using the <i>Letter, Drawing, Abstract</i> Gesture Recogniser to predict the target. The error bars stand for one standard error about the mean. . . . .	96
5.11	The mean percentage of the correct matches contributed by the Name and Function Library, Icon Template and the Log. The error bar stands for one standard error about the mean. . . . .	97
5.12	The average position of the correct target located in the candidate list. The error bar stands for one standard error about the mean. . . . .	98
5.13	The Launcher's prediction accuracy as a function of number of target candidates. . . . .	98
6.1	Mean reason <b>consistency</b> of four Preferred Creation Reasons on three different Retention Intervals for <i>frequent used app</i> (left) and <i>infrequent used app</i> (right). N.S. = not significant. *** $p < .001$ . . . . .	110
6.2	Mean <b>rate</b> of top: <i>letter</i> , middle: <i>drawing</i> and bottom: <i>abstract</i> gesture shape for four Gesture Creation Reasons on three different Retention Intervals to access <i>frequent</i> (left) and <i>infrequent</i> (right) used apps. The different bar in each group represent the <b>rate</b> of shape created on the Retention Interval of <i>0-day, 1-day</i> and <i>7-days</i> . The error bars stand for one standard error about the mean. N.S. = not significant. . . . .	112
6.3	Mean <b>rate</b> of top: <i>Letter</i> , middle: <i>Drawing</i> and bottom: <i>Abstract</i> gesture shape for three Preferred Search Method on three different Retention Intervals to access <i>frequent</i> (left) and <i>infrequent</i> (right) used apps. The different bar in each group represents the <b>rate</b> of shape created on the Retention Interval of <i>0-day, 1-day</i> and <i>7-days</i> . The error bars stand for one standard error about the mean. N.S. = not significant. ** $p < .05$ . . . .	115



- 6.4 Mean **rate** of three types for *frequent* (left) and *infrequent* (right) used apps created by the *Android* and the *iOS* user. The different bars in each group represent the **rate** of gesture created by the *Android* or the *iOS* user. The error bars stand for one standard error about the mean. N.S. = not significant. \*\*\*  $p < .001$ . . . . . 118
- 6.5 Mean **rate** of three gesture types of the *Android* user (left) and the *iOS* user (right) for *frequent* and *infrequent used app*. The different bars in each group represent the **rate** of gesture created for *frequent* or *infrequent* used apps. The error bars stand for one standard error about the mean. N.S. = not significant. \*\*\*  $p < .001$ . . . . . 118
- 6.6 An example of app icons display on an *Android* phone (left) and on an *iOS* phone (right). . . . . 121
- 6.7 The architecture we proposed for target app prediction. An example with input gesture "M" is given. The value  $W_x$  on the line which links the predictor source manager and each source represents the weight used for prediction. . . . . 123
- 6.8 Bayesian network model for intelligent launcher. . . . . 125

# List of Publications

- Zhang, C., 2017, May. Improving App Lookup speed on Mobile via User-defined Touch Gesture. In Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems (pp. 196-201). ACM.
- Zhang, C., Jiang, N. and Tian, F., 2016, November. Accessing Mobile Apps with User Defined Gesture Shortcuts: An Exploratory Study. In Proceedings of the 2016 ACM on Interactive Surfaces and Spaces (pp. 385-390). ACM.
- Zhang, C., 2016, July. Agile search: a tool for searching apps easily on mobile by the gesture. In Proceedings of the 30th International BCS Human Computer Interaction Conference: Fusion! (p. 12). BCS Learning & Development Ltd..

# Chapter 1

## Introduction

### 1.1 Background

With the rapid development of apps on smartphones, the pattern of app usage has changed considerably (Statista 2016a; Carrascal and Church 2015). It has been estimated that current smartphone users spend nearly three hours on their apps per day (Lipsman 2017). Their app access has become more frequent, which is now estimated to be 150 times, on average, per day (Slideshare.net 2017). Additionally, with such a large number of apps available in the app market (Statista 2015a), having hundreds of apps installed on a smartphone for a variety of tasks is becoming increasingly common (Yahoo 2014). A consequence of these changes is that the conventional methods of app access on smartphones can no longer effectively handle the new situation (Li 2012).

Take an Android phone as an example: a typical Android phone can hold up to 16 normal sized app icons in one single screen (see Figure 1.1). That means the smartphone users who have more than 16 installed apps need to navigate through several screens to find any given target app. Even in average conditions, six to seven screens are needed to hold these apps. Finding a target app by navigating and browsing these GUI screens is very time-consuming and inconvenient (Felt et al. 2011).

A lot of research has been done dedicated to tackling this problem. These search

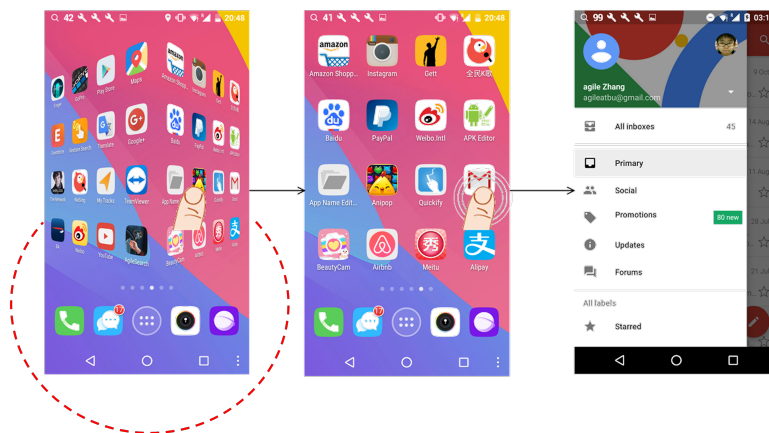


Figure 1.1: An example of using GUI for app access. The actions circled by the red dotted line need to be repeated several times.

methods either rely on grouping apps with similar functions into the same category or on the same screen (Böhmer and Krüger 2013; Lulu and Kuflik 2015; Singh et al. 2015), or on searching for the target according to the app's name, which users input via the keyboard or through voice access (Apple-siri 2016; Lewe 2002; Google 2015a). However, these approaches are still not commercially available due to the following reasons. First, app searching by category can fail if a user understands a category differently from that defined by the system (Zhang et al. 2012). Second, searching for an app's name using a keyboard can be inconvenient in scenarios such as walking (Wang et al. 2015) or when using only one hand (Bragdon et al. 2011). Third, searching by voice has low public acceptance. Moreover, both keyboard input and voice search require the user to remember an app's name, problems arise when the user forgets the name.

Using touch gestures can overcome these limitations and allow for more efficient, natural and direct interactions with smartphones (Morris et al. 2010; Albinsson and Zhai 2003). A touch gesture shortcut does not rely on any grouping/display method, nor does it require a keyboard or voice input. Furthermore, touch gesture shortcuts are easy to input compared with keywords. Using them does not rely on the keyboard or any small buttons, thus avoiding the fat finger problem (Baudisch and Chu 2009). This method can be used for a wide range of everyday tasks, including text editing, picture drawing (Gould and Salaun 1987), audio control (Strachan et al. 2010) and content browsing (Billinghurst and Vu 2015). Research has revealed that among these commonly performed tasks, most of the time users tend to use gestures for accessing the installed apps (Poppinga

et al. 2014).

However, the current touch gesture shortcut methods may not work effectively when a large number of apps have been installed because they all employ a universal built-in system-defined shortcut template (Li 2010a 2012; Ouyang and Li 2012; Lu and Li 2015; Roy et al. 2013). They require users to learn a template first and then draw the gesture shortcut to access the target app associated with it. For example, the gesture "G" is a shortcut template for accessing the Gmail app. Users can only open Gmail app by drawing "G". Although it is an effective method, a significant limitation is that it is not easy to pre-define a set of gestures for all apps on a user's smartphone, because the apps installed by users vary from one person to another. Moreover, the number of gestures that can meet all users' requirements is large. Even if a comprehensive template could be created, users may not be satisfied with the predefined gestures. They may instead prefer gestures to be defined differently.

On the other hand, several Artificial Intelligence (AI) approaches for quick app access avoid this problem by predicting and prompting a short list of apps that a user is likely to use based on specific contexts (Baeza-Yates et al. 2015; Zhang et al. 2012; Böhmer et al. 2011; Xu et al. 2013; Huang et al. 2012; Shin et al. 2012) or the user's personal preferences (Kim et al. 2017; Gillespie et al. 2014; Raptis et al. 2017). However, these approaches do not benefit from the advantages of a touch gesture shortcut. Approaches such as Gesture Search (Li 2012), Gesture Marks (Ouyang and Li 2012) and Gesture On (Lu and Li 2015) benefit from combining two methods. They use the wise of AI to infer the user's intent target items from the touch gesture shortcut they draw. A common feature of these approaches is that they rely on either a purely handwriting recogniser (Bahlmann et al. 2002; Connell and Jain 2001; Hu et al. 2000) or a combination of this with a pre-collected gesture-app library from the population of users. As a result, they can infer the meaning of a gesture that has not been defined. Nonetheless, the accuracy of prediction is limited. The average prediction accuracy of Gesture Marks was under 61% (when the top match of all three repetitions of the test are considered).

Another method uses the Personalized Gesture Shortcut (**PGS**) to access the target app. However, research which focused on system functions (Wobbrock et al. 2009) discovered that users tended to create the shortcuts relying on different information. For ex-

ample, when users were asked to create gestures for shortcut access to a smart phone's key functions (e.g., answering/ rejecting the coming call, turning the phone to silent mode, etc.), the users did not always create gestures based on the textual information of such functions as a quarter of the gestures created by participants resembled certain visual elements in the app icons. This seemed to echo the picture superiority effect (Paivio and Csapo 1973; Childers and Houston 1984), whereby a concept is easier to recall if it is presented as a picture rather than as words.

Overall, the current touch gesture approaches are still not commercially available because they (1) require users to learn a system-defined template before using them; (2) they are all limited by the prediction accuracy; and (3) they do not support recognition of PGS which has been created based on information other than textual information.

## 1.2 Motivation and Research Aims

By observing the limitations noted in the previous section, including:

- Having to shift backwards and forwards several times between different screens when searching for a target app if using the current widely used method.
- Grouping the apps with similar functions into the same category in order to save the app finding time was limited by the accuracy of the grouping algorithm and the user's understanding of how it worked.
- Being limited by the scalability (the pre-system defined template).
- Requiring learning to take place before using (via system-defined gestures).
- Users tending to create gestures by relying not only on textual information but also other information.

These limitations motivate this thesis. Consequently, this thesis aims to propose methods to improve the interaction method for digital target accessing experiences on mobile devices (e.g., smartphones and tablets).

### 1.2.1 PGS Method

Although current studies have shown the great potential of touch gesture shortcuts based on a template, the effort required to learn the template has hindered any wide spread adoption.

To overcome this limitation, this thesis proposes an alternative approach that uses a Personalized Gesture Shortcut (PGS) for fast app access. This method allows users to personalize their own gesture shortcuts. We tested this approach using an Android app named Agile Gesture (to be introduced in detail in 3.2) in two steps. First, we explored whether the approach is effective for the selected 15 frequently used apps. Second, we tested it under the settings which mimic daily use of smartphones.

This approach requires a learning process similar to that of the template-based methods in that it also relied on the user's memory to recall the shortcuts they had defined, but it was much less demanding (Oh and Findlater 2013). Previous work has examined recall memory 24 hours after learning a gesture template predefined by a system (Nacenta et al. 2013). Although the study showed the average recall rates on the next day were 55% for gestures predefined by the system designer, it was unknown whether the memory of PGS would also follow the same pattern. Moreover, it was unclear how a longer retention time beyond 24 hours might affect the memory of PGS. This is important because an app may not be used until a few days after its gesture shortcut is personalized (Wortham 2013).

Another important justification for the PGS should be the speed, relative to the GUI, at which users are able to access the target apps. Previous touch gesture methods, including those aimed at improving the speed of accessing data on smartphones, did not measure the time taken to complete app access. To assess the potential speed advantage of PGS, we compared the speed of PGS with GUI for app access tasks. The speed of GUI for app access is likely to depend on the position of an app icon on a smartphone (Zhang et al. 2012). Apps on the home screen should take less time to access compared to those on other screens. Because the space on the home screen is limited, users usually keep the most frequently-used apps there for easy access. However, for less frequently-used apps, it could take users much longer to find.

In summary, the research aims of evaluating the effectiveness of the PGS method are

as follows:

- To test user recall accuracy using the PGS for 15 selected frequently-used apps 24 hours after their personalization.
- To test user recall accuracy using the PGS to access both frequently and infrequently used apps 24 hours and 7 days after their personalization.
- To compare the speed of the PGS with GUI when accessing both frequently and infrequently used apps 24 hours and 7 days after their personalization.

### 1.2.2 Intelligent Launcher

Although the speed advantage and effectiveness of the PGS method has been confirmed through the first (frequently-used apps) and second experiment (both frequently and infrequently used apps) in this thesis, PGS still can not be used without predefined gestures. This thesis intends to overcome this, with the help of AI techniques, through the proposed Intelligent Launcher, which is an interface that can predict a user's intended app from the touch gesture shortcut they have drawn.

Several pieces of previous research have motivated this thesis. Gesture Search (Li 2010a) relies on an alphabetical recogniser, which allows users to select their apps from a shortened app list. It recognises the sequence of Letter gestures drawn by the user and displays a list of apps ordered by the degree of frequency of use of the letter(s) matched to the app's name. Gesture On (Lu and Li 2015) further saves accessing time by allowing the user to draw a Letter gesture from the lock-on screen mode. However, studies have shown that users do not always create Letter gestures. They also use Drawing gestures when they are given the freedom to define their Gesture Shortcut (Zhai et al. 2012; Poppinga et al. 2014; Malloch et al. 2017). Gesture Marks (Ouyang and Li 2012) addressed this issue and improved the approach's scalability by adding a non-handwriting (Drawing gesture) template constructed from a population of end-users.

However, these approaches have not considered gestures being created based on an app's *function*. Users do not always remember the target app's name. When they fail to recall the name of an app, the name-based approaches cannot be used. Additionally,



the Drawing gesture's prediction relies on the template created by another population. Because users tend to use different apps, such templates are not effective for all users (Böhmer et al. 2011; Shin et al. 2012).

To overcome the existing limitations and to build a more efficient method, we first conducted a study to find out the reasons underlining the creation of gestures. Based on the findings, the Intelligent Launcher was developed. In order to evaluate the Intelligent Launcher, we conducted a second evaluation study.

In summary, the research aims are as follows:

- To establish the pattern of gesture creation by app users to inform the design of an intelligent target-item-finding interface based on a touch gesture shortcut.
- To provide design details regarding how to construct a gesture-app mapping set.
- To evaluate the effectiveness of the Intelligent Launcher.

### **1.2.3 Improve the prediction of Intelligent Launcher**

Figure 1.2 shows the prediction process of the improved Intelligent Launcher. Although this thesis has given guidelines on and demonstrated how to predict three types of gesture in Chapter 5, users do not use these types of gestures equally. For more accurate predictions, the Launcher should know which type of gesture the user draws before predicting since the prediction of the different types of gesture rely on different predictors. For example, the prediction of the Letter and the Drawing gestures rely on two gesture-app sets, but the power of each set's contribution to the prediction is different.

Previous studies have identified different types of gestures created by users for different reasons (Nacenta et al. 2013; Poppinga et al. 2014; Wobbrock et al. 2009; Zhang et al. 2016; Lü and Li 2011). Therefore, this thesis has reviewed related work to uncover the factors which may influence the user's choice on type of gesture and conducted an experiment to explore the relationship between these selected factors and the gesture types. Specifically, this thesis has analysed the influence of user preference (i.e. whether an Android or iOS user, the preference for an app finding method and gesture creation reason, frequency of target app usage) on the gesture type (i.e., Letter, Drawing or Abstract) they create. Our predictor indicates all input gestures as Letter, Drawing

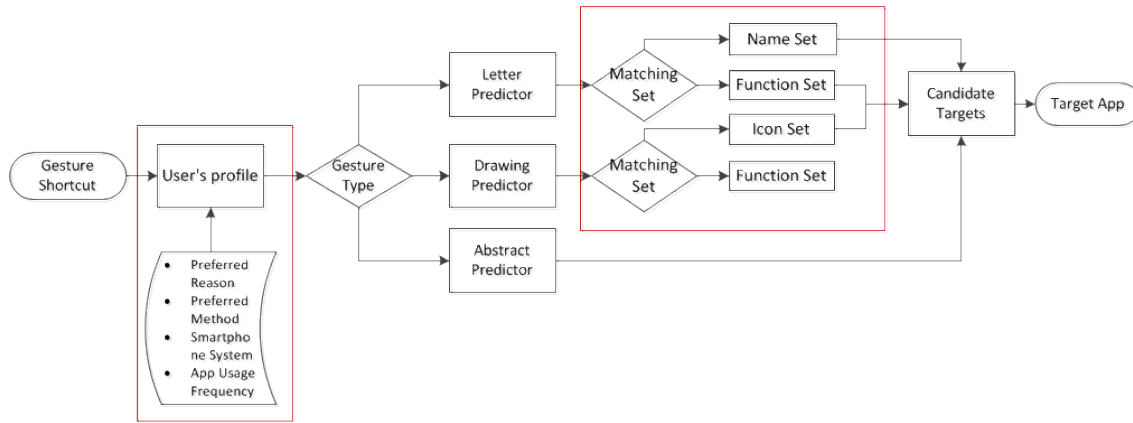


Figure 1.2: Overall prediction process of improved Intelligent Launcher. The process added as improvement is shown in the red rectangles. From left to right, the first red rectangle shows the process of using the factors which are extracted from the user profile to predict the type of gesture. The second red rectangle shows the process of improving the candidate's target prediction via learning from the relationship between the matching set and the app's Name, Function, and the Icon set. For more details, please refer to Chapter 6.

or Abstract gestures based on the result of T scores which have been calculated from equation (6.1). The type with the highest T scores among the three types is indicated as the type of gesture. For gestures that indicate as a Letter gesture, we adopted the alphabetical template as the source of recognizer; for those using Drawing gestures, we adopted the template consisting of all the installed app's icons as the source; for those using Abstract gestures, we adopt eight items from the Abstract gestures template which had also been used in previous research (Appert and Zhai 2009; Dittmar et al. 2015; Gillespie et al. 2014).

Previous work (Escalera et al. 2017; Rautaray and Agrawal 2015) predicted the target based on a universal template. Our work has proposed a method to predict the target based on different templates (i.e., app's Name, Function and Icon sets). These sets do not contribute equally to the different types of gesture. For example, a Letter gesture may better explain and predict via the app's name or function, but not the app's icon. Thus, the Name and Function sets have more power when predicting a Letter gesture's target than the Icon set. Therefore, for a better prediction, we need to know how much the power of each set contributes to the particular type of gesture prediction.

Overall, the last challenge was to find a proper AI method predicting based on all these related known factors (i.e., the user's profile). In this way, the Intelligent Launcher

can re-fine the prediction results based on the users' profile. Bayesian networks are ideal for predicting the likelihood that any one of several possible known causes may be the contributing factor (Blodgett and Anderson 2000; Margaritis 2003). Bayesian networks are a type of probabilistic graphic model that uses Bayesian inference for probability computations. The higher the probability value the candidate target has, the more likely it will be the target the user wants. The probabilistic graphic can be constructed based on the known relationships between the input gesture and the factors extracted from the user's profile.

In summary, to improve the Intelligent Launcher's predictions, the research aims are as follows:

- Review existing studies which focus on fast app access and identify the aspects which may influence the choice of gesture.
- Explore the relationship between these factors and the gesture type (i.e., letter, drawing or abstract) chosen.
- Evaluate how much the name, function and icon sets contribute to the letter or drawing gesture's ability to predict based on the user's profile.
- Specify a Bayesian network model for the improvement of the predictive power of the Intelligent Launcher based on the user's profile.

### 1.3 Contributions

This thesis has proposed two alternative methods of using a touch gesture shortcut for target app access: PGS and the Intelligent Launcher. A method for improving the prediction accuracy of the Intelligent Launcher has also been proposed.

By seeing the potential of using a touch gesture shortcut to interact with a mobile device, this thesis first focused on using the **PGS method** to improve the app accessing process. Two experiments were conducted to verify the effectiveness of the PGS approach. The first experiment showed that the participants could recall 80% of their personalized gesture shortcuts for accessing 15 frequently-used apps (Chapter 3). The

second experiment showed that PGS held a clear speed advantage over GUI, especially when accessing infrequently-used apps (Chapter 4). The advantage was observed after both a relatively short (24 hours) and a longer (7 days) retention period following the participants' initial personalization. The mean proportion recall accuracy of PGS was 0.84 after 24 hours of initial personalization, which was higher than 0.55, the mean proportion recall accuracy of the gesture shortcut template reported by Nacenta et al. (2013). Based on these findings, the advantages and limitations of our method are discussed. One major challenge is that users may not recall the gesture correctly as they have personalized it.

To face the challenge of the PGS method, the **Intelligent Launcher** provided another way of finding the target app via the touch gesture shortcut (Chapter 5 and 6). It does not require the user to memorise the shortcut anymore. The Intelligent Launcher could predict the potential target app via the gesture the user has given without it being predefined. Unlike Gesture Search (Li 2010a) or Gesture On (Lu and Li 2015), the Intelligent Launcher can not only predict based on a Letter shape gesture but also a Drawing or Abstract shaped Gesture.

Using the Intelligent Launcher also revealed that Letter gestures were not always based on the app's *name*. They could also arise from an app's *function*: e.g., a gesture "N" for "Google Map" being based on its "Navigation" function. This thesis also found that users tend to use a single letter gesture (e.g., "M") arising from the app's *name* (e.g., for "Messenger") more often than a multiple letter gesture (e.g., "Mes"), while its exact opposite occurs for gestures created from an app's *function*. For example, gestures like "Nav" which consisted of more than one letter, was used more often than "N" (i.e., an abbreviation of "navigation"). Most Drawing gestures were created similar to the target app's *icon*, e.g., a "triangle" gesture for "YouTube". It should be noted that a small number of Drawing gestures have no semantic meaning, e.g., a horizontal line, yet these can be created for the most frequently-used apps. We name this kind of gesture the "Abstract-Type" in this thesis, and they infer their target app via the app *usage history*.

Overall, the Intelligent Launcher's novelty is three-fold, as follows:

- It employs an alternative method to infer the target app from a drawing gesture by assessing its visual similarities with the icons of the installed apps rather than mak-

ing inferences based on the predefined gesture templates from another population. This improves the accuracy of prediction because the source for predictions comes from the users' own apps rather than the apps shared with other users.

- It expands the existing Gesture Search and Gesture On programs by allowing a letter gesture based on either an app's *name* or its *function*.
- To the best of our knowledge, this is the first interface that supports the prediction of Abstract-Type gestures by analyzing the app's usage frequency recorded on the app *usage history*.

By investigating the prediction accuracy of the Intelligent Launcher, this thesis found that prediction can be improved if:

- The gesture type can be predicted first since different types of gestures can be explained and predicted using different sources. For example, the majority of correct predictions of a letter gesture was predicted from the app's name. However, the majority of correct predictions of a drawing gesture was predicted from the app's icon.
- Knowing the power (potential) of each template (set) contributes to the three predictors (see Figure 1.2) when predicting. The rational is the Name, Function and Icon sets equally contribute to the three predictors. For example, the Name set contributes more (60%) than the function set (16%) to the Letter Predictor.

Based on this, Chapter 6 focuses on **improving the prediction of the Intelligent Launcher**. A Bayesian network model which specifies the relationship of the type of gesture, the user's profile and the target app (conditional dependency) is proposed based on the findings of the experiment. The network obtains the probability value of the target app by taking advantage of the user's profile. For example, if a user draws a gesture "N", our approach predicts this to be a Letter gesture since their profile shows they prefer the *Keyword Search Method* when finding the target. Then, the Intelligent Launcher predicts his/her Letter gesture-target app set as having two matching candidates: "Notepad" and "Google Map". Again, the profile tells us that he/she preferred to create a gesture based on the app's name. Thus, we can finally predict that "Notepad" is the most probable target app for the user.

This is different from previous studies that predict app usage based on contextual information, app usage patterns or functionality. Their prediction accuracy is limited and they do not benefit from the touch gesture shortcut. This thesis has revealed the potential of the user's profile included on their smartphone's system, their preferences regarding gesture creation, their app finding method and the app's usage frequency in predicting the user gesture's type. Additionally, this thesis has demonstrated the method of improving the target prediction of the Letter and Drawing gestures according to the user's profile.

## 1.4 Thesis Outline

- **Chapter 1** introduces the research background, motivation, research aims and the main contributions of this thesis.
- **Chapter 2** sums up the related work of this thesis.
- By seeing the advantages of the touch gesture shortcut in target accessing reflected in the literature, **Chapter 3** focuses on personalized gesture shortcuts. **Chapter 3** conducts an experiment to provide evidence that the PGS is effective in accessing frequently-used apps.
- **Chapter 4** extends the study of Chapter 3. It further focuses on the PGS method for app access under the settings which mimic our daily lives. A novel touch gesture shortcut method allows the users to use whatever gesture they like as a shortcut to access both frequently and infrequently used target apps. However, the PGS will be seen to be limited by the user's memory concerning their self-defined gesture shortcuts.
- Noting the challenges of the PGS method, **Chapter 5** focuses on the Intelligent Launcher. It is a novel method combined which combines the methods proposed in Chapter 3 and 4 (PGS) and AI techniques which can predict a user's intended target app from the gesture they draw. The gesture does not need to be predefined or memorized in advance.
- **Chapter 6** focuses on improving the predicting accuracy of the method proposed in

Chapter 5 (the Intelligent Launcher). The improvement in the method is motivated by the results provided in Chapter 5.

- **Chapter 7** summarises the work of this thesis and discusses potential future research.

## Chapter 2

# Literature Review

This section will provide a survey of related work that includes: (1) research which has attempted to improve the app finding task by organizing apps by categories; searching by app name; and using touch gesture shortcut methods; (2) a review of the gesture's definition and its history; (3) an assessment of the advantages and usages of the touch gesture shortcut in HCI.

### 2.1 Improvement on the App finding task

A lot of research has been aimed at simplifying the process of app access on smart-phones in recent years. The Traditional Graphic Interface (GUI) can be difficult to use and be time-consuming due to the increased number of apps currently being installed by users (Li 2010a). This thesis reviews the research which has focused on improving the app finding task by the methods which follow.

#### 2.1.1 Organize apps by Categorise

Inevitably, apps need to be displayed on multiple pages. To allow users to access the installed apps more easily, Lavid Ben Lulu and Kuflik (2013) proposed an unsupervised machine learning method to classify installed apps into functionality-based categories.



The program then displays the apps with similar functions on the same page of the screen. They further developed a set of menus which help users to operate the app finding process with ease by cataloguing hints (such as app function and icon colour) in the corresponding menu (Lulu and Kuflik 2015). Similarly, ZapDroid (Singh et al. 2015) has classified installed apps into different categories both according to the apps' functionality and the frequency of use. Such approaches help users to find apps more easily. However, they all rely on the user's understanding of a category.

### **2.1.2 Keyword Search Method**

Keyword search is another way to assist app access. It does not rely on pre-classification. Recent smartphones all have a built-in search function, such as Spotlight Search (Apple 2017) on the iPhone, and Google Now (Google 2015a) on Android phones. However, these approaches need users to input the keywords required to implement a search by either typing them or speaking them aloud. These methods are either limited by the input space or by social tolerance in public. The input space issue is also known as the "fat finger" problem, which refers to the situation when the area available on display for a key input is too small (Baudisch and Chu 2009). Voice input can be a nuisance in public or in the presence of others. These issues have been addressed by Fitzmaurice et al. (1993), whose experiment showed that gesture-based approaches provide a more natural way to overcome these two issues. Gesture-based methods allow a user to use a touch gesture for a particular task, such as, the "-" gesture for volume down and "+" gesture for volume up.

### **2.1.3 Voice Search Method**

The development of voice recognition technologies has opened the door for voice search on mobile devices. Similar to the keyword search, voice search is another way in which users can access a target item on their mobile devices (Apple-siri 2016; Lewe 2002; Google 2015a). Voice Search had already been used before it was introduced to mobile devices. It had been used for searching on mobile web browsers (Guy 2016; Acero et al. 2008) and on mobile device's operating the Google Search Engine (Schalkwyk

et al. 2010). Voice search applications utilize speech as the primary input modality. Unfortunately, mobile settings often involve extraneous noise, which results in low voice recognition accuracy (Paek et al. 2008). Despite this, the voice search method can work in a relative quiet environment. However, the tolerance for voice input in the public environment is low. Therefore, an input modality other than voice input, such as the touch gesture shortcut is needed.

#### **2.1.4 Touch Gesture Shortcut**

Even though the methods mentioned in the previous three sections (i.e., Section 2.1.1; 2.1.2 and 2.1.3) seemed to have already solved the research problem, but they suffered from limitations. These included reliance on the user's personal understanding of the category, the "fat finger" problem and the low public tolerance.

The touch gesture shortcut is a new dimension in mobile interaction which can help improve the app finding task. On the one hand, the touch gesture shortcut does not suffer from the limitations mentioned above. On the other hand, it has advantages over the traditional interaction methods (i.e., selecting from a hierarchical menu, such as the category menu or using keyword or voice input for to carry out a search). The touch gesture shortcut has better communicative attributes than the traditional means of interaction, such as keyboard input or voice input. Plus, the touch gesture shortcut is a more intuitive way for humans to interact compared with the traditional ways (e.g., using select buttons, controlling a slider, etc.).

Various forms of gesture have also been proposed for accessing different target items, their advantages having been identified. For example, the location information on a mobile can be accessed by sweep and shake gestures (Robinson et al. 2009), while applications, contact records etc. can be accessed by alphabetical gestures (Li 2010a). Specific objects in a scene can be accessed by pointing gestures in a human-robot interface (Schauerte and Fink 2010), while images or sequences of videos can be accessed by horizontal or vertical swipe gestures on a touch-based device (Schoeffmann 2014). Various tasks can also be performed via touch gestures: for example, editing a text or drawing pictures (Gould and Salaun 1987), controlling the music player (Strachan et al. 2010) or web browsing (Billinghurst and Vu 2015).

Several approaches that use gestures for app data and information access have achieved good results (Li 2010a; Lü and Li 2011; Lu and Li 2015). However, these approaches still rely on the textual information of apps as cues for gestures (i.e., alphabet-shaped gestures often refer to the first letter of an app's name). This means they will become less effective when users forget the names of these apps. This may happen more often with less frequently-used apps. Moreover, such gestures are still predefined by the system/developer, which makes them harder for users to recall than user-defined gestures (Morris et al. 2010). A study (Wobbrock et al. 2009) on universal gesture patterns found that, when users were asked to create gesture shortcuts to access smartphone data, they did not always create gestures based on the textual information alone (Heuwing et al. 2015). In fact, a quarter of gestures created by participants for accessing apps result in drawings shaped to resemble certain visual features of the app icons. This suggests gestures other than alphabet-based shapes can also play a key role in a user's creation of gestures. It should be noted that the time spent on accessing the target item was not measured in this experiment.

Our proposed method does rely on AI but in a different way from previous methods. We predict the target app not only based on the function of but also on other information about the apps. The PGS method does require a gesture as input; however, this could be not only a letter-like gesture but also drawing-shaped gestures.

## 2.2 Gesture definition and history

Gestures play an important role in the field of human-computer interaction (Westerman et al. 2001). Differing from the original definition of gesture as “a physical expression of human mental concepts” (McNeill 2000), in the HCI field, a gesture is defined as a particular hand moving pattern performance by a user who wants to interact with a computer interface (Pavlovic et al. 1997).

The technology of gesture interaction in the HCI field has developed along with the gesture-controlled user interface and gesture recognition techniques. These two techniques have been inseparable in their support of and influence over each other. In the following section, the history of the development of the gesture in the HCI field will be

demonstrated following the development of the gesture-controlled user interface and gesture recognition techniques.

The first study of how humans use gestures to represent their intentions dates back to 1975 (Bulwer 1975). Since then, researchers of anthropology, neurophysiology and psychology have developed an interest in observing the relationship between human gesture behaviour and their language in use (McCafferty and Stam 2009; McNeill 2000). More recently, researchers in the HCI field have become interested in how to benefit from gesture interaction techniques to control computer systems.

The first finger-driven touchscreen device was invented by E.A. Johnson in 1965 in the United Kingdom (Johnson 1965). This device was sensitive to the point touch of a finger and provided the first effective way of using point touch-based gestures, such as tap and double-tap, to interact with a computer. This approach was improved by Johnson himself, two years later, but it still could process only one touch at a time (Johnson 1967). Then, in 1982, the first multi-touch system was designed by Mehta (1982) at the University of Toronto; it supported multi-touch gestures such as double-tap at the same time. It could detect the point of a finger or several fingers on the panel for interaction, and register the finger gesture as an input to the system. This was when the era of gesture-based interaction opened.

With the development of technology, not only the physical touch on the screen, but also a gesture in the middle of the air can somehow control a system. In addition to the multi-touch gesture, the in-air gesture is another kind of gesture that users employ performing a gesture without touching the device. The in-air gesture was first used in Krueger's (Krueger 1983) system for interaction between two users. The system could determine who or what was touching it. It was a video-based approach which could track hands, fingers, and their owner through video cameras and could display these shadows on a projector.

At that time, the gesture recognition technique was important for the development of gesture interaction (Derpanis 2004; Mitra and Acharya 2007) since, without a proper recognition method, we could not imagine how we might benefit from adopting gestures as a form of interaction. Gestures could be captured and recognized through a variety of techniques, such as "data gloves", deep camera-based analysis and data mining. The

aim of gesture recognition is to identify a gesture performed by a user and interpret it for the computer. Basically, the recognition approaches can be divided into two categories: approaches using data gloves and vision-based approaches (Murthy and Jadon 2009).

**The data glove based gesture recognition** method was researched by Fels and Hinton (1993), Sturman and Zeltzer (1994) in the early stage, and then, in 2003, Tarchanidis and Lygouras proposed a method to recognise gestures by attaching force sensors to the rubber-coated data gloves using a particular kind of glue (Tarchanidis and Lygouras 2003). However these sensors were inconvenient for users when performing gestures because they were attached at the finger joints. The “blind finger gesture” was hindered by these sensors. Fahn and Sun addressed this problem in 2005. They proposed a new glove which had the sensors attached to the back of the glove instead of at the finger joints. This new glove also used magnetic induction coils as sensors to achieve better results of measurement while, at the same time, the service life was prolonged compared with other kinds of sensors adopted by earlier approaches (Fahn and Sun 2005). Then, in 2009, the KHU-1 data glove, consisting of three tri-axis accelerometer sensors, one controller and one Bluetooth, was developed by Kim et al. (Kim et al. 2009), taking development another step forward. The glove had a built-in 3D digital hand model which could track the hand motion and transform the signals to the PC via Bluetooth. Overall, these methods mainly relied on the sensors attached to the glove, which could capture and transform the motion of the fingers into electrical signals for recognition.

However, with the data glove-based recognition method, gloves or other similar equipment were needed when the user performed the gestures not only for a computer system but also on a mobile device. In 1993, IBM and BellSouth introduced the Simon Personal Communicator- the first mobile device which allowed users to interact by using gestures (Rodriguez-Feliz and Roth 2012). A stylus was required to navigate the menus and to input data. In the same year, Apple also released their touch-capable Newton PDA, which was the origin of the Message Pad. It still needed a stylus for inputting text (Gessler and Kotulla 1995).

**Vision-based recognition** methods have been intensively researched recently, as they can support easy and natural gesture interactions without the need for gloves (Erol et al. 2007). The most comprehensive review of vision-based recognition approaches

was completed by Pavlovic et al. in 1997 (Pavlovic et al. 1997). They reviewed the gesture recognition approach based on the three processes employed in gesture identification: modelling, analysis and recognition. Another review presented by Garg et al. (Garg et al. 2009), introduced the various aspects of gesture recognition, reviewed the available algorithms for gesture recognition and gave examples of the applied range of the recognition. The applications of vision-based gesture recognition were reviewed by Wachs et al. in 2011 (Wachs et al. 2011). The most recent study was done in 2015. It reviewed the previous work related to gesture recognition in the HCI field and analysed the advantages and shortcomings of vision-based recognition (Rautaray and Agrawal 2015). They found that the methods which required the physical contact of the user were uncomfortable, they were more accurate regarding recognition and implementation was less complex. Vision-based devices were more user-friendly but suffered from complex configuration and occlusion problems.

In 1999, iGesture was introduced by Wayne Westerman at the University of Delaware (Westerman 1999). With iGesture, additional equipment was no longer needed to achieve interaction. Users could use their fingers to directly interact with the multi-touch surface. Simple finger gestures, such as touch and slide across the screen, were supported in this approach. Westerman's work has become an example of a modern touchscreen-equipped device which is supported by gesture-based interactions.

Gesture-based interaction became more and more popular with the development of touch screen equipment technologies. The twentieth century was the time when gesture-based technologies flourished. Touch gestures have been used to fulfil many kinds of tasks in the HCI field. Gestures have been adopted even on devices with relatively small screens such as wearable watches. The touch gesture was first used on the Gesture Wrist in 2001 (Rekimoto 2001). Gesture Wrist is a wristband-type input device that can follow hand gesture directions; it allows users to use their fingers to interact with it directly. However, the gesture inputs are limited to simple single finger gestures, such as pointing, sliding, etc.

In 2006, the multi-touch finger gesture was used on the Tablet PC; it was introduced and demonstrated by Jeff Han at a TED Conference in Monterey, CA (Han 2006). With Han's interface, users could use their fingers to flick photos, stretch them out and pinch

them in an easy and intuitive way. At the same time, the gestures vocabulary was enriched due to the fact that the interface could support multi-touch. For example, pinch gestures could be used for zooming in or zooming out and gestures of two-finger dragging at the same time could be used to rotate an object.

One year later, Apple released their iPhone, stating that they had “invented multi-touch” as part of this device. Almost at the same time, Microsoft released their 10-inch tabletop touch platform named “Microsoft PixelSense” (ENDERLE 2012). Since then, the public have had the chance to be exposed to multi-touch technology, and can now use their fingers to control their phones and PDA devices.

In brief, the development history of gestures used in the HCI field can be viewed from four aspects. First, the form of the gesture used to interact with the computer system has changed from point-based gestures to in-air gestures and then, finally, to touch-based (point & path) gestures. Second, the number of fingers engaged in the interaction tasks has increased from one to several. Third, the devices have gradually become smaller (moving from computers and large screen projectors to tablets and mobile phones and then to wearable devices). Today, multi-touch gesture is widely used on smartphones and intelligent devices, such as the iPhone and iPad. Finally, the additional equipment, such as the stylus, which was needed at first, is no longer required. Now, users can directly use their fingers as a tool to interact with their smart devices anywhere and anytime they want (Touch Gesture).

### **2.3 The advantages of Touch Gesture and its usage scenario**

With the unceasing progress of user interface, from textual to 2D graphical, to touch and multi-touch screen interfaces (Garg et al. 2009), and the development of touch screen technology, touch-based gesture interaction has become a new dimension in mobile interaction (Li 2012). Touch gesture interaction is different from the traditional methods of interaction with graphic interfaces, and it has the advantage that it cannot be underrated. Note that the gesture in this section refers to the touch gesture.

### 2.3.1 Naturalness and gesture based NUI

Gesture is a powerful means of communication that is used frequently in daily life, and it is an important component of human body languages. The good communicative attributes of gesture make it reflect what humans think without needing any training. We always unconsciously use gestures to express our intended meaning. So, using gestures to achieve the communication between humans and computers is a natural way compared to the other traditional interaction approaches such as keyboards, pens, etc. (Fang et al. 2007). Researchers who have recognized this advantage, including Kipp et al. (2007), Stern et al. (2008), Song et al. (2012), and Chaudhary et al. (2013) in the HCI field, have done a lot of work in their attempts to benefit from the natural of gestures. Research conducted by Wachs and Wigdor has also shown that using gestures has great potential in establishing a natural user interface (Wachs et al. 2011; Wigdor and Wixon 2011). The potential is expressed in three main aspects. First, users can use gestures without requiring long periods of learning and adaptation due to their basic communicative form. Second, the cost of the gesture interface and the development of gesture detection, tracking and recognition technology promise more possibilities. Third, the gesture patterns used to control applications are easy to perform and remember.

Due to the naturalness of gestures, they are used as an approach for building up the NUI in the HCI field (Sanna et al. 2011 2013). In 2012, an NUI called Kinect Presenter (KiP) was introduced which allowed users using gestures to control the slides they were presenting (Cuccurullo et al. 2012). An NUI on a Microsoft Kinect was developed by Furbach and Maron (Furbach and Maron 2013), through which users could use their gestures to navigate through the virtually digital university building. For example, to turn left in the virtual building, users could move their left hand and point to the left. More recently, a study has been done to learn more about user satisfaction and enjoyment and the learnability of gestures adopted in an NUI for accomplishing simple computer tasks (Vaidyanathan and Rosenberg 2014).



### 2.3.2 Intuitiveness and gesture control

Apart from the naturalness of gestures, using them to interact is intuitive. Note that the “intuitive” used here follows the definition given by Naumann et al. (Naumann et al. 2007), which refers to interactions based on a user’s intuition. That is to say, if a user can apply prior knowledge unconsciously with less mental effort during an interaction, it is an intuitive interaction (Blackler et al. 2010). An example of showing the nature of a gestures’ intuitiveness is Fitz-Walter’s study. He claimed that using gestures to navigate Nintendo Wii menus is done in an intuitive way compared to the way of using the pointer technique (Fitz-Walter et al. 2008). The intuitiveness of the gesture makes it is very easy for a user to use. For example, users can directly manipulate an object with their fingers and the result (feedback) of the manipulation can be seen immediately. The convenience can be found in at least three aspects: 1) users do not need to control additional equipment such as a mouse or keyboard; instead they can use their fingers to interact (Poupyrev and Maruyama 2003); 2) it is much easier to manipulate an object using one’s fingers than typing a series of commands or choosing the desired action from a given menu of gestures (Minsky 1984); 3) it is easier to use gesture than virtual buttons when users are in a mobile context: e.g., while walking or being distracted (Bragdon et al. 2011).

Obviously, the process of using gestures to control a virtual object is similar to control a real object: for example, rotating and moving an object. Since the users do not need to invest a lot of energy in learning how to control a virtual object with gestures, using gestures has been adopted in many cases.

1. Gestures have been used to control objects, remote devices or objects in remote devices.

Gesture Avatar (Lü and Li 2011) is an interface where users can draw gestures that represent the target objects and then control them. For example, the user can draw a rectangle gesture to represent the video slider and select it as the target, then use dragging gestures to control the video slider. The AirTouch panel (Lin et al. 2013) allows users to control remote home appliances, such as the television or the air conditioner, by gestures. Similarly, U-Remo (Ujima et al. 2014) is another approach to control home electronics. User-friendly gestures set for control tasks has been developed. Doors can be controlled by gestures, and this approach is

more convenient than traditional methods for users to open, close, lock and unlock the door. Moreover, this is a great solution for physically challenged people (Zeiß et al. 2014). The Gesture Mote allows users to use gestures under many circumstances (including pointing, clicking, navigation and shortcut gestures) to interact with objects on a remote display (Lü et al., 2014).

2. Gestures have been used to control the objects on a large display.

Since the objects in large displays are out of a user's reach, they are hard to control directly via touch gestures. A remote pointing method has been introduced to address this problem. Users can select, rotate, and resize photos displayed on a large screen by remote gestures using a pinching gesture with the thumb and forefinger (Tochihara et al. 2016). Handyscope (Kuribara et al. 2014) has introduced a technique using pull-out gestures to control objects on a large display which allows multiple users to interact with the same object simultaneously, without conflict.

3. Gestures have been used to control music players.

An innovative approach has been presented by Strachan et al. (2010). They give a conceptual demonstration of how to use gestures to interact with and control a music player.

4. Gestures have been used to control the browsers.

Aside from the common tasks that we frequently encounter when browsing, the flick gesture for controlling the browser to go backwards and forwards has been comprehensively studied by Moyle and Cockburn (2003). Billingham and Vu (2015) have proposed a natural gesture set for common web browsing tasks, including submitting forms, going back to previous pages, etc.

However, unwanted movements, accidental touching, occlusion and other problems have hindered the widespread adoption of gesture control. Palomares et al. (2014) proposed two different methods to reduce this problem. They used the "Borders" surrounding the object to reduce the occlusion problem which arises when a whole screen object is touched by multi-fingers. They have proposed a technique involving "the help of a proxy" to address the unwanted movement issue. To move an object, the user needs to

connect an object to the proxy. The user must first tap on the proxy and then tap on the object and then a line is drawn between the two objects. Then, the proxy movements are reflected in the object. Our methods are different from those mentioned in the research. Both PGS and the Intelligent launcher do not require the UI to be reconfigured, nor a "proxy" to be designed. We use natural gestures to control the objects which the user intends to access.

### **2.3.3 Estate free and usage on smart-watch**

Using gestures has another advantage. It is space-saving when used on small screens. Unlike keyboard or virtual buttons, gestures do not take up the screen's real estate (Nacenti et al. 2013) due to the combination of the display and the input space (Poupyrev and Maruyama 2003). For today's smart device, space-saving is important, especially when the screen size is limited. One of the reasons is that the screen becomes crowded with the increasing amount of content that needs to be displayed; thus, for the input area, it is a case of the smaller, the better. Another reason is that the input area may sometimes overlap the information or virtual buttons needed (Wigdor et al. 2007); gestures may be one way to save the input area for displaying bigger buttons.

With traditional touch-based input methods (virtual keyboard or menu), the users' hands may block out necessary information appearing on the screen. This problem is especially significant on the smartwatch since the screen real estate is limited. The way to overcome this disadvantage by using gestures to interact with these small screens has been studied. Arefin Shimon et al. (2016) explored how to use gestures to extend the input space for 31 common smartwatch tasks. In the same year, another approach of using 16 bezel to bezel (start from bezel and end at another bezel) swipe gestures to enrich the input vocabulary were presented by Kubo et al. (2016).

### **2.3.4 Rich vocabulary and usage for inputting**

The gesture vocabulary (Nielsen et al. 2003) can be extended easily, as gestures have the potential to provide an unlimited number of possible interactions (Heuwing et al. 2015). This advantage is especially obvious in touch gestures compared with in-air gestures,

since 1) the number of in-air poses that users can easily perform is limited, and 2) the recognition of complex in-air gestures depends on the development of corresponding techniques. For touch gestures, almost every finger point path that a user may perform can be a new gesture to extend the scalability of touch gesture vocabulary.

When considering operations such as insert, delete, etc. during the input/editing and the text/graphical entry, there are two parts to the task. A study involving five experiments was done to monitor the gesture flexibility when using it to accomplish common operations occurring during the editing of text and drawing pictures (Gould and Salaun 1987). It was found that the users in the experiments tended to use similar gestures for the same tasks (e.g., move, insert, etc.) when editing. The results of this research inspired the editor designers to develop a better editing system. In the same year, Wolf and Morrel-Samuels (1987) proposed a template for operations required in editing tasks, such as a “” gesture can launch the “insert” command. They encouraged interface developers to build the editor interface based on their findings. These two pieces of research have confirmed that using gestures for editing text or pictures is feasible since users have shown great consistency in creating them and can easily recall them in the editing task.

Simultaneously, gestures are also effective for text/graphic entry. First, gestures are very flexible and can be used to extend non-alphanumeric input (Findlater et al. 2012). Gestures can enrich input vocabulary (Roudaut et al. 2009; Song et al. 2011; Heo and Lee 2011; Heo et al. 2014): what can be inputted by keyboard can also be done by gesture. Furthermore, gestures are especially helpful for information that is hard to express or describe (e.g., shapes or symbols). InkAnchor, proposed by Ren et al. (2014), is a digital ink editor that allows users to easily create ink-based notes by drawing and writing on a mobile phone touchscreen with one’s fingers. Second, gestures can assist users to input with a current virtual keyboard.

The performance of gesture inputting may be good on a smartphone-sized device, but when the operating area is squeezed, such as on a smart watch, it is hard to ensure the same performance compared with devices with bigger screens. To address this issue, a finger-mounted fine-tip stylus, called a “NanoStylus” has been introduced. With the stylus, users can point on a smart watch quickly and accurately, with no occlusion (Xia et al. 2015). Even though a stylus is needed, a one-dimensional finger-interaction interface

has been developed which can greatly save the input area (Yu et al. 2016). Users can use single-stroke gestures in one-dimension, such as a line gesture, to input on wearable glasses. The vocabulary and the learnability of this interface are balanced by the developer. All the words in English can be input through this one-dimensional interface. Another approach using fingers to input has been developed by Gordon et al. (2016). They used human motor control adaptability, modern statistical decoding and error correction technologies to achieve good gesture typing performance on smartphones.

### **2.3.5 Eyes-free and usage in moving environments**

Gestures could be eyes-free: that is to say that users do not need to keep their eyes on the screen when performing gestures. This is confirmed by an experiment in which the eye-gaze data of the user looking at the phone were recorded (Bragdon et al. 2011). The data was recorded when a user completed given gestures tasks. The tasks were designed to mimic three different levels of distraction (i.e., No Distraction, Moderate Situational Awareness Task and Attention-Saturating Task). In our everyday lives, when users need to pay attention to other things, such as driving, eyes-free interaction is especially important (Ecker et al. 2009).

Therefore, in addition to the ease of performing gestures in moving environments, when users need to pay attention to other things, using gestures is more convenient than other interaction methods. Two examples of the benefits of using gestures are: gesture have been used in interactions with the navigation system on both a bike (Dancu et al. 2015) and in a car (Alpern and Minardo 2003; Ahmad et al. 2014; Werner 2014).

### **2.3.6 Privacy and usage for authentication**

Touch gestures are highly private, which has been confirmed by users when interacting with computers (Siong-Hoe and Hui 2012). It is more private than voice or mid-air gesture-based interaction in public (Ahlström et al. 2014) as touch gestures can be performed in a silent and secret way compared with other interaction methods.

Concerning the traditional password, it is either too short and weak or too long and difficult to memorise, so the user needs to find a balance between the password's security

and its memorability. Gesture-based security authentication is a better alternative solution because gestures have a better memory than text, based on the picture superiority effect (Paivio and Csapo 1973). This means graphic items (e.g., drawing gestures) are more likely to be remembered than words. This was confirmed by Chong and Marsden (2009), who also proved the effectiveness of the gesture password: a gesture-based authentication scheme on a mobile. One year later, GesturePin (Chong et al. 2010) was proposed with the same idea of using a gesture password to reduce the users' cognitive load. The research confirmed that a user could achieve a better performance regarding both speed and accuracy when using a gesture-based authentication method than the traditional PIN authentication method. SwiPin (Von Zezschwitz et al. 2015), using touch gestures instead of traditional PINs, makes it secure against human observers. Considering users can be observed when entering their PINs, anyone observing only has a 0.16% chance to guess the correct gesture PIN but has a 100% chance for the traditional PIN (Von Zezschwitz et al. 2015).

Additionally, more recent research has shown that, compared with textual authentication methods, gesture-based authentication schemes perform better in terms of speed and memorisation. At the same time, users are more willing to retry by gesture authentication (Yang et al. 2016).

Inspired by these advantages brought by using gestures for authentication, researchers began to adopt gestures for many different authentication tasks. Shahzad et al. (2013) proposed a gesture-based user authentication scheme to unlock touch screen devices securely. Zhao et al. (2013) brought forward a gesture-based approach for user identification authentication on mobile phones. TIPS (Feng et al. 2014) extended this work by enhancing the identification process in uncontrolled environments. They adopted a context-aware technology which can achieve gesture identification accuracy of over 90% in real-life conditions.

With the advantages above, the touch gesture has become a popular interaction method in today's user interface.

## Chapter 3

# Personalized Gesture Shortcuts for Frequent Used App Access

### 3.1 Introduction

Recent research on touch screen gestures has acknowledged the convenience of using gesture shortcuts to access system functions (e.g., wireless on/off), execute system commands (e.g., rotate) (Morris et al. 2010; Wobbrock et al. 2009) and even perform specific tasks (e.g., web browsing) on touch screen devices (Billinghurst and Vu 2015). However, these gestures have been predefined by system developers and users need to learn them first before they can be used. For example, the "long press" gesture is used for "turn on wireless": users can turn the wireless on by either using a long press on the wireless icon on the screen or go to the "system setting" and find the subtitle "Wireless & Networks" and select "WLAN" and then tap on it. Obviously, the shortcut "long press" is much easier for users, but it will be a problem if the user does not know that this is the shortcut. In other words, such a method would be helpful to a user only after a learning process. During the learning process, the user needs to learn what the shortcuts can do according to the system definitions.

Further to this, several studies have noted that Personalized Gesture Shortcuts (PGS) are more easily memorized by users when compared with system-defined gesture shortcuts (Nacenta et al. 2013). Therefore, this thesis has been motivated to focus on using the more remember-able method for accessing target apps. The PGS method allows users to create their own shortcuts to access the target apps they want to open. They can use whatever touch gesture they choose as the shortcut. Such as a "location mark" gesture for accessing "Google Map" or the gesture letter "W" for accessing the "WeChat" app. These gestures are not limited by the system or the system developer.

Despite the merits of personalized gesture shortcuts, current studies only focus on system functions. So, it is still unclear whether similar approaches can also be applied to mobile apps. One may argue that the representation of system functions is very similar to the representation of mobile apps on the smart phone user interface (e.g., icons with/without textual information). The fact is, apps present a much higher degree of complexity and diversity than system functions as users can choose what apps to install, how many they want to install and even how many similar ones they want to install (e.g., Skype and LINE are both commonly installed as instant messengers). This "freedom of choice" can trigger an immediate question, which is: although users can define a gesture shortcut for a mobile app freely, can they always recall the gesture easily?

This chapter presents an exploratory study that aims to understand user memorisation of PGS for accessing mobile apps, starting with 15 commonly-used mobile apps to identify several factors that may influence the recall performance and then discussing how performance could be improved, from a technical perspective.

## **3.2 Methods**







### **3.2.1 Gesture shortcut definition strategy**

A previous study on understanding how users define gesture shortcuts for commonly-used tasks on mobiles has noticed that users are selective, but they tend to only use certain strategies instead of drawing gestures randomly (Poppinga et al. 2014). These strategies can be classified as three types: the name-based strategy, the system icon-



based strategy, and the function/purpose-based strategy. Accordingly, we add the “other” strategy to cover those gestures with unknown user behaviours, and define four definition strategy categories based on the ways mobile apps are displayed, as well as their purposes. Table 3.2.1 shows an example of the strategies and the corresponding gesture shortcuts suggested by users.

Table 3.1: Examples of Definition Strategy. The gesture shortcut examples listed in this table were provided by users.

Strategy	Visual	Textual	Functional	Other
Icon				
Gesture Shortcut				

### 3.2.2 The Agile Search App

Several attempts related to using gesture shortcuts to access apps have been made, but there is no native support for personalized gesture shortcuts offered by current smart device systems. In order to enable the support and capture of these shortcuts, this thesis has designed, developed and built the Agile Search app, a standalone Android app that allows users to define gesture shortcuts to launch associated apps. The gestures can be both uni-stroke or multi-stroke gestures proposed in Zhai et al. (2012). Any stroke(s) performed by a user can be and is allowed to be a potential gesture shortcut. In other words, it is the user who has been given the freedom to choose the gesture, not the system developer.

The Agile Gesture app was developed using Java for Android. The development environment was Apple’s Android Studio 2.2 (Android 2019b). Agile Search uses Google’s gesture recognition algorithm (Lemort and Vales 2013), following Gesture Search (Lü and Li 2012). The algorithm provides a method for the continuous recognition of multi-touch gestures performed by users on the multi-touch input device. The Agile Search app was built based on the Android SDK (Android 2019a), which also uses this algorithm. It allows developers to design custom gestures, which are then stored in a gesture file bundled with an Android application package (i.e., Gesture Library). These custom gesture files are created and supplied as part of the Android SDK.

The Gesture Library can then be used to compare the input gesture to those contained in the gestures file previously loaded into the application. The Gesture Library reports the probability that the gesture performed by the user matches an entry in the Library by calculating a prediction score for each gesture. A prediction score of 1.0 or greater is treated as a successful match between a gesture stored in the file and that performed by the user on the device display. This thesis used this most popular gesture recognition algorithm as it has been used in previous work, such as that done by Li (2010a), Lu and Li (2015) and Lulu and Kuflik (2015). Consequently, this thesis achieved the same gesture recognition accuracy as these related studies.

Two steps are needed to personalize a gesture shortcut under the Setting mode. First, a user needs to select an app that they intend to access via gesture shortcuts among the installed apps' list populated in Agile Search. Second, the user draws a gesture shortcut for the app and confirms whether they are satisfied with the gesture (by pressing the "confirm" button). If the user is not satisfied with the gesture shortcut, they can redraw it (by pressing the "redraw" button). Once the gesture is confirmed, Agile Search opens the target app and saves the gesture as a PNG file locally. After defining the gesture, the user can launch the app by redrawing the gesture in Agile Search under the Using mode. If the gesture matches what the user defined earlier, Agile Search will then launch the app associated with the gesture (see Figure 3.1).

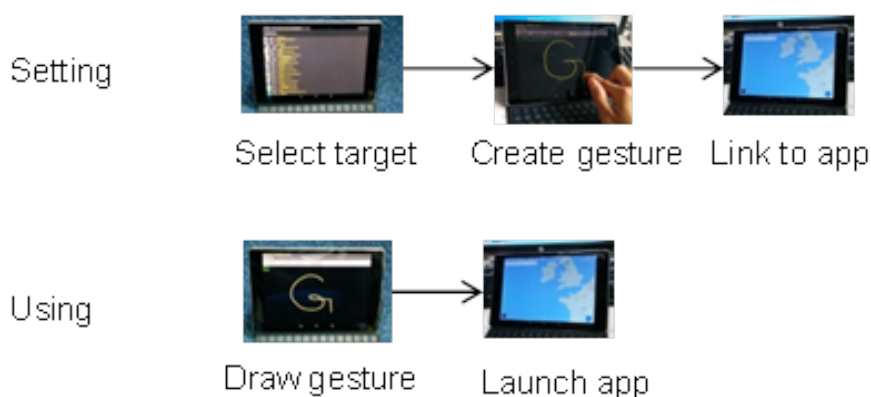


Figure 3.1: Workflow of Agile Search app. **Setting** and **Using** are two working modes and can be toggled by a button at the top right of the app.

### 3.2.3 App sampling

Due to the diversity and the large number of installed apps, it is impractical to ask users to draw gestures for each for this exploratory study. To sample the apps, a user survey was developed in which participants were asked to select the apps they used at least once a week from a list of 40 extracted from Google Play's top downloads chart. 67 students responded to the survey. Eventually, 15 apps commonly used by 33 students were selected, as shown in Table 3.2. To narrow down the app accessing experience (history), the students who installed and used (accessed) these 15 apps at least once a week were selected. Therefore, the app recall accuracy of this study does not differ due to some apps being used more frequently than others.

The categories in Table 3.2 are based on Google Play's categorisation (Google 2018d). The 33 students were invited to participate in the experiment via email. The students who did not select the required apps also received an email as a response, expressing thanks.

Table 3.2: Apps selected for the experiment.

category	name	abbreviation
communication	Wechat	wcht.
	Messenger	msgr.
	WhatsApp	wapp.
	Facebook	fb.
	Skype	skyp.
	Chrome	chrn.
	Outlook	otlk.
	Gmail	gml.
travel	Tripadvisor	trip.
	Booking	bk.
	Maps	map.
entertainment	YouTube	y tub.
finance	Paypal	pp.
photography	Instagram	ins.
shopping	Amazon shopping	ashp.

### 3.2.4 Measurement

To understand the users' recall of their self-defined gesture shortcuts and explore the factors that may influence their recall, two measures are defined: (1) gesture shortcut recall rate and (2) definition strategy recall rate. For (1), it is defined as the proportion

of gesture shortcuts for a specific number of apps that are recalled successfully by a specific number of users. For (2), it is defined as the proportion of definition strategies for gesture shortcuts that are recalled successfully by users. Note that this calculation does not consider whether a user has successfully recalled a gesture shortcut or not. This thesis also investigate whether the (2) can be a factor for improving the AI prediction accuracy in Chapter 6.

### **3.2.5 Experiment design**

#### **Participants**

During the app sampling process mentioned in the Methodology section, 33 university students who studied different subjects nominated 15 common apps. The participants included 15 females and 18 males with an average age of 22.3 years (SD= 2.6). These students were then invited to take part in this experiment. They are all had been active smart phone and tablet users for over three years and owned at least one smart phone or tablet. They all had experience in accessing mobile apps on different devices for at least three years. They all had less experience in using the gesture method to access apps. 30 out of the 33 participants reported they had no experience in using the gesture method to access an app. Three of them had installed Gesture Search before but had not used it in more than one year.

#### **Testing device**

A Google Nexus 9 tablet with Agile Search pre-installed was given to the participants during the experiment to avoid compatibility issues when installing the app on their own devices. This also suggested that our app could be used on Android mobile devices such as a smartphone or tablet.

## Procedure

The participants were asked to first familiarize themselves with the device, if they had not used a Nexus 9 tablet before. After that, they were shown how to use Agile Search and were offered a 15-minute self-practice session to try it out with five apps (not included in the sample apps' list) pre-installed on the device. The experiment began when the participants confirmed that they were ready for the test, following the practice. Then the participants were asked to draw in order to define a gesture (encoding) or recall the gesture via the Agile Search. After each gesture was drawn via the Agile Search app, the participants were asked to report the strategy they had employed for this gesture. The strategy is explained in Section 3.2.1.

Each participant was asked to come to the lab on two consecutive days to complete the trial consisting of different sequenced phases. Participants were asked to complete the gesture shortcut definition (Encoding) (Day 1), Report strategy (Day 1) and Reinforcement in a sequence on the first day of the experiment. The Recall Gesture (Day 2) and Report strategy (Day 2) tasks were completed by the participants on the second day of the experiment (see Figure 3.2). The time interval between the two days was set at 24 hours. This time interval was decided upon following previous studies focused on users' memorisation of the gesture shortcut (Nacenti et al. 2013). Note that, in practice, we tried our best to ensure the second day's recall experiment was conducted as close to the 24 hours after the first day's definition experiment as possible. The participants who had done the first day of the experiment in the morning were asked to come to the lab on the second day in the morning as well, as with the afternoon participants. The actual interval times were 22 to 26 hours. We believe that this did not affect the study results.

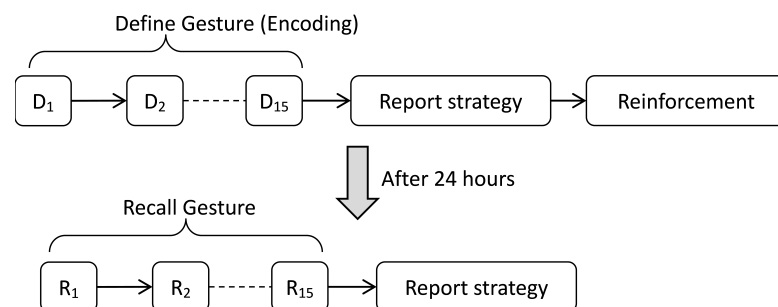


Figure 3.2: The sequence of the experiment.

### 3.3 Results

#### 3.3.1 Gesture shortcut recall

An overview of the participants' gesture shortcut recall rate is shown in Figure 3.3. The average recall rate was 90.9% (SD= 10.61%, range of 60% to 100%). Out of 33 participants, 16 recalled all the gesture shortcuts they had defined correctly. For the remainder, only two had a recall rate under 80% (P11: 60%, P23: 67%).

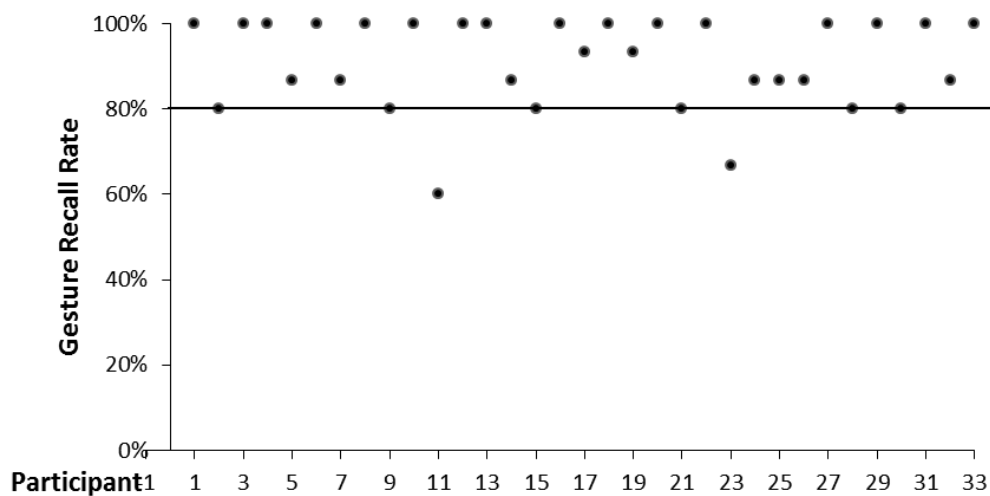


Figure 3.3: Overview of participants' gesture shortcut recall rate.

#### 3.3.2 Definition strategy recall

Figure 3.4 shows an overview of the participants' gesture shortcut definition strategy recall rate. The average recall rate was 91% (SD=10.48%, range of 73.3% to 100%) indicating most participants remembered the strategy used for defining gesture shortcuts. In detail, 12 participants remembered all the strategies they had used correctly, while 4 participants remembered fewer than 80% of the strategies they had used.

#### 3.3.3 Recall failures over apps

Figure 3.5 shows the detailed breakdown of gesture shortcut recall failures over apps. The gesture which failed in recall but which the participants reported they used with the

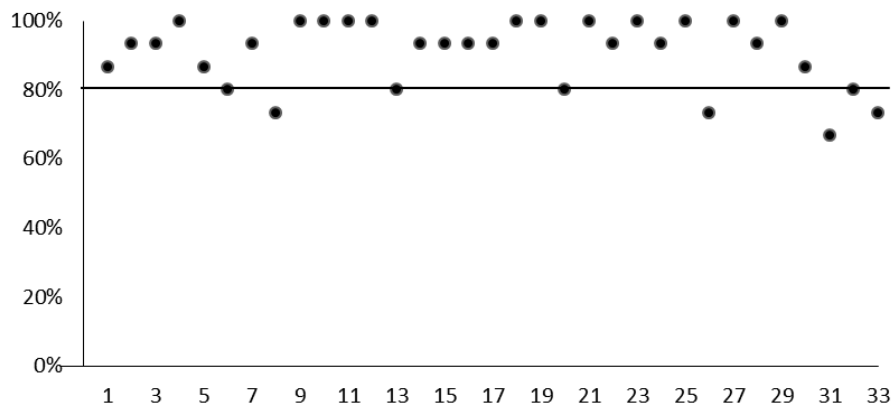


Figure 3.4: Overview of participants’ gesture shortcut definition strategy recall rate.

same gesture definition strategy was calculated as a **Consistent Pair**. For those failures caused by the inconsistency in definition strategy were calculated as an **Inconsistent Pair**. Therefore, the bar chart is divided into two parts: the top part represents the number of failures resulting from the same gesture shortcut definition strategy and the bottom part represents the number of failures resulted from mis-remembered strategies. Note that the height of each bar represents the total number of failures registered for each app among the 33 participants in the recall test. The participants were able to remember the gesture shortcuts when they used the same defined strategy (gesture shortcut recall rate: 96.4%); otherwise, the recall rate dropped significantly to 31.1%.

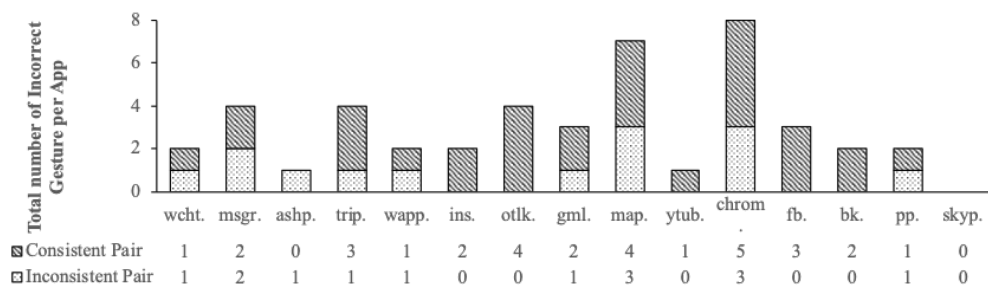


Figure 3.5: Detailed breakdown of gesture shortcut recall failures over apps.

Figure 3.6 shows the detailed breakdown of failures in the recall of the gesture shortcut pairs (definition vs. recall) recorded for apps when the definition strategies were misremembered. The results reveal that 8 recall failures were cases where participants misremembered the textual strategy used in the definition phase as a visual strategy in the recall test phase; and 6 failures were related to misremembered the visual strategy as a functional strategy.

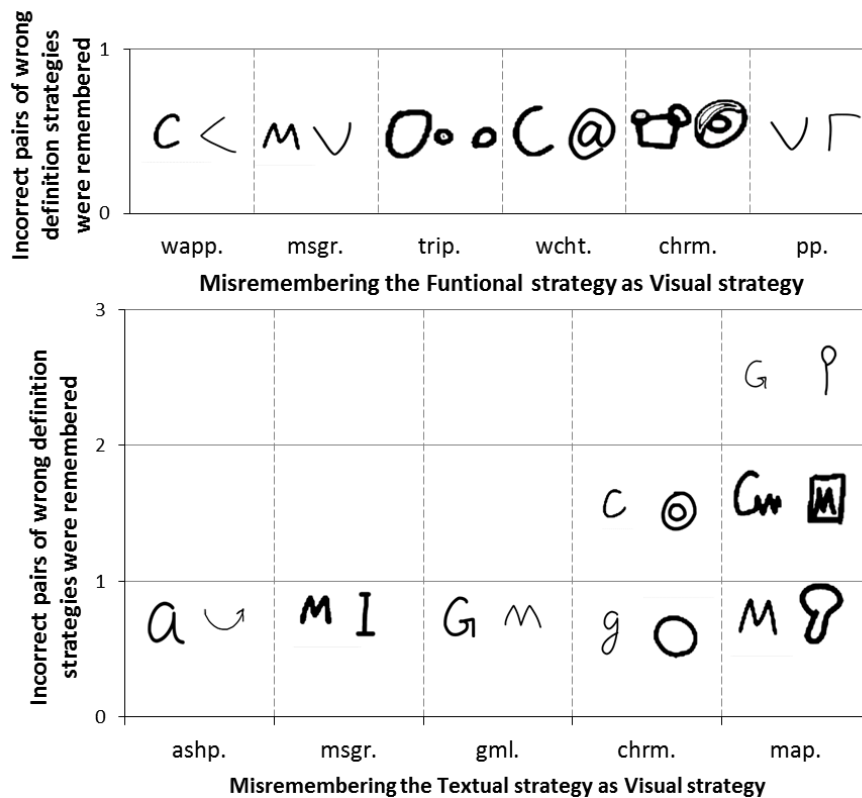


Figure 3.6: Detailed breakdowns of incorrect gesture shortcut pairs based on different definition strategies.

Figure 3.7 shows the detailed breakdown of failures in recall of the gesture shortcut pairs (definition vs. recall) recorded for apps when the definition strategies were remembered. There were 17 failures related to the textual strategy, 6 failures related to the visual strategy and 5 pairs related to the functional strategy. Note, there were also 3 failures related to the "other" strategy that are not presented in the figure, but they are discussed in the next section.

### 3.4 Discussion

The experiment results show that, compared with recall accuracy reported in Nacenta et al. (2013), participants effectively recalled the gesture shortcuts in most cases. The average recall accuracy of PGS on the next day was 79% for Nacenta et al. (2013) and 91% for our experiment. This indicates the effectiveness of using self-defined gesture shortcuts to access frequently-used mobile apps. However, the failures should not be



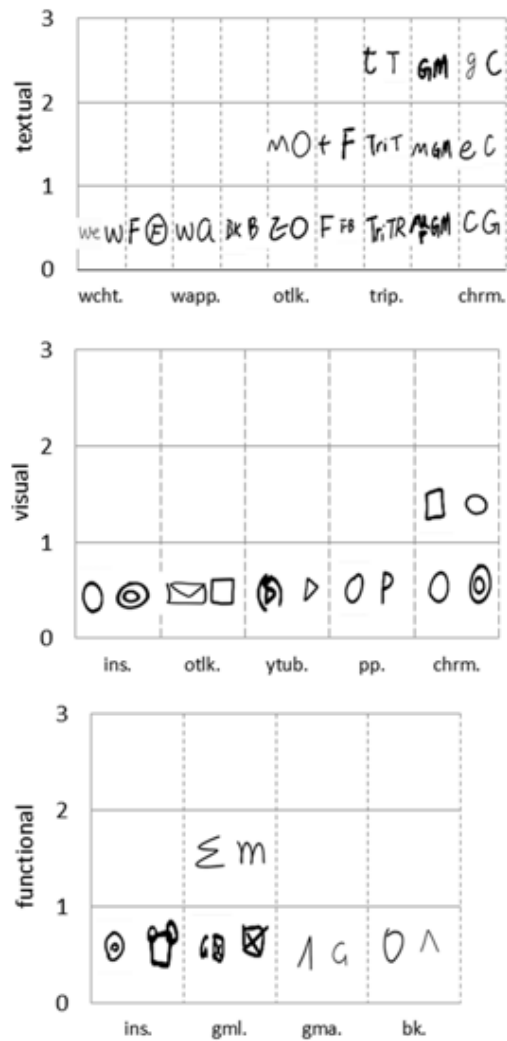


Figure 3.7: Detailed breakdowns of incorrect gesture shortcut pairs based on same definition strategy.

ignored as they were found with almost all apps (14 out of 15) in the sample list. After examining the link between the definition strategy recall and gesture shortcut recall, this thesis found that the users could successfully recall most of the gestures when the definition strategies used were consistent between define and recall. This suggests that a mechanism to help users better remember the definition strategy they have used is needed to improve their gesture shortcut recall. That is to say, once the mechanism tells the user to create their gesture based on a specific strategy, then the user can only recall the gesture by following this strategy. In this way, the recall error caused by inconsistency with the gesture define strategy can be eliminated. Therefore, it would be helpful if the users' choices of strategies can be limited to just one option. The option can be provided

as a hint when a user is trying to recall the gesture shortcuts.

It was also found that many failures were related to the textual strategy (i.e., the app's name) as participants could not remember the exact letter they had used when defining the gesture shortcuts. These failures are shown in the top part of Figure 3.7 (17 pairs). For example, "G" was defined for "Google Map" but it was recalled as "M" or "GM". These cases suggest that the gesture-matching algorithm for any interface like Agile Search should be more flexible to allow for the use of either capital or lowercase letters or any letter or combination of letters from an app's name as correct gesture shortcuts, regardless of what was defined as the initial gesture shortcut.

The participants' confusion regarding uni-stroke and multi-stroke gestures was another cause of recall failure. For example, the circle gesture shortcut was first drawn for Instagram, but then changed to a nest circle gesture shortcut later. This suggests that only allowing uni-stroke gestures could be a means of reducing the probability of gesture shortcut recall failure.

Also worth to mention is that only a very few number of failures were related to the "functional" (5 pairs) and "other" (3 pairs) strategies. Unlike the textual and visual strategies where cues can be easily drawn from an app's name or icon, the "functional" and "other" strategies are very vague. For example, users may think "Google Map" is a map app but they can change their mind and think of it as a navigation app over time, depending on their main usage (Figure 3.6). Therefore, a possible solution is to limit the definition strategy to the textual and visual only to avoid the confusion. This exploratory study has a number of limitations due to the controlled environment of the experiment.

First, the time interval for running the recall task was set at 24 hours for all the participants. This does not seem to represent a real-life scenario as in reality users may be able to launch an app using their self-defined gesture shortcuts more often, which may help them remember the shortcuts better. Therefore, different intervals based on the user's actual app use behaviours need to be considered in future experiments to obtain a more accurate understanding of their memorisation of self-defined gesture shortcuts. In future experiments, more intervals should be set to mimic real life; these could be immediately after the gesture definition, 24 hours and 7 days after.

Second, the sample apps in this exploratory study were the apps most frequently used

by participants. It remains unclear whether such findings can be applied to infrequently used apps as users may have more memorisation issues with these apps. Therefore, the memorisation of gesture shortcuts for accessing the infrequently used app should be investigated in future experiments.

Third, in this exploratory study, our analysis has been mainly focused on strategies that were clearly identified, such as the visual strategy, textual strategy and functional strategy. Although the "other" strategy was also used, we did not look into these strategies due to the very small number of incorrect gesture shortcut pairs presented. In the future, we may also look into this category when more sample apps are used in the experiment.

Finally, the recall of definition strategies was only checked within 24 hours. It remains unclear whether users would change their definition strategies after they have been used for some time or which definition strategy is more effective. Therefore, future experiments focusing on a single strategy need to be considered.

### **3.5 Summary**

An exploratory study has been conducted to study the ability of users to recall their self-defined gesture shortcuts to access mobile apps, as well as understand the strategies which they used to define these gesture shortcuts. The results were quite promising, with an average recall rate of 91%, which demonstrated the effectiveness of using self-defined gesture shortcuts to access an app. Certain factors that may have influenced users' recall rates have also been noticed in the experiment and possible solutions to improve recall have been discussed. Despite the findings, the study clearly has some limitations. For example, this section only examined the recall accuracy of the apps which users accessed multiple times over a seven-day period. This section did not check the speed advantages of our proposed PGS method. So, our study will now focus on addressing these limitations and aim to provide recommendations on the design of gesture shortcut definition interfaces and icon designs.

## **Chapter 4**

# **Personalized Gesture Shortcuts for App Access on Smartphones**

### **4.1 Introduction**

The increasing number and variety of apps in the app stores have persuaded smartphone users to install more and more apps than ever before. This, however, has also made it more difficult for users to locate target apps quickly. Moreover, the requirement for finding the target app is very often. Based on KPCB's report, smartphone users access installed apps 150 times per day (Hodgkins 2015). Thus, finding a more efficient and convenient way for app access is becoming crucial.

In this thesis, we propose a novel personalized gesture shortcut (PGS) approach which allows users to use their self-defined gestures rather than those defined by the system-developer to access the target app rather than system functions (more details see Chapter 3). This approach is implemented as an Android app and installed on an Android tablet in Chapter 3. The user can quickly and easily access a target app by drawing the shortcut gesture they have defined rather than look for it through the whole list of apps. The key to this approach is that the user can remember their PGS when they

want to access the target app. Therefore, in Chapter 3, this thesis checked to confirm whether and how well users could remember their PGS for 15 frequently-used apps.

In our daily life, not only the 15 apps we selected in Chapter 3, but all the apps that have been installed are potential target apps. Therefore, to extend the findings in Chapter 3, this chapter aims to investigate the PGS recall rate for not only frequently but also infrequently used apps. Using gesture shortcuts directly to activate an app can simplify the apps' look-up process, but the performance of such shortcuts has not been investigated in previous research. To know how much of an advantage the PGS approach can achieve, this chapter will also check the time spent accessing a given target app and compare this with the conventional method: the Graphic User Interface (GUI).

This chapter evaluates the PGS and GUI methods in an experiment with 36 participants. The results showed that (1) the proposed approach produces shorter access time than the conventional graphic interface; (2) the speed advantage is mainly seen with infrequently used apps; (3) the margin of the speed advantage is roughly the same, whether the test is given 24 hours or 7 days after the initial personalized definitions of the gesture shortcuts; (4) the advantage over the conventional graphic interface is greater when an app is accessed among a relatively large number of apps; and (5) the gesture recall accuracy declines as a function of retention interval between the initial personalization and the time of recall. These results show an encouraging future for the potential application of the proposed method for more efficient app access on smartphones.

## 4.2 Methods

To investigate whether PGS could facilitate fast app access on smartphones, we compared it with GUI. This section will introduce the Agile Gesture Android app that we have developed to make this comparison. We will then explain the factors in our experiment, which include our rationales for classifying the installed apps based on frequency of usage, for choosing the retention interval for recall tests and for selecting the app sample set for the experiment. The details of the method, design, and the task procedures and tasks involved in the experiment will be provided.

### 4.2.1 GUI

This refers to the current graphic interface found on most touch-screen devices. Users typically access an app by touching its icon. They can navigate backwards and forwards to different screen pages to find a target app. Figure 1.1 shows the process of accessing a target app via this graphic interface. We chose to compare our proposed method with GUI for two reasons. First, currently, the GUI is the most popular app accessing method among smart device users. Users are all familiar with this method. Second, GUI is the basic system-level interface which is provided by almost every smart device provider. Users do not to pay extra for this method.

### 4.2.2 Agile Gesture Interface

For our purpose, we needed to collect data on both PGS recall accuracy and app accessing time. The app mentioned in Chapter 3 could not fulfil this need. The Agile Search app can only store the predefined gestures and then calculate the visual similarity of the given gesture to the ones in the gesture library. However, the length of time the user spends accessing a target app via either Agile Gesture or GUI cannot be recorded by Agile Search. Therefore, we developed Agile Gesture, using Java for Android. We choose Android again because first, it is widely used and second, because it is an open source so anyone who is interested in this project can build their own app based on this work. The development environment we used was Apple's Android Studio 2.2. The app implemented PGS which we used in our experiment allowed users to access an app by drawing a predefined gesture that was associated with the app. To create the association, a user needed to define a gesture shortcut first. After this, the shortcut could be recalled and redrawn when the app was needed, as seen in Figure 4.1.

To create a personalized gesture, users pressed the "Add" button shown on the interface. If they were not satisfied with the gesture they had just created, they could press the "Bin" icon in the top right corner of the Agile Gesture app to delete it and then redraw a new gesture. This could be repeated until they were satisfied, and then they could press the "File" icon to save it. After this, they needed to tell Agile Gesture which app they intended to access by selecting the target app from the installed apps list. This completes

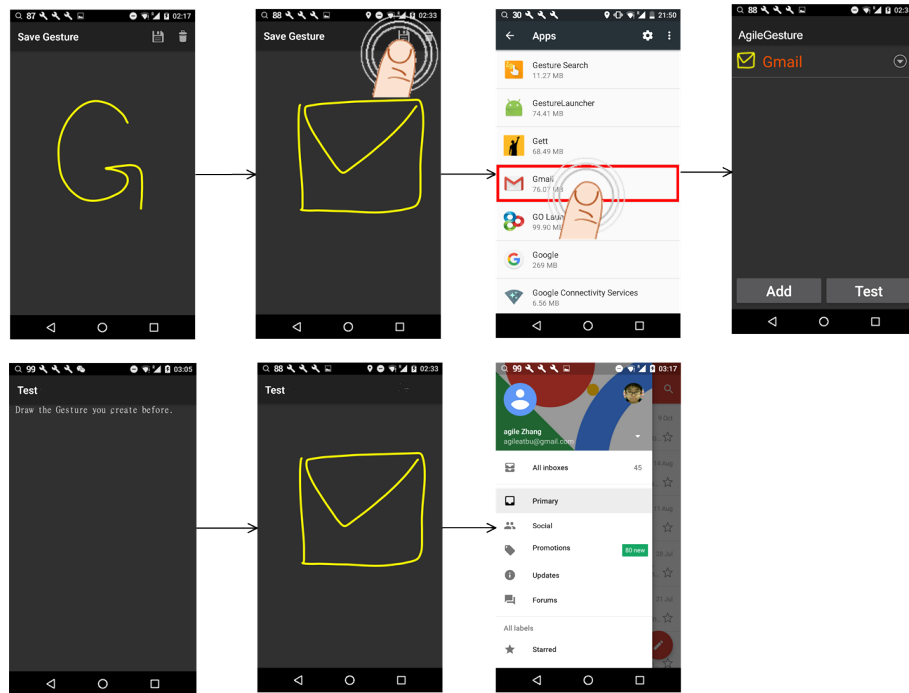


Figure 4.1: Agile Gesture for app access. Top row: procedure for app personalization. Bottom row: procedure for testing app access.

the process of personalizing the gesture shortcut. Once a shortcut had been created, users could press the "Test" button to access the target app. This launched the app if they were able to redraw the gesture shortcut correctly. The gesture recognition algorithm Agile Gesture used here is the same as Agile Search in Chapter 3. Therefore, the gesture recognition accuracy which relates to the recall accuracy is same as discussed in Chapter 3.

Agile Gesture also records the time a user takes to access an app. This was designed to provide a performance measurement for the experiment in this study. The timer of Agile Gesture starts as soon as the "Test" button in Agile Gesture is pressed. It stops when the gesture is completed. Notice that the time spent on launching the app is not included. Agile Gesture automatically recorded the time of each test trial during the experiment. For the GUI method in the experiment, the timer of Agile Gesture started as soon as the participants pressed the "Home" button and stopped when they touched the app icon. Similarly, the app launching time was not included. Two example outputs which ran in the background of Agile Gesture when a participant accessed Gmail via PGS and GUI are shown in Figure 4.2.

```

I/OpenGLRenderer: Initialized EGL, version 1.4
W/OpenGLRenderer: Failed to choose config with EGL_SWAP_BEHAVIOR_PRESERVED, retrying without.
D/OpenGLRenderer: Enabling debug mode 0
D/AppMonitorService: run: 2017-10-09 04:35:06 com.uk.bu.chi.gesturelauncher
D/GestureLauncher: FloatWindowBigView: viewWidth=-1 viewHeight=-1
D/GestureLauncher: PGS method is used
D/GestureLauncher: onTestMode: Gesture Starting: 2017-10-09 04:35:07.23
D/GestureLauncher: onTestMode: Gesture Ended: 2017-10-09 04:35:09.56
D/GestureLauncher: onTestMode: Gesture Time: 2.33s
-----
D/GestureLauncher: CGI method is used
D/GestureLauncher: onHomeButtonPressed: Home button pressed 2017-10-09 04:36:24.25
D/AppMonitorService: run: 2017-10-09 04:36:28.07 com.google.android.gm
D/GestureLauncher: onTestMode: Gmail is open at 2017-10-09 04:36:28.07
D/GestureLauncher: onTestMode: Gmail Open Time: 3.42s

```

Figure 4.2: Agile Gesture records the task completion time in the background. The two example trials show access times by the PGS and GUI, respectively.

### 4.2.3 Apparatus

The device used for testing the app access performance via PGS was a Nexus 5 designed by Google (2.3 GHz processor with 2 GB of RAM). Agile Gesture was installed on this device for data collection. Figure 4.3 shows the phone device used in the experiment.

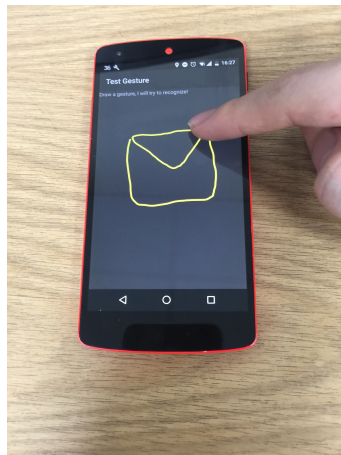


Figure 4.3: Nexus 5 used in the experiment for testing PGS.

The devices used for testing the app access performance via GUI were the participants' own smartphones. These were popular Android phones, such as Nexus 5 and Samsung Galaxy 8, which ran on the Android 7.0 system released by Google. Agile Gesture was pre-installed on these phones for data collection.

### 4.2.4 Participants

A total of 36 participants (15 female, 21 male) took part in the experiment. All were university students aged between 18 to 36 years old (mean age = 26.3). They were all



daily users of smartphones, reporting 5 to 6 hours of usage per day ( $M = 4.5$ ,  $SD = 4.3$ ), with 50 to 131 third-party apps installed on their phone at the time of the experiment ( $M = 84$ ,  $SD = 21.7$ ). None of the participants had prior experience of Agile Gesture. Prior to taking part in the study, the participants were asked to provide a screen-shot of apps on their phones and the names of the apps they had used in the previous seven days.

#### 4.2.5 App sample set

For the experiment, the participants were asked to access their installed apps via both PGS and GUI. Ideally, all participants should have used the same set of apps. However, we noticed that participants tended to have different apps on their smartphones. In fact, their installed apps were so different that we could not find more than three shared infrequently used apps on their phones. The number of apps on each participant's phone and the number of frequently & infrequently used apps also varied greatly. As it is unrealistic to find a number of participants who have exactly the same apps installed, our experiment used a different number of apps as the sample set for each participant.

We asked the participants to report their installed app list via email to the researchers before inviting them to participate in the experiment. We found that they all had installed more than 50 apps, and six of them had installed more than 100 apps. This is consistent with a mobile usage data analytical report (Kearl 2016). Ideally, participants would define and then use their PGS to access every app they installed. In that case, the participants would have a different number of app access tasks to conduct. To narrow this difference down, we divided the participants into six groups, based on the number of apps installed on their smartphones. The range of the number of apps in each group is shown in Table 4.1. There were six participants in each of these groups. In Chapter 3 we defined the target apps which participants used at least once a week as frequently used apps. This thesis used on this definition. That is, the app which a user used at least once a week was a frequently used app, otherwise, it was an infrequently used app. The reason for using this definition can be found in Section 4.2.6.

As the Table 4.1 shows, the participants with the smallest number of apps installed (i.e., 50-59) were assigned 5 frequent and 5 infrequent app samples in their set, whereas participants with the largest number of apps installed (i.e., >100) were assigned 10 fre-



















Table 4.1: App's amount allocation in the experiment.

Number of installed apps	50-59	60-69	70-79	80-89	90-99	> 100
Assigned <i>frequently</i> used apps	5	6	7	8	9	10
Assigned <i>infrequently</i> used apps	5	6	7	8	9	10

quent and 10 infrequent test samples in their set. Table 4.2 shows an example of an app sample set with 9 frequently used apps and 9 infrequently used apps being randomly assigned.

Table 4.2: An example of an app sample set used in the experiment.

Participant id :1

<i>Frequently used app</i>			<i>Infrequently used app</i>		
App Name	App icon	Personalized Gesture	App Name	App icon	Personalized Gesture
Photos		□	ZhiHu		Z
ingress			Line		L
Wechat		W	Google+		+
CameraFV5		○	Facebook		f
Shadow		∞	Flickr		oo
Translate		T	PayPay		P
Fit		♡	TrueSkate		S
Chrome		C	BaiDuMap		du
Realracing		Y	MeiYou		M

#### 4.2.6 Experimental design

The advantage of PGS may depend on how frequently an app is used (Nacenta et al. 2013). To examine how the frequencies of app usage interact with our approach, we included it as a variable in our experimental design. This thesis aims to find a method to classify the installed apps into two groups based on the usage frequency. Ideally, each group had an equal number of apps. According to a report, 49% of smartphone apps are accessed at least once a week (eMarket 2015). This means that about half of the installed apps are not used in the course of a week. This thesis regards the apps that have been

used within seven days as frequently used apps, whereas the rest are infrequently used apps. Therefore, the number of apps in the frequently and infrequently used groups is roughly the same. This definition is consistent with Chapter 3.

Another key variable in this study was the retention interval between the initial gesture shortcut creation and the gesture recall test. The gesture recall accuracy should be modulated by the length of retention time since the initial creation. According to Thalheimer (2010), people usually forget 77% of what they have learned after seven days. About a half of the installed apps are not accessed by the users within this period. This means that users may forget most gestures they have personalized for the infrequently used apps after 7 days. The difficulty in recalling these apps also means that the user has to make greater efforts to retrieve the retained gesture memory, which would result in a longer search time. To assess whether our approach is better than GUI under such different retention periods, we tested gesture recall accuracy for both frequently and infrequently used apps after 24 hours and 7 days following the initial creation.

The experiment was thus a  $2 \times 2 \times 2$  within-subjects design. The three independent variables were Access Method (*PGS* vs. *GUI*), App Usage Frequency (*Frequent* vs. *Infrequent*), and Retention Interval (*24 hours* vs. *7-days*).

The two dependent variables were the participant's speed to access the apps, and the recall accuracy of the gestures that the participant had initially created. Each successful access to the target app via Agile Gesture was counted as an accurate recall. The recall accuracy was calculated as a proportion of the successful recall trials out of all the test trials with Agile Gesture. This accuracy measurement was not relevant to *GUI* because, unlike *PGS*, users were not required to remember any new information. Thus, the three-way variable design was used for accessing speed only. For the accuracy results of Agile Gesture, we employed a  $2$  (App Usage Frequency)  $\times$   $2$  (Retention Interval) design.

In addition to the quantitative measurement, we also conducted a qualitative analysis of the post-task interview, where participants gave feedback on *PGS*.

#### 4.2.7 Procedure

Figure 4.4 illustrates the experimental procedure. The experiment consisted of four sessions over a period of seven days. The detailed procedures on each day were as follows:

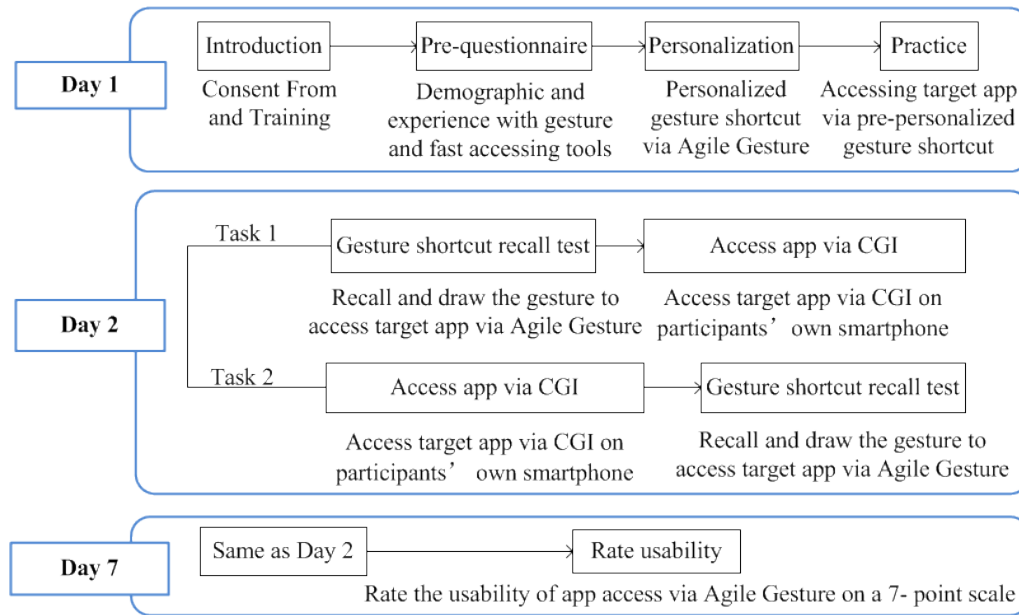


Figure 4.4: Experiment procedure. The order of the two tasks (Task 1 and Task 2) in Day 2 and Day 7 was counterbalanced.

Table 4.3: Pre-experiment Questionnaire

Questions	Descriptive results			
	M	SD	MAX	MIN
How many installed apps do you have on your smartphone?	82.42	28.55	134	50
How many apps do you use in recent seven days?	7.5	1.7	10	5
How long since you own a smartphone?	6.01	1.36	9	4
How many hours do you spend on your smartphone every day?	4.03	1.6	8	.67
Have you used gesture-based tools on your smartphone before?	All participants have used.			
Have you used tools for improving the app access performance before?	Two participants have no related experience before.			
How do you think about the existing app finding assistant tools?	Majority think should be improved.			
Give the name of the app you used in recent sever days.	Recorded by experimenter.			

**Day 1.** Participants first gave informed consent. They then completed a pre-questionnaire that collected their demographic data, and their experience with gesture and fast app accessing tools (see Table 4.3). The experiment started with a training session, where the researcher introduced the participants to how to use Agile Gesture. The participants then familiarized themselves with Agile Gesture. After the training, the participants completed the following personalization and practice sessions:

- i *Personalization.* The researcher presented a participant with apps in the sample set

with their icons and names printed on a sheet of paper. The apps were presented one at a time in random order. The participant was instructed to create a gesture shortcut for the app using Agile Gesture. The participant was allowed to try out different gesture shortcuts for the app until satisfied. There was no time limit for this process. This was repeated until gesture shortcuts for all the apps in the sample set were personalized. The personalized gesture shortcuts were then recorded by Agile Gesture. Table 4.2 shows some examples.

- ii *Practice.* Immediately after personalizing the gestures for the apps in the sample set, the participant was given the apps again and was told to practice using personalized shortcuts to access the target apps. If the participant failed to launch a target app due to an incorrect gesture, the correct gesture would be shown as a cue. Once participants were confident that they had remembered all the gesture shortcuts, they confirmed this orally, and the practice session ended. No time limit was given for this practice, but every participant completed it within 3 to 5 minutes.

**Day 2 and Day 7.** The tests on Day 2 and Day 7 followed the same procedure. Participants were tested for their memory of the personalized gesture shortcuts they had created on Day 1 and for their speed of accessing these targets using Agile Gesture or GUI. The order of the two methods for *frequently* and *infrequently* used apps was counterbalanced by a Latin square.

- i *Procedure for the gesture shortcut recall test.* Before the test, Agile Gesture was launched on the testing device (Nexus 5) and was made ready for recording the gestures and the task completion time. The researcher presented a participant with apps in the same format as had been used on *Day 1*. The order of the apps was randomized. The participant was told to reproduce the previously personalized gesture shortcut on Agile Gesture to access each app in the sample set as quickly and as accurately as possible. To do this, the participant was instructed to press the Test button on Agile Gesture to start drawing the gesture. At the same time, whether the recall was correct or not, the task completion time was recorded by Agile Gesture.
- ii *Procedure for the test using GUI.* This was done using participants' own smart-

phone in which Agile Gesture was pre-installed and ready to record. A participant was shown the same apps as in the gesture shortcut recall test described above. However, instead of using a gesture, the participant was told to locate each app's icon and access it using GUI as quickly as possible. Before searching for an app, the participant was instructed to press the "Home" button so that (1) the search always started from the home screen; (2) the timer of Agile Gesture for GUI started to record.

After the participants finished all the tests on *Day 7*, they were asked to rate the perceived effectiveness of app access via Agile Gesture on a 7- point scale questionnaire (see Table 4.4). The statements they rated included: (1) The likelihood of using a tool like Agile Gesture for their daily use; (2) Satisfaction with time to complete the app accessing task; (3) The possibilities of using PGS to access frequently used and infrequently used items separately.

Table 4.4: 7-point Likert scale for post-experiment rating

---

<b>Q1</b>	How likely would you to use a tool like Agile Gesture for your daily app accessing?
<b>R1</b>	<input type="radio"/> Very unlikely <input type="radio"/> Unlikely <input type="radio"/> Somewhat unlikely <input type="radio"/> Undecided <input type="radio"/> Somewhat likely <input type="radio"/> Likely <input type="radio"/> Very likely
<b>Q2</b>	Please rate your satisfaction with the speed of Agile Gesture for accessing app?
<b>Q3</b>	How likely would you to use Agile Gesture for your frequently used app accessing?
<b>R1</b>	<input type="radio"/> Very unlikely <input type="radio"/> Unlikely <input type="radio"/> Somewhat unlikely <input type="radio"/> Undecided <input type="radio"/> Somewhat likely <input type="radio"/> Likely <input type="radio"/> Very likely
<b>Q4</b>	How likely would you to use Agile Gesture for your infrequently used app accessing?
<b>R1</b>	<input type="radio"/> Very unlikely <input type="radio"/> Unlikely <input type="radio"/> Somewhat unlikely <input type="radio"/> Undecided <input type="radio"/> Somewhat likely <input type="radio"/> Likely <input type="radio"/> Very likely

---

Note that, according to the procedure proposed above, the testing device (Nexus 5) was used for the recall test, and participants' own smartphone was used for testing the accessing speed via the GUI method. The 36 smartphones employed in the experiment were all Android devices which ran on the Android 7.0 or above system. The testing device for the PGS method was the lowest version and had the lowest performance of Android devices compared with the devices used for testing the GUI method. In this case, the GUI method achieved the potential of a speed advantage since better performing phones were used. Based on this setting, the results did not lean towards our proposed methods.

There are two main reasons for using different devices for testing different methods (i.e., PGS vs. GUI). First, to make sure the participants of the PGS test did not practice

the gesture for recall after they left the lab, we kept the gesture library they had created on the testing device by not allowing it to be carried away. Secondly, the participants had their own ways of arranging their app icons in their smartphones. If using a device different from their own, they may have had difficulties finding the target app due to them not knowing where the icon was located. To best mimic how the GUI was being used in real life and to eliminate the time delay caused by the unfamiliarity with the apps' icon arrangement, this experiment allowed participants to use their personal phones for testing the GUI method.

## 4.3 Results

### 4.3.1 App access speed

Table 4.5 shows the results of the mean app access time in each condition. A three-way repeated-measures analysis of variance (ANOVA) found an effect with the Access Method, where apps took less time to access when PGS was used than when GUI was used,  $F(1, 34) = 79.07, p < .001$ . There was also a significant effect with the App Usage Frequency, where *frequently used apps* were accessed more quickly than *infrequently used apps*,  $F(1, 34) = 76.29, p < .001$ . The effect of the Retention Interval was not significant,  $F(1, 34) = 0.99, p = .327$ . The two-way interaction between Access Method and App Usage Frequency was significant,  $F(1, 34) = 25.03, p < .001$ . All other interaction effects were not significant. These included the two-way interactions between App Usage Frequency and Retention Interval,  $F(1, 34) = .23, p = .630$ , between Access Method and Retention Interval,  $F(1, 34) = 0.11, p = .750$ , and the three-way interaction,  $F(1, 34) = 2.47, p = .125$ .

The significant interaction between Access Method and App Usage Frequency is illustrated in Figure 4.5. To analyze the source of the interaction, we conducted paired t-tests. The results showed that a significant difference was found between the access speeds of the two methods for *infrequently used apps*,  $t(35) = 2.65, p = .003$ , but not for *frequently used apps*,  $t(35) = 4.13, p = .130$ .

Table 4.5: Mean task completion time (s) as a function of *Access Method*, *App Usage Frequency*, *Retention Interval*. Values in parentheses represent the standard deviation.

Access Method	<i>Frequently used app</i>		<i>Infrequently used app</i>	
	<i>24 hours</i>	<i>7 days</i>	<i>24 hours</i>	<i>7 days</i>
<i>Agile Gesture</i>	2.22 (1.10)	2.58 (1.05)	2.94 (1.27)	2.90 (1.59)
<i>GUI</i>	2.54 (1.03)	2.50 (1.06)	4.71 (1.86)	5.08 (3.20)

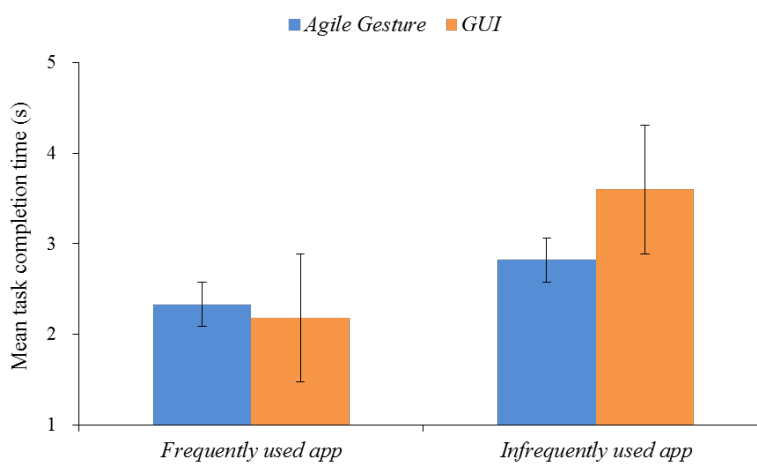


Figure 4.5: Mean task completion time for accessing *frequently* and *infrequently* used app via *Agile Gesture* and *GUI*. The error bars stand for one standard error about the mean.

### 4.3.2 Relationship between access speed and the number of installed apps

According to the pre-experiment questionnaire (see details in Table 4.3), the number of installed apps on the participants' smartphones ranged from 50 to 131. To find out how this variation affected performance, we plotted app access speed against the number of installed apps in Figure 4.6. The figure shows the speed of the two methods for each participant. It can be seen that the clearest speed advantage of our PGS was when the number of installed apps was large.

This was confirmed by two linear regression analyses. The results showed that the time participants spent on accessing the target app via Agile Gesture had no correlation with the number of installed apps ( $p=.670$ ). However, a significant regression equation was found,  $F(1, 33) = 221.15, p < .001$ , with an  $R^2$  of 0.87. The search time via GUI increased by 0.12 seconds for each additional installed app.



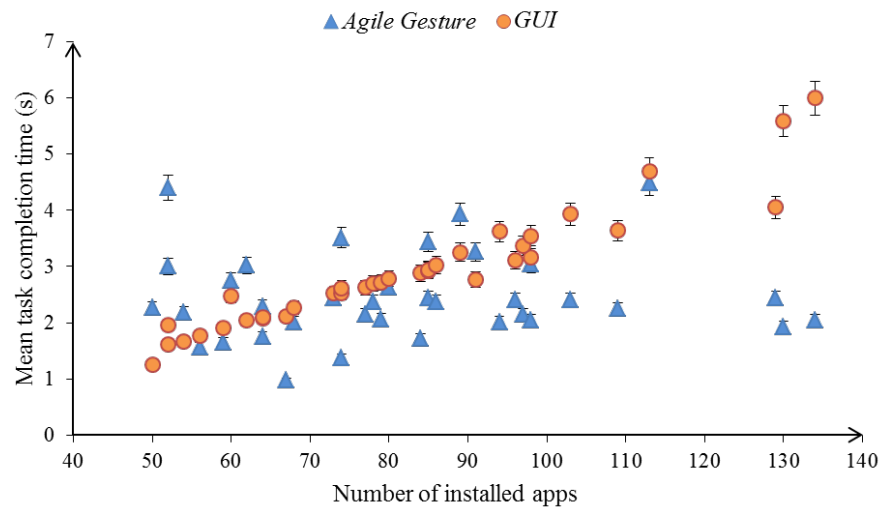


Figure 4.6: Task completion time by *Agile Gesture* and *GUI* as a function of number of apps on participants' phones.

### 4.3.3 Gesture recall accuracy

The mean proportion recall accuracy is shown in Table 4.6. A two-way repeated-measures ANOVA found a significant main effect with App Usage Frequency, where participants produced better recall accuracy for *frequently used apps* ( $M = .86$ ,  $SD = .19$ ) than for *infrequently used apps* ( $M = .73$ ,  $SD = .23$ ),  $F(1, 34) = 14.70$ ,  $p < .001$ . Their recall accuracy after *24 hours* of Retention Interval ( $M = .88$ ,  $SD = .14$ ) was better than after *7 days* ( $M = .70$ ,  $SD = 0.25$ ),  $F(1, 34) = 6.49$ ,  $p = .015$ . The interaction between the two variables was not significant,  $F(1, 34) = .07$ ,  $p = .799$ . This absence of interaction shows that the *infrequently used apps* did not suffer from the extended retention period than the *frequently used ones*.

Table 4.6: Mean proportion recall accuracy as a function of *App Usage Frequency* and *Retention Interval*. Values in parentheses represent the standard deviation.

Retention Interval	<i>Frequently used app</i>	<i>Infrequent used app</i>
After 24 hours	.80	.72
	(.24)	(.22)
After 7 days	.70	.61
	(.26)	(.24)

#### 4.3.4 Sample set size and recall accuracy

We also analyzed how the size of the sample set affected the accuracy of gesture recall. The distribution of average gesture shortcut recall accuracy over different sizes of sample sets is shown in Figure 4.7. It can be seen that the recall performance decreased when the size of the sample set increased. A linear regression was used to predict the gesture recall accuracy based on the size of the participants' app sample set. There was a significant regression equation,  $F(1, 4) = 166.99, p < .001$  with an  $R^2$  of 0.97. The average gesture recall accuracy was 0.8.

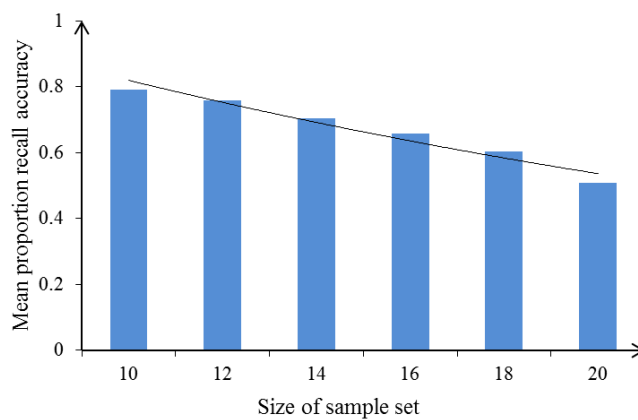


Figure 4.7: Mean proportion recall accuracy as a function of test sample set size.

#### 4.3.5 Unsuccessfully recalled gestures
















There were a total of 340 unsuccessful recall out of 1080 trials. We classified these into five categories. The results are summarized in Table 4.7, with examples for each:

1. Most failed access (133 cases) was due to using a wrong letter or a wrong case of a letter.
2. The next common mistake (85 cases) was confusion between gestures personalized as letters or as shapes.
3. The third common mistake (72 cases) was recalling a wrong abstract shape. A personalized shortcut is classified as "abstract" when the gesture neither resembles the icon nor the name. According to our data, only 18 abstract gestures of the total

of 90 were recalled correctly. Even though the users may consider gestures of this type to be simple and easy, they turned out to be very hard to remember.

4. A less common mistake (42 cases) was due to using a wrong drawing shape (see Table 4.4). For example, a "car" shape gesture was defined as a shortcut for Google map, but then tried to use a "location mark" gesture to access it.
5. Other kinds of mistakes (8 cases).

Table 4.7: Summary of common recall errors and examples in each category

Type of recall errors	App Name	App Icon	Personalized Gesture	User recalled Gestruer
Letter confusion	FaceBook		f	face
	Realracing		r	R
	Yelp		p	Y
Letter-drawing confusion	Trainline		↑	T
	Photos		□	P
	Fit		♥	f
Abstract shape confusion	Email		∟	—
	Messenger			└
	Instagram		△	∩
Drawing shape confusion	Justeat		▽	⊞
	Uber		◇	⊞
	Wangyimusic		♫	♫
Other confusion	GooglePlus		+	g+
	DropBox		∪	D
	Citynumber		⊞	C

### 4.3.6 Perceived effectiveness (Usability)

The participants' overall user experience rating of Agile Gesture at the end of the experiment was 5.67 on the 7-point scale. The results showed that 94% of the participants viewed Agile Gesture favorably and expressed willingness to use Agile Gesture in the

future (mean rating = 5.32). Moreover, 89% of participants were satisfied with the speed of the method for app access (mean rating = 5.87). The participants rated both accessing *frequently* and *infrequently* used items via PGS as possibly (on a 7-point scale, where 7= very likely). A paired t-test was used to find whether there was any rating difference between the two frequencies of items' usage. There was no significant difference over the participant's rating for using the PGS method for *frequently* (M= 5.09, SD=0.92) and *infrequently* (M= 5.55, SD=0.87) used apps;  $t(35) = 3.09, p = .120$ .

## 4.4 Discussion

This study aimed to improve the speed of app access on smartphones. We proposed a PGS approach for this purpose and conducted an experiment to compare it with GUI. We predicted a speed advantage of our method over GUI, particularly for *infrequently used apps*. The main findings of our experiment confirms this hypothesis. Here we summarize and discuss our key findings.

### 4.4.1 App access speed

In our study, the mean app access speed was 2.7 seconds with PGS and 3.8 seconds with GUI. The advantage was only found for apps that were not frequently used. This was predicted because it is common for GUI to position the frequently used apps on or near the home screen for easy and quick access. In contrast, the *infrequently used apps* usually require more extensive browsing through multiple screens. Launching *infrequently used apps* with PGS would not necessarily suffer from such setbacks.

Our results also show, rather unexpectedly, that the memory Retention Interval, whether the PGS method was tested after 24 hours or 7 days of the initial personalization, had little negative impact on the speed advantage. This means that the speed advantage of the PGS approach survives decaying memory trace.

Another important finding of this study was that the speed advantage of PGS was greater when participants had a larger number of apps on their phones. This suggests that the method can be particularly beneficial for users who install a large number of

apps. Our results show that, while the access time increased steadily as the number of installed apps increased with GUI, the access time slope remained relatively flat with PGS. The likely reason for this difference is that users of GUI had little idea of which screen the target app was placed. Furthermore, many users might have flipped back and forth to find the *infrequently used apps*' icon. According to the post-experiment questionnaire, this was given as one of the reasons why more participants would like to use Agile Gesture for accessing *infrequently used app* than *frequently used app* on their smartphones.

#### 4.4.2 Accuracy

Although PGS approach enjoyed a clear access speed advantage, it inevitably had to endure a cost of gesture recall accuracy. Not surprisingly, participants were unable to remember all the gesture shortcuts they had created initially. On average the proportion of recall accuracy was about .88 after 24 hours, and this dropped to .70 after a week. The proportion recall accuracy in our study was somewhat better than the previous report of .55 after 24 hours (Nacenta et al. 2013). The likely reason for this is that in our study, the participants recalled the shortcuts they had created themselves, but in Nacenta et al. (2013) the participants recalled the shortcuts predefined by the system. The difference could be due to the advantage of personalizing gestures. Perhaps actively engaging in the process of shortcut creation makes the created gesture more memorable.

The recall performance in our experiment also depended on how many gesture shortcuts needed to be created and retained in memory. The size of the app test sample set varied from 10 to 20 (including both frequently and infrequently used apps). The participants showed a higher recall accuracy when they were asked to create and remember a smaller set of gestures. In reality, it would be more likely for users to create a small set, perhaps even just one or two gesture shortcuts at a time. This could significantly boost the accuracy relative to that which was obtained in this study. Furthermore, the reason for creating a gesture shortcut is likely because users want to use it more often. Hence it is unlikely for them to be used after a week's delay. Repeated use within short intervals will likely consolidate the memory of the shortcut and hence improve the accuracy of the gesture recall. For these reasons, the assessment of PGS based on the impact of recall

accuracy in our experiment could have overstated the negative impact of memorability because the implementation of the approach in real-world settings may suffer less from the problem.

Finally, despite some failed trials to recall the correct gesture shortcuts, our post experiment questionnaire revealed that 34 out of 36 participants still expressed an interest to use Agile Gesture for app access.

## 4.5 Summary

This chapter has proposed a PGS approach for more efficient app access on touchscreen devices. The participants in our experiment themselves created gesture shortcuts for app access. After they created a gesture and associated it with a target app, we tested how quickly the participants were able to access the app after 24 hours or 7 days following their initial creation. The speed of accessing the target app using PGS was compared to GUI. Our results show that the average app access speed of our PGS was 2.7 seconds per app, which was 1.3 seconds faster than GUI. The benefit was mainly found in infrequently used apps, and the advantage was observed both after 24 hours and 7 days of the initial personalization. Our results further demonstrate that the speed advantage of PGS could be even stronger and more salient when a smartphone has a large number of installed apps.

The participants expressed a strong interest in using PGS for their future app searches, even though their memory for the gesture shortcuts was far from perfect. Our results show that the overall recall accuracy after 24 hours of gesture shortcut creation was 88%, which was further reduced to 70% after 7 days. These data should be understood in the context of the test conditions. The performance is expected to vary with the number of gesture shortcuts to be retained and the frequency of usage.

To overcome this limitation caused by human's memorisation, predicting the next app a user may want to access from the machine learning method is an alternative path. Prediction methods do not require the users to keep the predefined shortcuts in their mind, so this may overcome the memory limitations shown in this chapter. The details of this method are discussed in the next chapter.

# Chapter 5

## Intelligent Launcher: An Interface for App Search on Smartphones

### 5.1 Introduction

The Intelligent Launcher is another touch gesture shortcut interactive approach (different from Chapter 3 and 4). It can quickly locate an app based on a user's hand-drawn gestures, as shown in Figure 5.1. It does not require the user to keep the predefined set in

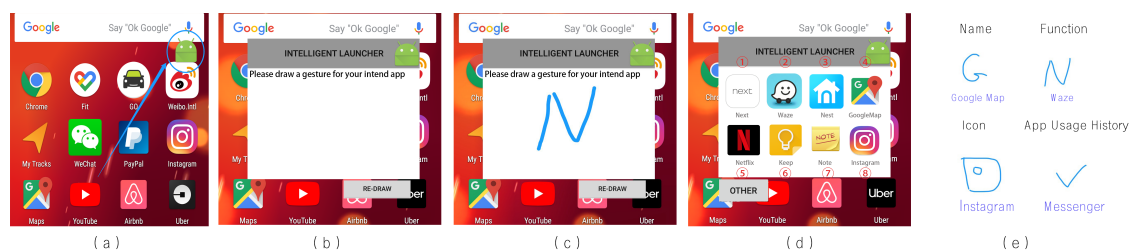


Figure 5.1: The Intelligent Launcher interface on Android. (a) The Intelligent Launcher is initially hidden with a floating icon (in the blue circle). (b) Pressing the icon opens the Launcher Screen. (c) When a user draws a gesture on the Launch Screen, (d) The Intelligent Launcher predicts the gesture's eight best matched target apps and presents them to the user. (e) The gesture is interpreted based on the app's name, function, icon and app usage history.

mind anymore.

Chapter 3 and 4 have already reported the advantages of using PGS access both frequent and infrequent used apps. However, the PGS method still requires the user to define the gesture shortcut first. Users can not benefit from PGS without the pre-defining process. Unlike the existing approaches (i.e., PGS), a method which would allow the users to access their intend app without pre-defining it would help to improve the interaction process. In the light of this, this chapter proposes the Intelligent Launcher. Whatever the user draws, the Intelligent Launcher uses that to help predict which target app the user wants to access.

To do this, the Intelligent Launcher predicts the target via three types of source (i.e., the app's icon, function and name library, and the app's usage history). First, the Launcher compares the visual similarity of the *Drawing-Type Gestures* to the app's icon rather than the predefined gesture templates. Second, for *Letter-Type Gestures*, it makes a prediction by developing and adding a library of the app's functionality to the existing name library in the smartphone. Last, for the *Abstract-Type Gestures*, it makes a prediction according to the app's usage history. The assumptions of the launcher were drawn from our empirical data, and our evaluation shows that it has achieved user satisfaction and 96% prediction accuracy.

## 5.2 Exploring Gesture Shortcut for App Access

Gesture Search (Li 2012) and Gesture On (Lu and Li 2015) guess the users' intended app from the *letter shape* gesture they draw on the screen. Gesture Marks extended the input modality by eliminating the limitation on gesture shape. It guesses the users' intended app by learning the gesture template (including *letter* and *non-letter* shape gestures) built by someone other than themselves. Rather than learn from the template, Gesture Magic (Play 2018) allows the user to access their intended app by recording the gesture-app matches the user has defined themselves in advance and then comparing the inputted gesture with them. However, these methods all rely on a single source (e.g., the Letter gesture template) for the prediction. In real life, people do not always use one type (shape) of gesture (e.g., *letter*, *drawing* or *abstract*) as a shortcut for app access. For



example, from our collected data, the *letter* "E" and the *drawing* shape "envelop" had both been used as the shortcut for accessing the email app. The prediction for the *letter* gesture relied on the handwriting recogniser, while the *drawing* gesture relied on the apps' icon template. Therefore, it is essential to find a way to guess which type (shape) of gesture the user is going to use. Since the different type (shape) of gesture could be interpreted from different sources.

As discussed earlier, we found that users tend to create their gesture shortcuts for accessing target apps differently due to the different references (sources) they rely on (Zhang 2016; Zhang et al. 2016; Zhang 2017). Therefore, this section conducted an exploratory study to find out why the users create the kinds of gestures for app access they do, and whether any patterns could be identified. We then used the findings to inform the design of the Intelligent Launcher.

### 5.2.1 Data Collection

We collected gesture shortcuts from the smartphone users for their installed apps. For this study, we recruited 36 users, 14 of whom were female, with a mean age of 24.2, and who had over 50 installed apps. They were asked to install a data collection tool on their smartphone. The tool randomly selected a set of targets from the installed apps, and the participants were asked to draw a gesture shortcut for accessing each of them. The target was prompted with its icon and name under it: see Figure 5.2. Participants were prompted with one target at a time. They were allowed to redraw their gesture by tapping the "redraw" button. After they gave the reason why, they pressed the "next" button and moved on to another target app. After each gesture was created, the participants were asked to explain why they had chosen that specific gesture. The size of the target app set ranged from 12 to 20; the participants who had more installed apps received a larger set size: e.g., 12 target apps were selected for participants who had 56 to 60 installed apps, but 13 target apps for those who had 65 to 70 installed apps. The target apps for the participants would be different because the selection came from personal smartphones. The selection of these targets approximated the actual usage model of apps for each user (Shin et al. 2012). Each participant was prompted with all the target apps in their set.

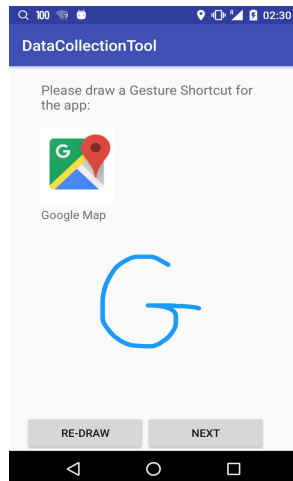


Figure 5.2: The data collection tool.

The gesture shortcuts drawn by participants were recorded by the data collection tool. No time limit was set for the data collection procedures, but it usually took 10 to 15 minutes per participant. These data were collected for further machine learning prediction.

### 5.2.2 Types of Gesture Shortcut and the Reasons for their Creation

Prior approaches such as Gesture Marks (Ouyang and Li 2012), classified gestures as either handwriting or non-handwriting. For example, the letter(s) like gestures (i.e. 26 alphabet letters and their combinations) are classified as handwriting gestures, otherwise (e.g., rectangle, heart shape, etc.) they are considered non-handwriting gestures. This classification was later subdivided by two methods: (1) a study focused on understanding the gesture shortcut on touch screen devices subdivided the non-handwriting gesture into, Abstract, Geometric and Icon gesture, the handwriting into Letter and Word gesture (Poppinga et al. 2014); (2) studies focused on using gesture shortcuts for app access named handwriting gestures as Letter gestures and subdivided non-handwriting gestures into *Drawing* and *Abstract* gesture (Zhang et al. 2016; Zhang 2016 2017). Following the latter, in this paper, we have labelled the gestures as either a *Letter*, *Drawing* or *Abstract* gesture, as seen in Figure 5.3 based on two aspects. One, some non-handwriting gestures such as a horizontal line for the most frequently used app are created for particular reasons (e.g., easy to draw) which is different from other non-handwriting gestures (i.e., created based on the app's icon). Two, for the purpose of the implementation of Intel-

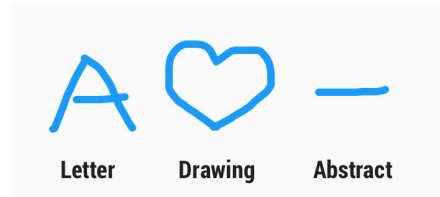


Figure 5.3: Gesture examples for *Letter*, *Drawing* and *Abstract* types.





















Intelligent Launcher, the *Abstract*-Type gesture needs separate recogniser from other non-handwriting gestures. This is because the *Abstract*-Type gesture cannot be identified based on the app's icon.

Based on the users' explanations of why they chose a certain type of gesture for an app, we classified their given reasons into five groups: see Table 5.1. First, for those created from an app's name, e.g., "f" for "Facebook", "gm" for "Google Map", we classified them as the *name* reason creations. Second, for gestures that were either created from the app's icon or similar to the icon e.g., a rectangular shape for "Instagram" to imitate the camera-like icon or a triangle shape to imitate the "YouTube" icon, we classified as the *icon* reason creations. Third, for gestures created from an app's function e.g., a cloud shaped gesture for the "BBC Weather" app, or a letter "P" gesture for the "FaceU" app because of its photo function, we classified as the *function* reason creations. Fourth, for some easy-to-draw gestures e.g., a check mark shape for fast access to frequently used apps we classified as the *time-saving* creations. Finally, for the 6 gestures for which users could not give a specific reason e.g., a poly-line we marked them as *unspecified* reason creations.

### 5.2.3 Results and Analyses

A total of 540 gesture shortcuts and the reasons related to their creation were collected from 36 participants. This section is aimed at finding out how the Intelligent Launcher can better predict a user's intended app. An analysis of these data revealed four main findings. This section will present the findings and demonstrate how to improve the Intelligent Launcher based on them.

Table 5.1: Sample gestures from five creation reasons.

Reason	Gesture1	App1	Gesture2	App2
<i>name</i>		 Facebook		 GoogleMap
<i>icon</i>		 Instagram		 YouTube
<i>function</i>		 Whether		 FaceU
<i>time-saving</i>		 Outlook		 WhatsApp
<i>unspecified</i>		 PayPal		 Skype

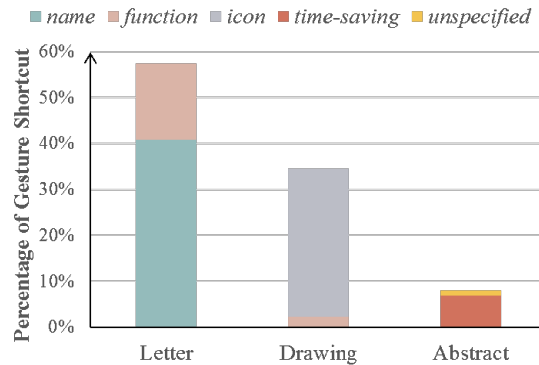


Figure 5.4: The percentages of three types of gestures and their creation reasons.

### Percentage of gesture types in participants' responses

We analysed the frequency of each type of gesture participants tended to use. As shown in Figure 5.4, 57% of all gestures were *Letter* gestures. Of the *Letter* gestures, 71% were created from the app's *name*, and the remaining 29% were created from the app's *function*. One possible reason for this percentage use of name gestures is the usage of the keyword search method of the iPhone (Apple 2018) and Android phone (Google 2015a), which can encourage the use of app names. Slightly over one-third (35%) of all gestures were *Drawing* gestures. Nearly all (95%) of the *Drawing* gestures were created from the app's *icon*. A small portion of them (5%) was created from the app's *function*. *Abstract* gestures account for 8% of all gestures. Of these, 86% were created for *time-saving* access to frequently used apps. The rest were created for *unspecified* reasons.

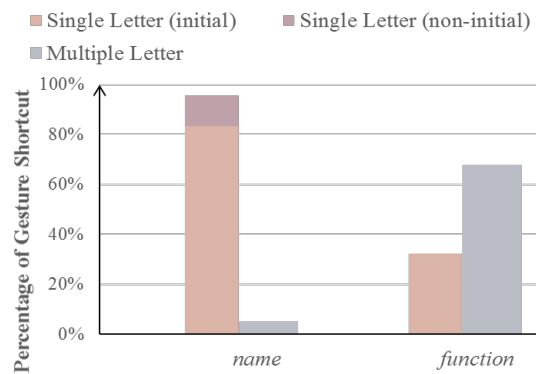


Figure 5.5: The percentage of *Single Letter* (initial), *Single Letter* (non-initial) and *Multiple Letter* gestures created from an app's *name* and *function*.

### Composition of *Letter* gesture shortcuts









Next we analysed the composition of the *Letter* gestures. Apart from knowing these gestures were created from the app's *name* or *function*, we identified the ones that consisted of one character, e.g., "f", as a *Single Letter* gesture, and the rest, e.g., "gm", as *Multiple Letter* gesture. The handwriting recognisers also addressed the *Single Letter* (Bahlmann et al. 2002; Connell and Jain 2001) and *Multiple Letter* (Hu et al. 2000; Plamondon and Srihari 2000) gestures differently since the latter requires segmentation. For this reason, we analysed the percentage of *name* and *function* within *Single-* or *Multiple-Letter* gesture types.

As shown in Figure 5.5, most gestures are of *Single Letter* type (87%) rather than of the *Multiple Letter* type (13%). It may be because drawing a *Single Letter* gesture takes less effort and time (Rekik et al. 2014; Vatavu et al. 2011). Nearly all (95%) gestures created from an app's *name* contained a *Single Letter*. The majority of (83%) *Single Letter* gestures used the first letter, exhibiting the typical "first-letter advantage" (Grainger et al. 2016; Scaltritti and Balota 2013). The results also show that the 5% of *Multiple Letter* gestures consisted of the initials of the app's *names*, e.g., "fb" for "Facebook", "wc" for "WeChat".

The percentage of *Multiple Letter* gestures (68%) created from the app's *function* was higher than with *Single Letter* Gestures (32%). This may be because of the difficulty to represent the function of an app with a *Single Letter*. For example, "job" would be easier than "j" to represent the function of "LinkedIn".

To investigate how to match these *function* gestures to their respective apps, we ex-

Table 5.2: Sample of *function* gestures that were successfully mapped to the target app and those unsuccessfully (uns.) mapped.

Sample App	Successful map			Uns. map
Gesture				
Icon				
Name	Waze	Rightmove	WeSing	Weather

explored their relationships. Previous studies have developed specific functional categories for their app sets manually (Sahami Shirazi et al. 2014; Poppinga et al. 2014). However, these categories cannot cover our set. We assigned the apps in our set to a particular functional category followed the "Google Play Category" (Google 2018d). All apps in the Google Play market belong to a predefined category, where the category name shows the app's main function. For example, "Google Map" is in the "Maps & Navigation" category. We found that 95% of the *function* gestures created by our participants could be found in the target app's category name. For example, the gesture "v" for "YouTube" is in the "Video Players" category and the gesture "job" for "LinkedIn" is under the "Job Search" category. Only 5% of the *function* gestures could not be found in a target's category name suggested by Google.

### The similarity of the *Drawing* gesture with the target app's icon

Since most *Drawing* gestures were created because of the icon's appearance, we employed an appearance-based distance metric (Ouyang and Davis 2009) to estimate how often the gestures were visually similar to the target's icons. We found that more than 95% of the *Drawing* gestures appeared visually similar. This included all gestures created from the app's *icon*. Also, rather unexpectedly, over one-third of the *Drawing* gestures were created from the app's *function*. This may be because the way of expressing the app's function is shared between the user and the icon designer, as illustrated by the examples given in Table 5.2.

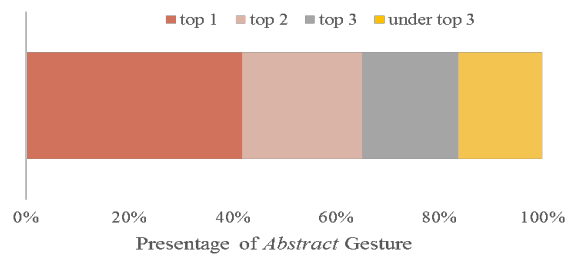


Figure 5.6: The percentage of *Abstract* gestures for the top 3 and below the top 3 most frequently used apps.

### ***Abstract* gesture and target app's usage frequency**

We investigated the frequency of target apps that were opened by *Abstract* gestures. The app usage frequency was found on a data collection tool (Zhang 2017) that keeps an app access record of the user. We used 7 days of a participant's app access log after they had taken part in the experiment to calculate the access frequency for each app. We found that when the top three frequently used apps were combined, (see Figure 5.6), they accounted for 84% of all *Abstract* gestures.

#### **5.2.4 Creating an AI Gesture Shortcut Interface.**

The results presented in Figures 5.4-5.6 show the potential predictive power of these findings. Based on the findings, we have developed the following guidelines for creating an interface that makes intelligent guesses about intended apps from users' gesture input.

**It should be able to predict the target of a *Single* or *Multiple Letter* gesture from an app's *name* or *function***

The results in Figure 5.5 show the power of the name and function for predicting a *Single* and *Multiple Letter* is different. *Single Letter* gestures should match the target with the same letter of the initial of the app's *name* or *function*. The weight of mapping it to the *name* should be greater than to the *function*. In contrast, most *Multiple Letter* gestures should be predicted based on the target's *function*. The function of the app should be extracted from the Google Play app category's name. We implemented the Letter Gesture

Recogniser based on these principles.

### **It should be able to use an apps' *icon* to predict a *Drawing* gesture**

The result for the *Drawing* gesture shows that visual similarity to the target's *icon* is a remarkably informative metric for predicting the target app, even for gestures created from the app's *function*. This implies that even before a user defines any gesture shortcuts, there is the possibility of interpreting the *Drawing* gesture by looking at their installed app's icon. This is especially useful since the icons are specific to a given user.

### **It should be able to predict the target for an *Abstract* gesture based on the app usage history**

Even though *Abstract* gestures lack semantics (Poppinga et al. 2014) and are hard to remember (Zhang et al. 2016), they are still predictable. As results show in Figure 5.6, *Abstract* gestures are predictable because they tend to be mapped to the top three most frequently used apps. The order of possible mapping is the same as that of the target's usage frequency.

## **5.3 Intelligent Launcher**

Based on the findings and guidelines achieved from the exploratory study, we have designed and developed the Intelligent Launcher, a user interface that can predict and launch the intended target app from an unconstrained gesture drawn by users.

### **5.3.1 Interaction with Intelligent Launcher**

The Intelligent Launcher is implemented as an initially hidden float icon on the screens of the Android phone. The icon is an Android logo, as shown at the top right of Figure 5.1 (a). Users can open the Launch Screen (the top part of the screen with the white background, shown in Figure 5.1(b)) by pressing the floating icon. When the user draws



a gesture on the Launch Screen (see Figure 5.1 (c)), the launcher interprets it based on three sources:

- 1) **LGR**: the *Letter Gesture* Recogniser
- 2) **DGR**: a library of *Drawing* gestures- target apps' icons
- 3) **AGR**: the app *usage history* of frequency

It then displays the best 8 matches in a  $4 \times 2$  Grid view (see an example in Figure 5.1 (d)). The recommended apps are displayed in the decreasing order of the best matches. As the red number shows, shown in the Figure 5.1 (d), the users' intended apps recommended by the Intelligent Launcher are displayed in the order of first to second row, left to right. That is, the top match is given at the top left of the Launch Screen, the last match is given at the bottom right of the Launch Screen. Note that the Intelligent Launcher always gives 8 of the best matches out of all the installed apps. Normal smartphone users always have more than 8 installed apps. Therefore, it is certain there are more than 8 apps to recommend (even though, sometimes, the matches are less likely accessed by the user). The Intelligent Launcher provides the 8 best matches, which are followed by a Gesture Mark (Ouyang and Li 2012). The users then choose the target from the list. If the target is not shown, they can press the "other" button. A list of all the installed apps pops up for the users to choose their target app from.

### 5.3.2 Architecture of the Intelligent Launcher

The architecture of the Intelligent Launcher is illustrated in Figure 5.7. When the user draws a gesture, the "Predictor" analyses it and provides the 8 best matches to the user. Then, the user needs to choose their target app by touch its icon on the recommended list. The the app then launches for the user. Meanwhile, the gesture-target app pair will update the "Recognizer" for improving further recommendations. The "App access history" (app usage information) is recorded in the "Log" and feed for *Abstract* gesture recognition.

The analysis is processed by a *Letter*, a *Drawing* and an *Abstract* Gesture Recogniser (LGR, FGR and AGR in the Figure). As shown in the Figure, "LGR", "DGR" and "AGR" rely on the "Name Library" and the "Function Library", the "Icon Template" and the "Log" respectively.

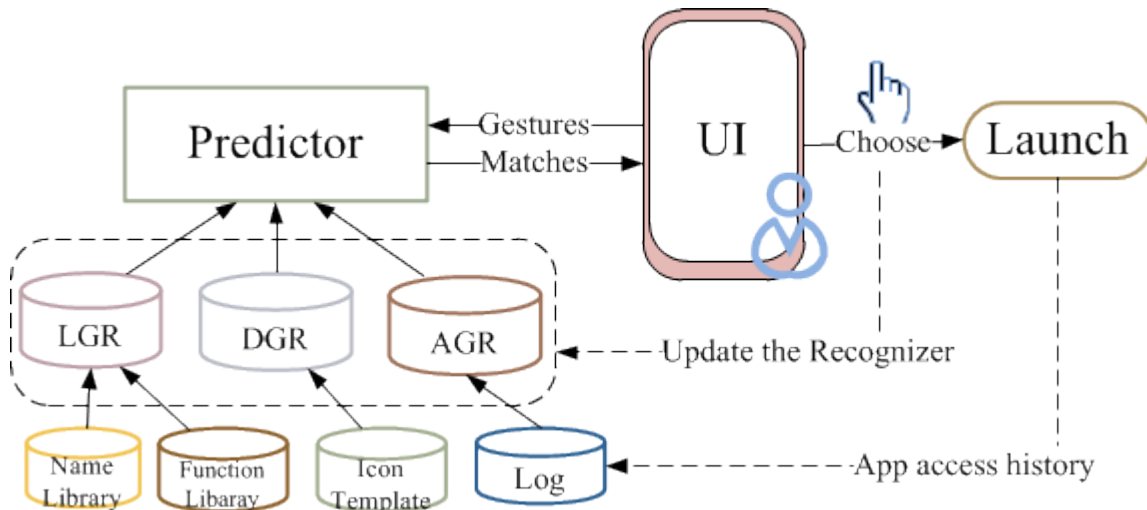


Figure 5.7: The architecture of Intelligent Launcher.

### 5.3.3 Algorithm of Intelligent Launcher

This section introduces how the Machine Learning (ML) method works in the Intelligent Launcher. First, we discuss how the whole recommendation (prediction) is updated based on a Hidden Markov Model (HMM). Then we provide the details of why and how the selected machine learning methods are engaged for "LGR", "DGR" and "AGR" separately by reviewing the literature.

#### HMM for Dynamic Update

Based on observing the data collected in Section 5.2, we found the app accessing history is important for predicting the target app. For example, a user who has already used the "gm" gesture for accessing "Google Map" is more likely to use the same gesture (i.e., "gm") again rather than any other gesture for accessing the same app (i.e., "Google Map"). Therefore, an algorithm that can dynamically update itself based on the users' input is better for the prediction. That is why we decided to optimise our prediction by learning from the history of gesture-target pairing.

Based on the literature, the HMM can best meet our requirements for dynamic updating (Russell and Norvig 2016). The basic principle of HMM is to define a set of states, each corresponding to a region or specific site in the proteins being modelled (Sonnhammer et al. 1998). It predicts based on the time-sequencing (Yamato et al. 1992).

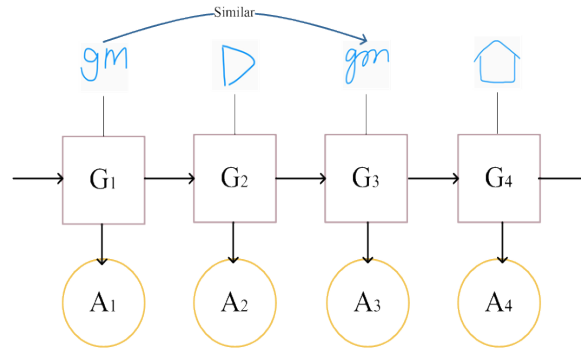


Figure 5.8: The Intelligent Launcher optimised the prediction based on a Hidden Markov Model (HMM).

Based on this basic principle, for a gesture that is visually similar to an old gesture, we updated the probability value of the target that matched the old gesture to 0.5. Note that the value is updated to 0.5 (i.e., 50%) here. The update is based on the results of our observation on the collected data.

In Figure 5.8,  $G_i$  represents the gesture input at time  $i$  and  $A_i$  represents the intended target app for this gesture. As  $G_3$  is visually similar to an old gesture  $G_1$ , the  $P(A_1 | G_3)$  is updated to 0.5. Otherwise, e.g.,  $P(A_2 | G_2)$ , there is no need to update.

The order of the apps appearing in the candidate list is determined by the order of their probability value ( $P(A_i | G_i)$ ).

### Letter Gesture Recogniser

The launcher employs a Letter Gesture Recogniser so that this type of gestures can be recognized and then matched to a possible target app. The launcher needs a letter gesture segmentation method to support both *Single* and *Multiple Letter* recognition. Then, the input gesture can be translated to a letter or a sequence of letters. The Intelligent Launcher can predict the app which the user intends to access, relying on the Name - App library by scanning the user's installed app list. Based on the results of the Letter Gesture Recognizer, the predictor calculates the probability value of this gesture for matching to a target app.

The handwriting recognition which supports segmentation is not our main contribution. We employed the recognition method based on what had been used successfully in previous studies for sketch recognition on pen-based gesture recognition (Ouyang and

Davis 2009). The performance of this recognition method is compatible with prior approaches (Li 2010a; Anthony and Wobbrock 2010; Wobbrock et al. 2007).

For a better understanding of why this handwriting recognition method was selected, we discuss the main special properties of our input (i.e., the finger-handwriting gestures). First, it is finger-based input with a limited drawing area. Second, the smartphone itself has fewer computation resources compared to a laptop or computer server. Third, the input tends to be an initial or abbreviation of the app's name/function instead of the complete app's name/function. The selected recognition method can well-address these properties since: (1) it is trained in the letter and words data collected on touchscreen smartphones, which is suitable for finger-based input recognition; (2) it can run in real time on a smartphone; (3) it does not require complementary word-level labels for training and can recognise natural multi-stroke and mixed cursive handwriting.

Therefore, we employed a support vector machine (SVM) with a linear kernel to recognize the input gestures, based on the list of properties listed above. The basic principle of the linear kernel SVM we used is discussed in (Boser et al. 1992). In brief, an SVM creates a line of output which separates the data into classes. It means, in our case, the SVM takes the user drawing gestures as input and separates them and then recognizes those gestures. More details about the data training and the feature selection of the SVM algorithm can be found in (Ouyang and Li 2012).

For  $N$  apps with the gesture letter in its name, the probability of matching to the target app  $P(T_i)$  is calculated from formula 5.1, and the weight  $w_j$  is calculated from formula 5.2. Where the *Single* initial letter gesture calculates  $P(T_i)$  use  $w_1$ , for *Single* non-initial letter use  $w_0$  and for *Multiple Letter* gestures use  $w_2$ . An example is shown in Table 5.3.

$$P(T_i) = P(T) \times w(j), \tag{5.1}$$

where  $i = 0, \dots, N, j \in \{0, 1, 2\}$ ,  $P(T) = \frac{1}{N}$ .

$$\sum_{i=1}^N P(T_i) = 1, \tag{5.2}$$

where  $w_0 : w_1 : w_2 = 83 : 12 : 5$

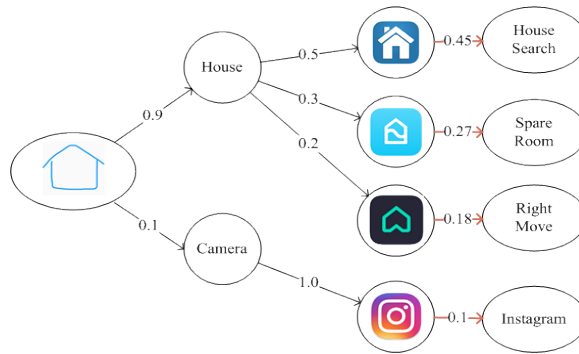


Figure 5.9: An example of how a "house" gesture matches the target app by learning its visual similarity to the icons in the template. The value on the black line is the visual similarity score: the value on the red line is the probability value of matching to that app.

The Function Library has been developed using a similar method, only extracting the letter from the Google Category name of the target app.

### The *Drawing* gesture template

The launcher infers the target app for a *Drawing* gesture by computing its visual similarity to the app's icon. To achieve this, a template with the user's installed app's icon and an open source tool (Abadi et al. 2016) are used to evaluate the visual similarity between the input gesture and each icon in the template. The possibility of a hit for matching the intended target app is determined by the order of the visual similarity score between the gesture and the target's icon. An example of how a gesture matches to a target app is given in Figure 5.9.

### The *Abstract* gesture predictor

The Intelligent launcher predicts the next app by combining the app access history information with the gesture produced by the user. Even in our exploratory study, only a small number of *Abstract* gestures were used by the participants, but the tendency to use it for the most frequently used apps was obvious. The launcher records the app access history in the back end (Log) and calculates the app usage frequency of each app for recent days. The Log beyond the previous seven days is automatically deleted to release more memory. The probability value of an *Abstract* gesture matching with a target app is based on the level of *usage frequency*.

### 5.3.4 Implementation

The launcher was implemented in Java using Android API 28 and is compatible with API under 28. It was tested on various Android-powered devices such as the Google Nexus 5, Samsung Galaxy S8 and HuaWei Honor 9.

## 5.4 Evaluations

To investigate how Intelligent Launcher performs for app access in the users' daily lives and how the different components contribute to the correct prediction, we collected 548 gestures for 197 target apps from 22 Android phone users.

The specific goals of the evaluation study were to learn:

- How often the Intelligent Launcher is able to correctly predict the target app based on a gesture.
- How various components, i.e., Name Library, Function Library, Icon Template, and app usage Log, contribute to correct predictions.
- The limitations of the Intelligent Launcher.
- How the users evaluate (think) the Intelligent Launcher: effectiveness and usability.

For these purposes, we analysed the participants' log for the period of their usage (the 7th to 14th day from the start). This is because during the first 7 days, the launcher recorded the app usage frequency to feed future predictions. Thus, from the 7th to 14th days, the launcher was able to use the collected data to predict and improve the target apps predictions.

### 5.4.1 Participants

We recruited the Android users in the University via email. Once the recipient had agreed to participate, we sent the "IntelligentLaunchertest.apk" by email, which allowed the participants to install the *testing version* of the Intelligent Launcher on their phones. For the purpose of data collection, this *testing version* recorded and sent the participant's app

usage frequency and launcher prediction information to our "log" server. The participants gave their informed consent for this procedure. We have access to the data on the "log" server.

From the logs of 30 participants, we selected and analysed 22 based on the criteria that they had used the Intelligent Launcher for app access at least once per day for at least 14 days. The reasons for choosing these criteria are: first, to make our results credible, we focused on the users who used our proposed method at least once a day; second, as discussed in the "Implementation" section, the launcher needed at least 7 days for pre-data collection before it could predict. To balance the prediction of frequently and infrequently used apps, we decided 14 days was the shortest usage cycle.

The selected participants included 7 females, and their ages ranged from 25 to 36 (mean = 28). They were all full-time students or researchers in a local university, majoring in business, animation and computer science.

### 5.4.2 Data Collection

Data collection followed a similar process to how the participants were recruited. The experimenter sent the *testing version* of the Intelligent Launcher to the participants via email and asked them to install it on their own smartphone. The participants were also asked to use the Intelligent Launcher in their daily lives for regular app access tasks. No upper limit for usage frequency was imposed. Fourteen days later, the participants were asked to answer an online survey about their user experience.

### 5.4.3 Results

#### Prediction Accuracy

The launcher was able to predict and list the correct target app within the top 8 matches for 96% of all gestures. The mean prediction accuracy of the Recogniser and Intelligent Launcher is shown in Figure 5.10. On average, 96% (M = 23.8, SD = 5.2, Min = 15, Max = 34) of gesture queries matched the target app correctly. The prediction accuracy of the *Letter*, *Drawing* and *Abstract* Gesture Recognisers was 98%, 92% and 70% ,

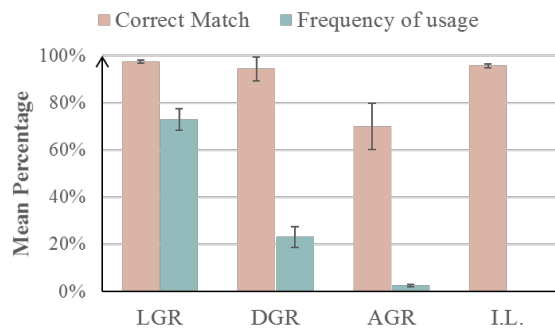


Figure 5.10: The mean percentage of (1) correct match by the three gesture recognisers and the Intelligent Launcher (I.L.) when considering the top 8 matches, and (2) the frequency of using the *Letter*, *Drawing*, *Abstract* Gesture Recogniser to predict the target. The error bars stand for one standard error about the mean.

respectively. Figure 5.10 also shows the mean percentage of gesture predicted by the *Letter*, *Drawing* and *Abstract* Gesture Recognisers. On average, participants used the Intelligent Launcher 24.9 times (SD = 5.3, Min = 15, Max = 34) during the 7 days period. Of these usages, 73% of them was predicted by the Letter Gesture Recogniser, 22% by *Drawing* Gesture Recogniser and only 5% by *Abstract* Gesture Recogniser. This shown a consistent trend with our exploring study, where, the Letter gesture was the most frequently used, *Drawing* gesture the second, and *Abstract*-Type gesture was the least used.

### Contributors to the correct predictions

As shown in Figure 5.10, a large number of gestures predictions relied on the Letter Gesture Recogniser. For correctly predicted target apps, we also recorded whether the correct match came from the Name Library or the Function Library, as well as recording the position of the target located in the candidate list. As shown in Figure 5.11, the majority- 60% (M = 14.1, SD = 5.33)- of correct predictions were contributed by the Name Library, and over one-fifth- 21% (M = 5.1, SD = 4.28)- were contributed by the Icon Template. The rest- 16% (M = 4.0, SD = 4.73)- were contributed by the Function Library, and only- 3% (M = .6, SD = .66)- by the Log.

For the total correct predictions for the Letter Gesture Recogniser, the Name Library contributed 80% and the Function Library contributed 20%. This means that being able to predict these *Letter* gesture representing the app's *function*, the Intelligent Launcher ex-



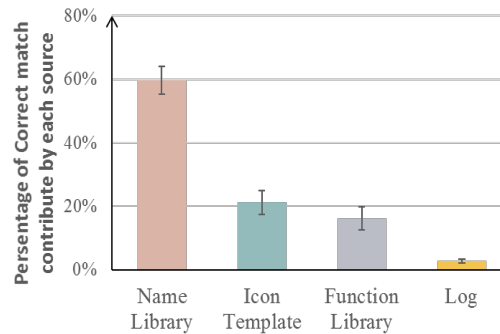


Figure 5.11: The mean percentage of the correct matches contributed by the Name and Function Library, Icon Template and the Log. The error bar stands for one standard error about the mean.

tended the *Letter* gesture prediction by one-fifth. A small number of the *Drawing* gestures that were also created from the target app's function were not taken into account here. The launcher processed these gestures by using the "icon" template, which provided 93% of the correct matches by the drawing gesture Recogniser. A 100 % of *Abstract* gestures' predictions relied on the "Log" information.

Figure 5.12 shows the average position of the correct target located in the candidate list. The Icon Template scored the best average position ( $M = 2.0$ ,  $SD = 0.9$ ) because *Drawing* gestures were closely similar to the target icons but quite different among each other. As a result the number of eligible candidate targets was relatively small ( $Min = 1$ ,  $Max = 4$ ), meaning participants could find their intended target on the top of the list. The Function Library scored the poorest average position ranking ( $M = 3.0$ ,  $SD = 1.2$ ), which was because the users had several apps within the same function category. For example, in our study, they had up to 6 apps within the same function category. The launcher assigns the same possibilities of matching function-based gestures to the same "category name", so that the intended target could be located in a relatively late position on the candidate list. A similar result was found for targets predicted by the Name Library ( $M = 2.6$ ,  $SD = 1.7$ ); this was due to a relatively high number of apps sharing the same letter. Within the *Letter* gestures, the *Multiple Letter* type ( $M = 2.0$ ,  $SD = 1.3$ ) enjoyed a position advantage compared to the *Single Letter* type ( $M = 3.0$ ,  $SD = 1.8$ );  $t(21) = 4.38$ ,  $p < .001$ , on a paired-samples t-test. This was because an additional letter helped to remove non-relevant apps.

Figure 5.13 shows the launcher's prediction accuracy when the number of candidate

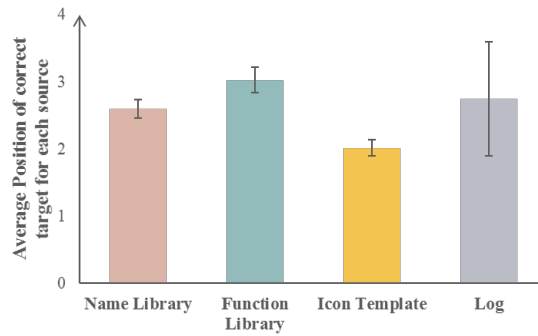


Figure 5.12: The average position of the correct target located in the candidate list. The error bar stands for one standard error about the mean.

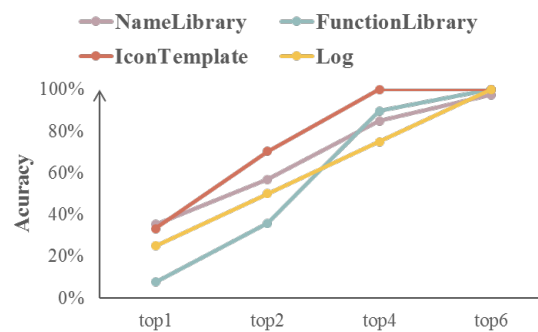


Figure 5.13: The Launcher's prediction accuracy as a function of number of target candidates.

targets is taken into consideration. These results can inform the prediction ability of the Intelligent Launcher when we consider alternative interfaces, which provide a smaller number of candidate target apps. All sources' accuracy significantly increased from the top 1 to top 2 and 4 matches, then slightly increased (from 70% to 100%) after extending the candidate list size to 6.

### User Feedback

All 22 participants answered the survey about their experiences after using the launcher. A 5-point Likert scale, with 1 for "strongly disagree" and 5 for "strongly agree" was used for the participants' to rate whether they:

- thought the Intelligent Launcher would be helpful for them for app access.
- thought the launcher was able to predict their intended target.
- would use Intelligent Launcher in the future for app access.

Most users agreed that the Intelligent Launcher was helpful (Median=4) and was able to predict the intended target (Median=4), and strongly agreed that they were satisfied with the Intelligent Launcher (Median=5) and would probably use the Intelligent Launcher in the future (Median=5). Note that we use the median not the mean value, as has been the case in other chapters in this thesis. Using the median instead of the mean value allows us to compare the user feedback when doing the task with and without the proposed approach (Zenebe et al. 2010). Additionally, the author found that similar work for analysing user feedback used median not mean values (Ouyang and Li 2012).

To understand how the Intelligent Launcher can improve prediction accuracy, we asked participants to comment on situations where they have to find their target app from the full app list. Participants mentioned in their comments that when they drew a *Drawing* gesture based on their target app's *function*, the Intelligent Launcher was unable to provide the correct match. This is understandable since, in the current stage, the Intelligent Launcher can only handle the *Drawing* gesture via the Icon Template. Among that 13% of *Drawing* gestures that were based on the app's function, only those which were occasionally visually similar to the app's icon were correctly matched.

## 5.5 Discussion and Conclusions

This chapter has presented the Intelligent Launcher, a novel approach to touch-gesture interaction that allows a user to find their intended apps using touch-gestures, without the need for them to be defined or there being a limitation on gesture type. Our exploratory experiment has contributed to the understanding of gesture shortcuts that users typically adopt when searching for their target apps. Based on the findings from this experiment, we proposed three guidelines for gesture-based prediction interface, which formed the building blocks of our Intelligent Launcher.

Then, we implemented the Intelligent Launcher based on our findings and chose the HMM and SVM machine learning methods for prediction, following the literature. This chapter has also evaluated the accuracy of the Intelligent Launcher in an experiment. The results show that it predicted the target app with a 96% of success rate. Of the *Letter* gestures, the prediction based on an app's *function* contributed more than one-fifth (21%)

of all correct predictions. On average, the target which was inferred from the app's *icon* located closest (position 2.2) to the top choice (position 1) in the candidate's target list (8 positions). The prediction accuracy based on an app's *function* and *app usage history* did not decline when the size of the candidate's target list was reduced to the top 6 matches. Also, the accuracy based on an app's *icon* did not decline when the size of the list was reduced to the top 4 matches.

The Intelligent Launcher provides another quick way for accessing target apps on smartphones by using the machine learning method, while the prediction relies on multiple resources (i.e., LGR, DGR and AGR). As we currently cannot tell what type (shape) of gesture the user has drawn, we give these three recognisers equal weight when calculating the portability value. Based on our observations, the user does not use these three types (shapes) of gestures equally. Therefore, we believe that the Intelligent Launcher has the potential to improve in the future once we have a better understanding of what influences the user employs to choose a gesture type (shape) access the target.

## Chapter 6

# Factors that Influence the Choice of a Touch Gesture for App Access

### 6.1 Introduction

In chapter 5, we proposed and evaluated the Intelligent Launcher. It does not need the user to define the gesture shortcut before using it to access a target app. It can predict which app the user wants to access based on the gesture drawn by the user. The prediction relies on whether and how the given gesture matches to the app's app's icon, function, name and usage history. However, this information is not enough for guessing which kind (shape) of gesture the user had drawn. Additionally, different from previous approaches (predicate the target app via a single source), the "Predictor" in Chapter 5 relies on three different sources (i.e., templates built with the app's name, function, icon and usage history) for three shapes (i.e., *Letter*, *Drawing* and *Abstract*) of gesture recognition separately. In this case, Chapter 5 balanced the prediction results by giving equal weight to the three recognisers. One method of improving this prediction accuracy is to find out the real weight (rather than weighting each part equally) of each recogniser. In other words, to find out the pattern of how often each shape of gesture is used in real

life would be lead to better prediction of the target app.

To this end, we have hypothesized that it would be essential to know the users' preferences regarding gesture creation reasons and app search methods, the smartphone systems they use, and the retention interval and the app usage frequency. The reason we believe these factors may influence the choice of a touch gesture is based on reviewing the literature. The details are presented in Section 6.2.

The main findings of our experiment with 36 participants have confirmed these hypotheses. Here we summarize and discuss our key findings. The results show that: (1) participants tended to use the same creation method as the preferred method on different days of the experiment; (2) those who preferred the name-based method used it more consistently and used more *Letter* gestures compared with those who preferred the other three methods; (3) those who preferred the keyword app search method created more *Letter* gestures than other types; (4) those who preferred the iOS system created more *Drawing* gestures than other types; (5) the *Letter* gestures were more often used for frequently used apps, whereas *Drawing* gestures were more often used for infrequently used apps. These results demonstrate the importance of the user's profile for predicting an intended app. An intelligent app launcher was designed based on these findings. We expect that the results from this study can inform the design of future gesture-based prediction interfaces.

The findings of our work aim to contribute to the design of future gesture-based prediction interfaces by:

- proposing a mixed sources gesture-app prediction method.
  
- specifying a Bayesian network model for our proposed method.
  
- reviewing existing studies which focusing on fast app access and proposing the factors may influence the choice of the gesture.
  
- evaluating the degree of the effect of each factor on the choice of the gesture.

## 6.2 Method

To investigate how to improve the gesture-based predictor, we conducted an experiment to collect (1) participants' personal preferences and smartphone systems via a pre-experiment questionnaire; (2) gestures created by participants for target apps via the Agile Gesture app (Zhang et al. 2016); and (3) reasons for gesture creation via gesture creation questions. Firstly, we explained the "personal preferences" and "smartphone systems" of our experiment. Secondly, we introduced why we classified the installed apps based on the using frequency and chose the interval time for repeated tasks. Thirdly, we presented the two questions and why the questions needed to be address. Finally, the experiment method, including the information from the participants who took part in the experiments, the procedures and tasks were compiled and presented in the form of the evaluation experiment and the experiment design.

### 6.2.1 Factors







The following is an explanation of why we believe that the selected factors can make the users employ different types of gestures as shortcuts.

#### Gesture Creation Reason

Studies have identified different types of gestures created by users for different reasons (Nacenta et al. 2013; Poppinga et al. 2014; Wobbrock et al. 2009; Zhang et al. 2016; Lü and Li 2011). These include *Letter*-like gestures based on the app's name and *Drawing*-shaped gestures based on the app's icon. However, users do not always create the same type of gestures over time, even for the same app. For example, they may use a letter "g" gesture for "Google Map" first and then use a "location mark" gesture. Discovering the reason behind the creation could help the predictor to work out the gesture type. For example, a gesture created from an app's name is always a letter shape and from an app's icon, a *Drawing* shape. In this study we asked our participants to specify their preferred gesture creation reason (i.e. *name, icon, function or other*) first. We then analysed how consistent their gesture choice was with their preference, and how this reason influenced

the gesture they had created. The gesture samples created based on each reason are demonstrated in Table 6.1. The "ct" gesture was created because the "Messenger" app has a "chatting" function. For the gestures that were not created via a "name", "icon" or "function" reasons they will be referred to as "other" in this thesis.

Table 6.1: Gesture samples of four gesture creation reasons in the experiment.

Reason	<i>name</i>	<i>icon</i>	<i>function</i>	<i>other</i>
Icon				
App	Gmail	YouTube	Messenger	WhatsApp
Gesture	GM		ct	

### App Finding Method

The *keyword search* (Apple 2017; Google 2015a), interacting with traditional *GUI* (Li 2012) and *gesture-based* (Li 2010a; Lü and Li 2011; Zhang 2016) methods were considered in our study since they are widely used. The *Keyword search* method allows users to find the target app by typing the keyword of an app via a keyboard. Then, users can select the app from the short-listed results. The *gesture-based* search provides users with the target app they want to access via a gesture shortcut that has they themselves have drawn. The *GUI*, in this thesis, refers to the method of accessing the target app by touching the app's icon. In a real-world setting, a user chooses one of these methods for accessing apps rather than liking them equally. Based on this, we evaluated whether this preference difference can influence the gesture choice.

### Smartphone System

The *Android* (Google 2015b) and *iOS* (Apple 2015) systems have more than a 99% market share (IDC 2017). At the same time, as (1) their UI and icon designs are very different (Google 2018b c) and (2) their users differ in demographic background, such as graduate degree and income level, etc. (Ubhi et al. 2017), they have established different characteristics (Gerpott et al. 2013) and behaviours (Benenson et al. 2013). A survey reported that Android users used over 30% more game apps than *iOS* users (Perez



2017). This may affect gesture choice for app access. Research has considered these differences by using different designs for *iOS* and *Android* users (Liu et al. 2013; Zimbra et al. 2017). Here, we aim to provide suggestions for both systems by investigating whether their users create their gestures differently.

### **App Usage Frequency**

The conventional next app prediction methods rely on the retention interval (Huang et al. 2012; Tan et al. 2012) and the app usage frequency (Liao et al. 2012; Yan et al. 2012; Zhang et al. 2012; Zou et al. 2013). Should the predictor assign the same prediction strategy for both frequently and infrequently used apps over time? To answer this, it is necessary to examine the effect of gesture creation on usage frequency and different retention intervals. We classified the apps in our study based on the degree of usage frequency. According to a report, just 11% of *Android* and 10.2% of *iOS* apps were retained within a week (eMarket 2015). Additionally, according to Thalheimer (2010), people usually forgot 77% of what they had learned after seven days. This means that there is a great number of installed apps which are hard to remember as they have not been retained in more than 7 days. Therefore, we defined the apps that had been retained within seven days as *frequent used apps*, whereas the rest were defined as *infrequent used apps*.




### **Retention Periods**

Smartphone app users use between 1 and 5 apps per day, 6 and 10 apps per week (Hopwood 2017). Less access means that the user may change the shortcut creation. To ensure the prediction works for different retention periods, we evaluated the gesture creation method and gesture type three times. The test was taken *immediately* after the users' preference and smartphone system was asked in order to mimic the first use in a real-world scenario, then *24 hours* after, in order to mimic a short time retention scenario, and then *7 days* after, to mimic a long-time retention scenario.

### 6.2.2 Participants

A total of 36 university students (15 female, 21 male), aged between 18 to 36 years old ( $M = 26.3$ ,  $SD = 6.7$ ), took part in the experiment. All were daily users of smartphones, reporting 0.5 to 6.0 hours usage per day ( $M = 4.5$ ,  $SD = 4.3$ ), with 50 to 131 third-party apps ( $M = 84$ ,  $SD = 22$ ) installed on their phones at the time of the experiment.

Table 6.2: Gesture samples of three gesture shapes in the experiment.

Shape	<i>letter</i>	<i>drawing</i>	<i>abstract</i>
Gesture			

### 6.2.3 Procedure

The experiment was conducted over a period of seven days. The detailed procedures on each day were as follows:

**Day 1.** The participants gave informed consent. They then completed a pre-questionnaire that collected their demographic data, and their personal preferences for app usage (see Table 6.3). Following this, they familiarized themselves with the data collection tool. Meanwhile, the experimenter randomly selected an equal number of frequent and infrequent used apps as the sample target apps for each participant according to the participant's response to the pre-experiment questionnaire. Six participant had the same number of target apps in their sample set.

After practising, the participants completed the gesture creation and creation method report sessions:

Table 6.3: Pre-experiment Questionnaire

Participant ID:

- Q1 How long since you own a smartphone?
- Q2 How many hours do you spend on your smartphone everyday?
- Q3 Give the name of the apps you used in recent seven days.
- Q4 Give the installed app list to experimenter.
- Q5 Which of the following methods do you prefer as a gesture shortcut for your app?  
 1) Name-based 2) Icon-based 3) Function-based 4) Other
- Q6 Which smartphone system do you use?  
 1) Android 2) iOS
- Q7 Which app searching method do you prefer?  
 1) Keyword search 2) Graphic interface 3) Gesture shortcut



- *Gesture creation.* Each participant was presented with a list of apps icons and names printed on the paper. The apps were presented one at a time in a random order. The participant was instructed to create a gesture shortcut access for each app. This was repeated until gestures for all sample target apps were created. The sample target apps included equal number of frequent and infrequent used app and its size was depended on the number of the installed apps the participants have. The size of the sample target set and the personal information of the participants in each group is shown in Table 6.4.

Table 6.4: App allocations and participant information in the experiment. (Note "SD" in the table means the Standard Deviation.)

<b>Number of installed apps</b>	50-59	60-69	70-79	80-89	90-99	> 100
<b>Assigned frequently used apps</b>	5	6	7	8	9	10
<b>Assigned infrequently used apps</b>	5	6	7	8	9	10
<b>Number of Participants</b>	6	6	6	6	6	6
<b>Mean of Participants' age</b>	30.3	25.5	27.2	24	26.7	24.3
<b>SD of Participants' age</b>	5.8	7.1	7.9	5.5	7.4	7.6
<b>Number of female</b>	3	2	3	2	3	2

- *Report gesture creation reason.* Immediately after gesture creation, participants were asked to report their gesture creation method on a form for each gesture they created, one at a time (see Table 6.5 for an example). Participants were not shown the answer they had used before. If the participants report they created the gesture by "other", they were asked to describe the reason they applied.

Table 6.5: Gesture creation reason question  
Participant ID:

Name: Flickr Icon:  Gesture:   
Please select the reason why you drew the gesture above.  
1) Name 2) Icon 3) Function 4) Other  
If you choose 4) Other please describe the reason.

**Days 2 and 7.** The tests and procedure on the second and seventh day of the experiment were identical. The participants were required to first do the "gesture creation"

and then the "report gesture creation reason" tasks for the same list of target apps as on **Day 1**. The procedure of the two tasks was the same as that of **Day 1**. The order of gesture creation for frequent and infrequent used apps was counterbalanced over the first, second and seventh day of the experiment.

#### 6.2.4 Experiment Design

We employed a mixed-design methodology. The within-participant variables were App Usage Frequency (*frequent used app vs. infrequent used app*), and Retention Interval (*0-day, 1-day and 7-days*). The between-participant variables were User Preference including: Preferred Creation Reason (*name, icon, function and other*), Preferred App Finding Method (*keyword search, GUI and gesture-based*), and the participants' Smartphone System (*Android vs. iOS*). The dependent variable was the participant's gesture creation reason consistency and the rate of the type of gesture. Here we defined the consistency as the proportion of created gestures consistent with the preferred method. The rate of the type of gesture was calculated as the proportion of a type's occurrence out of all gesture creating by a participant.

The ideal predictor can automatically set up the prediction algorithm based on the user's answer for their preference. However, whether and how the user can be consistent with their first answers remains unknown. Therefore, we designed a mixed-design experiment to check whether these User Preferences (i.e., the Preferred Creation Reason, Preferred App Finding Method and participant's Smartphone System) changed within both the short interval (24 hours) and the relatively long interval (7 days) when related to the initial creation for both *frequent* and *infrequent* used apps. Based on these results, we can infer the degree on contribution of each User Preference during the time lapsed.

### 6.3 Results

Note that, in this section, our data set has two properties. First, the data we collected are percentage data (i.e., the reason's consistency from 0 to 100% and the type's rate from 0 to 100%, which is defined in Section 6.2.4. Second, the values in the data set are

not normally distributed. Due to these properties, the methods addressing these issues are considered. The non-parametric methods is a way of addressing the normality issue which our data set has (Vickers 2005). However, for our percentage data, an arcsine transformation is recommended as it can provide a high percentage of correction for the non-normality, the heterogeneity of variance, and the nonadditivity (Ahrens et al. 1990). It has also been suggested that even though a difference is caused by the transformation appearing in mean separations determined using LSD (.005), this difference has only a minor effect on the interpretation of the results. Therefore, the proportion consistency data and the type's rate data were arcsine transformed before further statistical analysis. Where sphericity was violated, the degree of freedom was adjusted by applying the Greenhouse Geisser method.

### 6.3.1 Gesture creation reason consistency

The results are shown in Figure 6.1. To examine how the users' gesture creation preference, the target app's usage frequency and the retention interval (see the reason in 6.2.1) affected the **consistency**, we conducted a 4 Preferred Creation Reason  $\times$  2 App Usage Frequency  $\times$  3 Retention Interval mixed-measures analysis of variance (ANOVA). As the ANOVA results in Table 6.6 shows, the significant main effects and two-way interaction effects were qualified by significant three-way interaction. To identify the source of this three-way interaction, we conducted separate simple effects analyses for the results of the *frequent* and *infrequent* used apps.

Table 6.6: ANOVA of the **consistency** on Gesture Creation Reason for *frequent* and *infrequent* used apps with *0-day*, *1-day* and *7-days* Retention Interval of Preferred Gesture Creation Reason.

Source	<i>df</i>	MSE	F	<i>p</i>
App Usage Frequency (UF)	1	.152	.56	.46
Retention Interval (RI)	2	.017	3.59	.033
Preferred Creation Reason (CR)	3	.351	14.63	< .001
UF $\times$ RI	2	.013	6.83	.002
UF $\times$ CR	3	.152	2.89	.051
RI $\times$ CR	6	.017	3.01	.012
UF $\times$ RI $\times$ CR	4.94	.015	2.43	.048
Error	52.71			

note. *df* = degrees of freedom; MSE = Mean Squared Error.

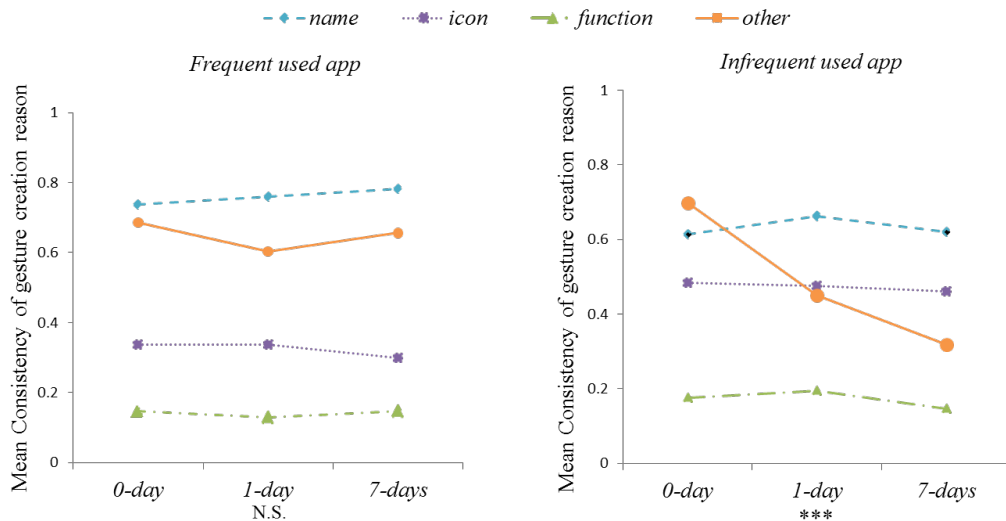


Figure 6.1: Mean reason **consistency** of four Preferred Creation Reasons on three different Retention Intervals for *frequent used app* (left) and *infrequent used app* (right). N.S. = not significant. \*\*\*  $p < .001$ .

For the *frequent* used apps, the main effect of the Retention Interval was not significant,  $F(2, 64) = 1.47$ ,  $p = .238$ . The interaction between this factor and the Preferred Creation Reason was also not significant,  $F(6, 64) = 0.88$ ,  $p = .514$ . However, the main effect of the Preferred Creation Reason was significant,  $F(3, 32) = 15.22$ ,  $p < .001$ . The Bonferroni post hoc test found that the consistency was greater for participants who preferred the name reason than those who preferred the icon reason,  $p < .001$  and the function reason,  $p < .001$ , the other reason than the icon reason,  $p = .038$ , and the function reason,  $p = .002$ .

For the *infrequent* used apps, the main effects of the Retention Interval,  $F(2, 64) = 8.34$ ,  $p = .001$  and the Preferred Creation Reason,  $F(3, 32) = 1.89$ ,  $p = .151$ , was qualified by a significant interaction between them,  $F(6, 64) = 4.74$ ,  $p < .001$  (see Figure 6.1). The pairwise comparisons for the main effect of the Retention Interval were conducted using a Bonferroni post hoc test. The Bonferroni post hoc test found that the consistency was significantly greater on the 0-day results than on the 7-days,  $p = .001$  and on the 1-day than the 7-days,  $p = .042$ . However there was no significant difference between the 0-day and 1-day,  $p = .442$ .

Overall, the results indicate that the three-way interaction is statistically significant. This is due to the two-way interaction of Preferred Creation  $\times$  Retention Interval not being significant for *frequent used apps* but significant for *infrequent used apps*. The

results show that for *frequent* used apps, a preference for *name* was more consistently linked to the gestures choice in practice than the preferences for *other* gesture types. For *infrequent* used apps, however, the consistent scores were affected by the Retention Interval.

### 6.3.2 User's Profile and Gesture they choose

To examine how the users' profiles affected their choice of touch gesture, we conducted a series of three-way ANOVAs with different independent variables separately as follows:

- (a) 4 Preferred Creation Reason  $\times$  2 App Usage Frequency  $\times$  3 Retention Interval.
- (b) 4 Preferred App Finding Method  $\times$  2 App usage Frequency  $\times$  3 Retention interval.
- (c) 2 Mobile System  $\times$  2 App usage Frequency  $\times$  3 Retention interval.

As introduced in Section 6.2.4, the App Usage Frequency and the Retention Interval are both within-variables and the Preferred Creation Reason, App Finding Method and the Mobile System are between-variables. The dependent variable was the **rate** of the corresponding shape of gestures.

#### (a) Preferred Creation Reason

To examine how the users' Preferred Creation Reason affected the Gesture they chose (i.e., the **rate** of the *Letter*, *Drawing* and *Abstract* shaped gestures being used), three separate three-way mixed-measures analyses of variance (ANOVA)s were conducted. Note here, the independent variable is the **rate** of the *Letter*, *Drawing* and *Abstract* shaped gestures being used.

- **Letter** gesture

As the ANOVA results in Table 6.7 show, the significant main effects and two-way interaction effects were qualified by significant three-way interaction. Separate simple effects analyses for the results of *frequent* and *infrequent* used apps was conducted (see Figure 6.2 top). For *frequent* used apps, there was only a main effect

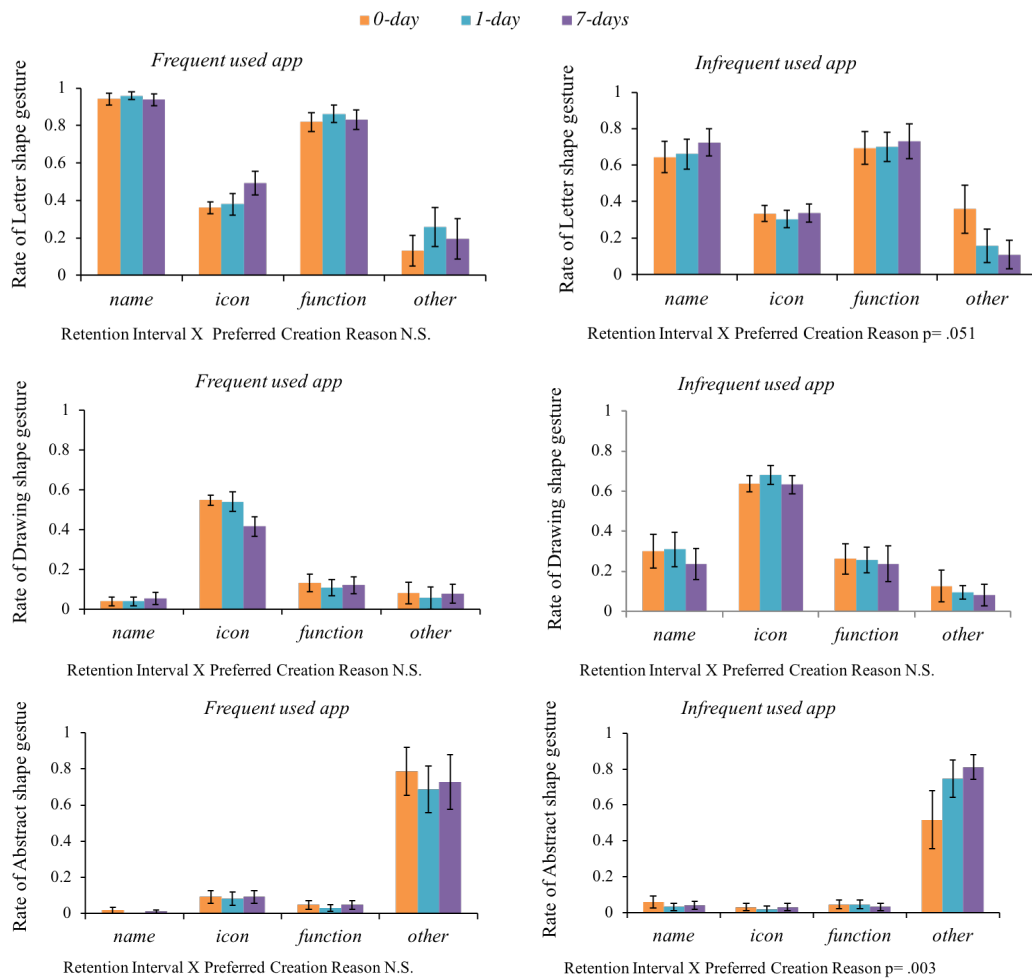


Figure 6.2: Mean **rate** of top: *letter*, middle: *drawing* and bottom: *abstract* gesture shape for four Gesture Creation Reasons on three different Retention Intervals to access *frequent* (left) and *infrequent* (right) used apps. The different bar in each group represent the **rate** of shape created on the Retention Interval of *0-day*, *1-day* and *7-days*. The error bars stand for one standard error about the mean. N.S. = not significant.



Table 6.7: ANOVA of the *Letter* gesture **rate** on four Preferred Gesture Creation Reasons for *frequent* and *infrequent* used apps with three Retention Intervals.

Source	<i>df</i>	MSE	F	<i>p</i>
App Usage Frequency (UF)	1	.002	.02	.877
Retention Interval (RI)	2	.180	8.34	.001
Preferred Creation Reason (CR)	3	6.52	11.04	< .001
UF × RI	2	.080	5.55	.006
UF × CR	3	.662	7.24	.001
RI × CR	6	.049	2.25	.049
UF × RI × CR	6	.060	4.19	.001
Error	64			

note. *df* = degrees of freedom; MSE = Mean Squared Error.

of Preferred Creation Reason,  $F(3, 32) = 38.57$ ,  $p < .001$ , where participants who preferred *name* and *function* got a higher rate of *Letter* gesture than *icon* and *other* gesture creation reasons,  $p < .001$ . For *infrequent* used apps the main effect of Preferred Creation Reason was significant,  $F(3, 32) = 9.75$ ,  $p < .001$ . The rate of *Letter* gesture was higher for participants who preferred the *name* and *function* reason than for the other two reasons,  $p < .001$ . Even though the simple effects for both *frequent* and *infrequent* used apps were both not significant, for the *infrequent* used apps, there was a strong tendency towards statistical significance ( $p = .051$ ).

Table 6.8: ANOVA of the *Drawing* gesture **rate** on four Preferred Gesture Creation Reasons for *frequent* and *infrequent* used apps with three Retention Intervals.

Source	<i>df</i>	MSE	F	<i>p</i>
App Usage Frequency (UF)	1	1.122	14.96	.001
Retention Interval (RI)	2	.001	.07	.933
Preferred Creation Reason (CR)	3	6.141	12.06	< .001
UF × RI	2	.011	.59	.555
UF × CR	3	.190	2.54	.074
RI × CR	6	.024	1.24	.3.01
UF × RI × CR	6	.039	2.02	.075
Error	64			

note. *df* = degrees of freedom; MSE = Mean Squared Error.

- ***Drawing*** gesture

As shown in Table 6.8, significant main effects were found for App Usage Frequency,  $F(1, 32) = 14.96$ ,  $p = .001$ , and Preferred Creation Reason,  $F(3, 32) = 12.06$ ,  $p < .001$ . All other main effects, the two-way interaction and the three-way interaction were not significant,  $p \geq .074$ . Separate simple effects analyses for *frequent* and *infrequent* used apps were conducted (see Figure 6.2 middle). They revealed a

significant main effect of Preferred Creation Reason for both *frequent*,  $F(3, 32) = 33.47$ ,  $p < .001$ , and *infrequent* used app,  $F(3, 32) = 4.08$ ,  $p = .015$ . In *frequent* used app, the **rate** of *Drawing* gesture was significantly higher for participants who preferred the *icon* reason more than the other three reasons,  $p < .001$ . However, for *infrequent* used apps, the **rate** was only significantly higher for participants who preferred the *icon* reason more than the *other* reason,  $p = .028$ .

Table 6.9: ANOVA of the *Abstract* gesture **rate** on four Preferred Gesture Creation Reasons for *frequent* and *infrequent* used apps with three Retention Intervals.

Source	df	MSE	F	p
App Usage Frequency (UF)	1	.968	13.28	.001
Retention Interval (RI)	2	.195	12.69	< .001
Preferred Creation Reason (CR)	3	5.613	34.37	< .001
UF × RI	2	.045	3.78	.028
UF × CR	3	.440	6.03	.002
RI × CR	6	.107	6.99	< .001
UF × RI × CR	6	.019	1.61	.158
Error	64			

note. df = degrees of freedom; MSE = Mean Squared Error.

- **Abstract** gesture

As the ANOVA results in Table 6.9 show, the significant main effects of Preferred Creation Reason,  $F(3, 32) = 34.37$ ,  $p < .001$ , App Usage Frequency,  $F(1, 32) = 13.28$ ,  $p = .001$ , and Retention Interval,  $F(2, 64) = 12.69$ ,  $p < .001$ , were qualified by significant two-way interactions. However, the three-way interaction was not significant,  $F(6, 64) = 1.16$ ,  $p = .158$ . The simple effects analyses are shown in the bottom of Figure 6.2. There were main effects of Preferred Creation Reason, for *frequent* used apps,  $F(3, 32) = 33.87$ ,  $p < .001$  and for *infrequent* used apps as well,  $F(3, 32) = 31.82$ ,  $p < .001$ . The **rate** of *Abstract* gesture was significantly higher for participants who preferred the *other* reason than the other three reasons for both *frequent* and *infrequent* used apps,  $p < .001$ .

**(b) Preferred Search Method**

Besides the user's Preferred Creation Reason, the user's Preferred Search Method may influence the Gesture (shape) they choose. We, therefore, identified how this factor affected the shape of the gesture the user chose by conducting three separate three-way

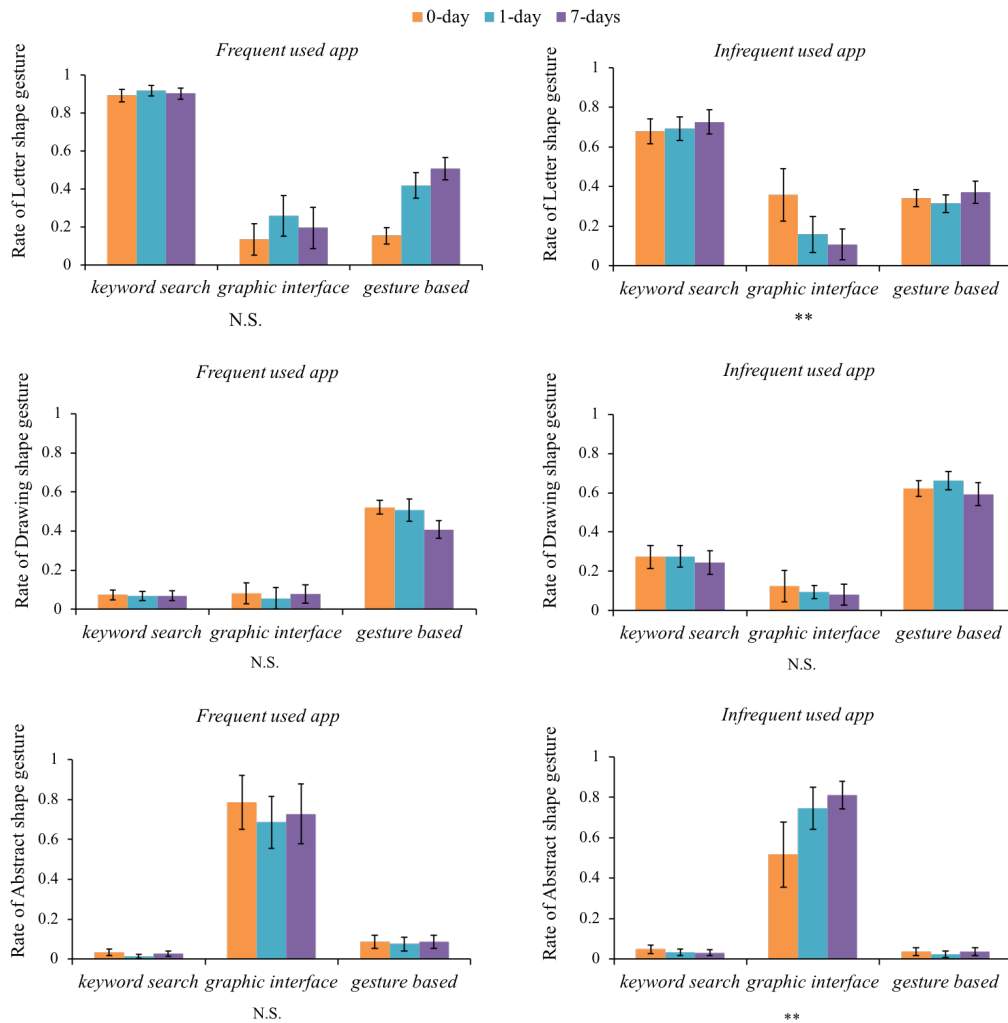


Figure 6.3: Mean **rate** of top: *Letter*, middle: *Drawing* and bottom: *Abstract* gesture shape for three Preferred Search Method on three different Retention Intervals to access *frequent* (left) and *infrequent* (right) used apps. The different bar in each group represents the **rate** of shape created on the Retention Interval of *0-day*, *1-day* and *7-days*. The error bars stand for one standard error about the mean. N.S. = not significant. \*\* p < .05

mixed-measures analyses for the *Letter*, *Drawing* and *Abstract* gestures. The independent variable is the **rate** of the *Letter*, *Drawing* and *Abstract* shape gestures been used.

- **Letter** gesture, results are shown in Figure 6.3 top.

The main effects of App Usage Frequency,  $F(1, 33) = 10.83$ ,  $p = .002$ , and Preferred Search Method,  $F(2, 33) = 31.22$ ,  $p < .001$ , were qualified by interaction between these two factors,  $F(2, 33) = 5.40$ ,  $p = .009$ , which was in turn qualified by a three-way interaction,  $F(4, 66) = 3.26$ ,  $p = .017$ .

To identify the source of the interaction, we conducted separate simple effects analyses for the results of the *frequent* and *infrequent* used apps. For the *frequent* used apps, there was a main effect of the Retention Interval,  $F(2, 66) = 3.49$ ,  $p = .036$ , where the **rate** of *Letter* gesture was higher for *0-day* than for *7-days*,  $p = .043$ . There was also a main effect of Preferred Search Method,  $F(2,33) = 46.88$ ,  $p < .001$ , where the *Letter* gesture **rate** was higher for the *keyword search* than for the *GUI*,  $p < .001$ , or *gesture-based* methods,  $p < .001$ . The interaction between the two variables was not significant,  $F(4, 66) = 1.76$ ,  $p = .147$ . For the *infrequent* used apps, the Preferred Search Method  $\times$  Retention Interval effect was significant,  $F(4, 66) = 3.20$ ,  $p = .018$ . The effect of the Retention Interval was not significant in participants who preferred the *keyword search* method,  $t(18) > .03$ ,  $p < .974$ , *GUI*,  $t(3) \geq .25$ ,  $p \leq .820$ , or the *gesture-based* method,  $t(12) \geq .72$ ,  $p \leq .483$ . However, the effect of the Preferred Search Method was significant on the three Retention Intervals,  $F(2, 33) \geq 8.28$ ,  $p \leq .001$ , and the participants who preferred the *keyword search* method created more *Letter* gestures than other two methods,  $p \leq .032$ .

- **Drawing** gesture (Figure 6.3 middle).

The main effects of the App Usage Frequency,  $F(1, 33) = 18.73$ ,  $p < .001$ , and the Preferred Search Method,  $F(2, 33) = 25.87$ ,  $p < .001$ , were significant. All other effects were not significant. More *Drawing* gesture was created for *infrequent* than *frequent* used apps,  $t(35) = 3.41$ ,  $p = .002$ . The participants who preferred *gesture-based* methods drew more *Drawing* gestures than the other two methods,  $p \leq .027$ .

- **Abstract** gesture (Figure 6.3 bottom).

The main effect of the Preferred Search Method,  $F(2,33) = 69.39$ ,  $p < .001$ , was qualified by a two-way interaction (App Usage Frequency  $\times$  Retention Interval,  $F(2, 66) = 5.78$ ,  $p = .005$ , and a three-way interaction,  $F(4, 66) = 4.23$ ,  $p = .004$ .

ANOVA for *frequent* used apps showed no main effect of the Retention Interval,  $F(2, 66) = 2.51$ ,  $p = .089$ , or interaction effect between the Preferred Search Method and Retention Interval,  $F(4, 66) = .60$ ,  $p = .661$ . However, there was a significant effect of the Preferred Search Method,  $F(2, 33) = 48.27$ ,  $p < .001$ , where the **rate** for *Abstract*-shaped gestures was higher for participants who preferred the *gesture-based* method than the other two methods,  $p < .001$ . For *infrequent* used apps, the effect of the Retention Interval,  $F(2, 66) = 4.94$ ,  $p = .010$ , the Preferred Search Method,  $F(2, 33) = 48.37$ ,  $p < .001$ , and the interaction between the two factors,  $F(4, 66) = 5.48$ ,  $p = .001$ , were significant. The effects of the Retention Interval on the three Preferred Search Methods were not significant,  $p \geq .061$ . However, the effect of the Preferred Search Method was significant on the three Retention Intervals,  $F(2, 33) \geq 19.09$ ,  $p < .001$ . The participants who preferred the *GUI* created more *Abstract* gestures than the other two methods,  $p < .001$ .

### (c) Smartphone System

Finally, whether the smartphone system the user used influenced the gesture they chose was investigated. Three three-way mixed-measures analyses were conducted. The dependent variables are as designed in the beginning of this section (6.3.2). The independent variable is the **rate** of the *Letter*, *Drawing* and *Abstract* shape gestures being used.

The results of two-way interactions which are broken down into *frequent* and *infrequent* used apps are shown in Figure 6.4. The two-way interaction for *frequent* used apps was significant,  $F(2, 68) = 19.91$ ,  $p < .001$ , where it is suggested that there is a higher rate of *Letter* gesture use for the *Android* user than for the *iOS* user,  $F(1, 34) = 36.74$ ,  $p < .001$ , and a higher rate of *Drawing* gesture for the *iOS* user than the *Android* user,  $F(1, 34) = 29.25$ ,  $p < .001$ . No effect for *Abstract* gesture was recorded. For *infrequent* used apps, the two-way interaction was also significant,  $F(2, 68) = 4.53$ ,  $p = .014$ .

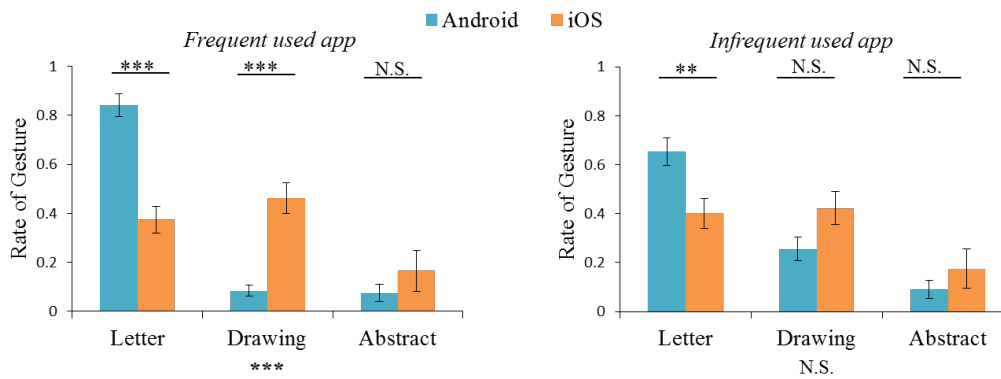


Figure 6.4: Mean **rate** of three types for *frequent* (left) and *infrequent* (right) used apps created by the *Android* and the *iOS* user. The different bars in each group represent the **rate** of gesture created by the *Android* or the *iOS* user. The error bars stand for one standard error about the mean. N.S. = not significant. \*\*\*  $p < .001$ .

The smartphone's system only affected the *Letter* gesture,  $F(1, 34) = 9.09, p = .005$ , where the result was consistent with the frequent used apps.

As shown in Figure 6.5, for the *Android* user, the two-way interaction was significant,  $F(1.50, 31.42) = 24.16, p < .001$ . The results of a paired t-test show a higher rate of *Letter* gesture for *frequent* than *infrequent* used apps,  $t(21) = 5.29, p < .001$ . However, there was a higher rate of *Drawing* gesture for the *infrequent* than the *frequent* used apps,  $t(21) = 5.53, p < .001$ . For the *iOS* user, the two-way interaction was not significant,  $F(2, 26) = 0.08, p = .919$ .

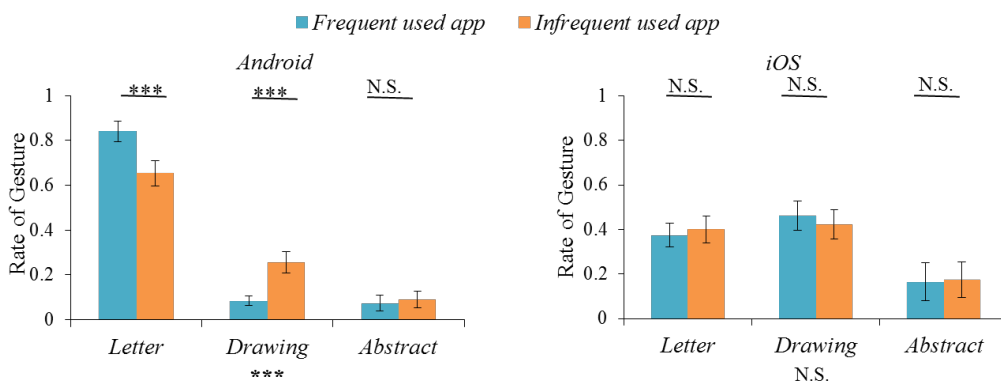


Figure 6.5: Mean **rate** of three gesture types of the *Android* user (left) and the *iOS* user (right) for *frequent* and *infrequent* used app. The different bars in each group represent the **rate** of gesture created for *frequent* or *infrequent* used apps. The error bars stand for one standard error about the mean. N.S. = not significant. \*\*\*  $p < .001$ .

## 6.4 Discussion

### 6.4.1 Gesture creation reason

When participants indicated a preference for using the names of apps, they showed a greater level of **consistency** (0.64) for using a letter of a *name* in their gesture creation than those who preferred *other* (0.57), *icon* (0.38) or *function* (0.11). This was predicted because there is a natural connection between the name of an app and the initial letter of the app's name. In contrast, the icon, function or other aspects of an app do not have a coherent rule to follow.

Additionally, there was a Preferred Creation Reason  $\times$  Retention Interval interaction effect on the mean reason consistency for *infrequent* but not for *frequent* used apps. Our results also show that, rather unexpectedly, the mean reason consistency was 0.45 on the first day, 0.42 on the second day and 0.40 on the seventh day, where the negative impact on the method consistency did not show significantly with either the short interval (24 hours) or the relatively long interval (7 days) following initial creation. This means that the degree of contribution of the preferred gesture creation reasons remained similar during the lapsed time.

### 6.4.2 User's Profile and Gesture Type

We predicted the user's profile would have an impact on the gesture types they created, particularly on the *Letter* and *Drawing* gestures. The main findings of our experiment confirm this hypothesis. Here we summarize and discuss our key findings.

#### Preferred Creation Reason and Gesture type

In our study, the participants created 64.8% *Letter* gestures, 25.9% *Drawing* gestures and 9.2% *Abstract* gestures. In general, the participants who preferred the *name* and *function* reasons created more *Letter* gestures than other participants. This was predicted because it is common to use a letter to represent an app's name: for example, using the gesture "G" to access the "Gmail" app since it is the initial letter of the app's name, or

using the gesture "C" to access a chatting app like "Messenger". In addition, rather interestingly, users tend to use a *Letter* but not a *Drawing* or *Abstract* gesture to represent an app's function: for example, using the gesture "N" to access the "GoogleMap" app since it is the initial letter of the app's navigation function. This may be because, in this case, it takes less effort to create a gesture directly based on a word (verbalization) than to create an image as a drawing (visualization) (Bell and Lindamood 1991; Blasius and Greenacre 2014).

Our results also show that among all *Drawing* gestures, most (78%) of them were created by those who preferred the *icon-based* reason. This may be because the icon is explicating to users and aiming for leaving user an impression (Chen 2015; Pol 2015). Therefore, users may create a gesture which is similar to the icon, based on their previous impressions.

Another finding of this study was that the probability of using an *Abstract* gesture was often influenced by two factors. First, the participants who preferred the *other* creation reason usually created an *Abstract* gesture. Second, the *Abstract* gestures were created to access *frequent* used apps more often than *infrequent* used apps.

Although the gesture type was closely related to the participants' Preferred Creation Reason, the effects were different for the three gesture types. This means that we need to combine three factors for consideration when predicting a gesture type.

### **Preferred App Search Method and Gesture type**

Participants who preferred the *keyword search* method were most likely to create a *Letter* gesture. In contrast, those who preferred the *gesture-based* method were most likely to create a *Drawing* gesture. Although the participants who preferred the *gesture-based* method displayed a clear pattern of being most likely to create a *Drawing* gesture, our experiment found that they also created more *Abstract* gestures than other participants.

### **Preferred Mobile System and Gesture type**

The participants who preferred an *iOS* system created more *Drawing* and *Abstract* gestures than those who preferred an *Android* system. This may be due to some of the



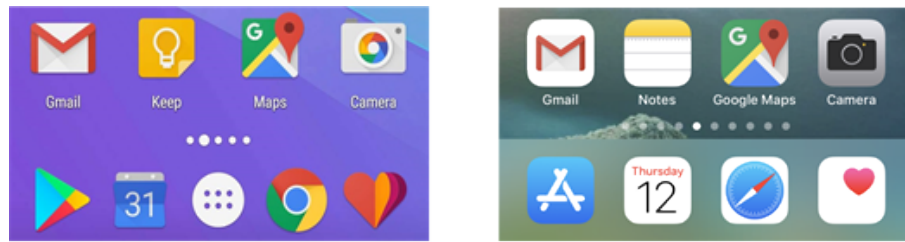


Figure 6.6: An example of app icons display on an *Android* phone (left) and on an *iOS* phone (right).

interface design differences between the *iOS* and *Android* systems. First, the *iOS* interface commonly uses flat style design icons which are surrounded by a nontransparent rectangle, while *Android* interface use a three-dimensional style icon without a surrounding. Examples are shown in Figure 6.6.

We can see that the icons of the *iOS* interface are easier to reproduce than the *Android* ones since the *iOS* icons (1) are flat rather than three dimensional, and (2) the background rectangle of the icon is like a canvas, which makes the icon stand out more. These two attributes make it easier for *iOS* users to create a *Drawing* gesture than *Android* users. Second, the *Android* interface has a universal navigation bar at the bottom. Users can switch to another app by clicking the "back button". However, there is no such button on the *iOS* interface, where a gesture of swiping from left to right is offered by Apple to support the go back action. This swiping gesture is very similar to our *Abstract* gesture, so users who were interested in this swiping gesture may have reproduced it when they switched between apps. Our study has confirmed this. A participant who preferred the *iOS* system and created 6 *Abstract* gestures told us that he had created the gestures for accessing the other apps by mimicking the swiping gestures in *iOS*.

Nevertheless, the participant's preference regarding their mobile system had almost no impact on the proportion of *Letter* gestures they created, except for the *frequent* used apps on *Day 1* of the experiment. Only in this condition, participants who preferred the *Android* system created more *Letter* gestures than those who preferred the *iOS* system. That is to say, some different patterns of use of the two mobile systems see the users tending to create different types of gesture for accessing their apps.

### 6.4.3 Gesture recognition method

Previous work for gesture recognition (Escalera et al. 2017; Rautaray and Agrawal 2015) needs a source template, and the source cannot change. This situation cannot meet our predictor's requirements. Here, we introduce our predictor's gesture recognition method. We have adopted different templates for different types of gesture, based on the gesture type results that we collected.

Our predictor indicates all input gestures as *Letter*, *Drawing* or *Abstract* gestures based on the result of T's scores which were calculated from equation (6.1). The score T is calculated as the *conditional probability* and it is used here with the user's Preferred System (S), Search Method (M), Creation Method (C) and App Usage Frequency (F). In our case, the *conditional probability* is the probability of a type of *Letter/Drawing/Abstract* shape gesture being used based on a given set of variables (S,M,C,F), denoted  $P(T | S, M, C, F)$ .

The type with the highest T score among the three types is indicated as the type of gesture used. For gestures that classify as a *Letter* gesture, we adopt the alphabetical template as the source of recognizer; for those classifying as a *Drawing* gesture, we adopt the template consisting of all the installed app's icons as the source; for those classifying as *Abstract* gesture, we adopt 8 of the *Abstract* gestures templates which were also used in research by Appert and Zhai (2009), Dittmar et al. (2015) and Gillespie et al. (2014).

$$Score(T) = P(T|S, M, C, F) = \frac{P(T, S, M, C, F)}{P(S, M, C, F)} \quad (6.1)$$

Based on these findings, with an example of input gesture "M", we illustrated the architecture that we proposed for the target app prediction in Figure 6.7. The app classifier divided the users' installed apps into four categories based on the input gestures. The aim of the "Predictor source manager" is to figure out the weights ( $W_x$ ) that the predictor should use for predicting the target app based on each source. The  $W_x$  indicates the possibility of the reason for users creating the gesture. The  $W_x$  is calculated from a  $4 \times 4$  constant matrix multiplied with the **consistency** matrix. The **consistency** matrix is a  $4 \times 6$  matrix which consists of the mean reason consistency value in Table 6.10.

The app in each group of "app classifier" that we have mentioned in Figure 6.7 is

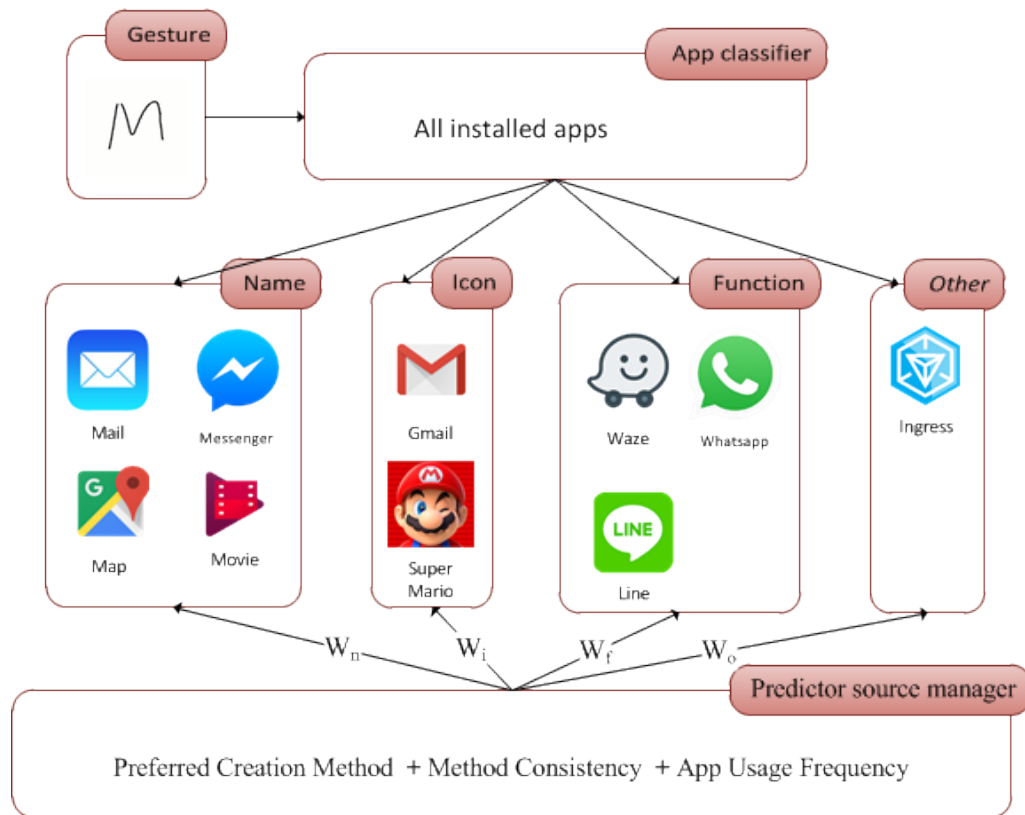


Figure 6.7: The architecture we proposed for target app prediction. An example with input gesture "M" is given. The value  $W_x$  on the line which links the predictor source manager and each source represents the weight used for prediction.

Table 6.10: Mean reason **consistency** value as a function of Preferred Creation Reason, App Usage Frequency and Retention Interval.

Reason	<i>frequent used apps</i>			<i>Infrequent used apps</i>		
	<i>0-day</i>	<i>1-day</i>	<i>7-days</i>	<i>0-day</i>	<i>1-day</i>	<i>7-days</i>
<b>name</b>	0.66	0.69	0.73	0.57	0.62	0.57
<b>icon</b>	0.35	0.35	0.30	0.45	0.42	0.4
<b>function</b>	0.08	0.07	0.09	0.13	0.15	0.11
<b>other</b>	0.69	0.60	0.66	0.70	0.45	0.32

Table 6.11: App selected principal for "app classifier".

Group	<i>Letter</i> shape gesture	<i>Drawing</i> shape gesture	<i>Abstract</i> shape gesture
<b>name</b>	name including the input letter	icon including the input letter	name including letter "i,l,z"
<b>icon</b>	icon including the input letter	icon including the input gesture	icon including the gesture
<b>function</b>	function including the input letter	function related to the input gesture	N/A
<b>other</b>	other <i>frequent used app</i>	other <i>infrequent used app</i>	other apps

elected, following Table 6.11.

Based on our studies, we have learned from the collected data and drawn up the probabilities tables (e.g., Figure 6.10) for which shape of gesture the user may adopt based on their preferences. The aim of the Intelligent Launcher is to predicate which app the user would be most likely to access based on these known factors and the gesture they draw.

Bayesian networks are ideal for this task as the sources (the probability tables which contain multiple conditions) we have already compiled and the output we require (the probability the app will be the target the user may want to access) are a suitable match for the Bayesian Network model. To justify this, it can be said that first, it is a type of probabilistic graphical model that uses Bayesian inference for probability predictions (Blodgett and Anderson 2000). Second, it has the advantage of taking an event that occurred (our probabilities tables) and predicting the likelihood that any one of several possible known causes was the contributing factor (of which type the gesture is) (Cano et al. 2004).

Another advantage is that Bayesian networks require only the known independents among the variables, rather than specifying a complete joint probability distribution, for the prediction (de Cristo et al. 2003). This advantage can be used for the prediction of quantities that are difficult, expensive or unethical to measure, based on other quantities that are easier to obtain (Margaritis 2003). In our scenario, only providing the probability of the gesture type under the conditions we already know (e.g.,  $P(T | S, M, C, F)$ ), rather than a complete probability distribution (e.g.,  $p(S | M, C, F)$ , etc.), is enough for the prediction.

So, based on the main findings in our two studies, we have constructed a Bayesian network model for the following intelligent app predictor, the Intelligent Launcher (Figure 6.8). The model assumes that the candidate for being the next app is dependent on the type of gesture which the user has created, the user's preferred gesture creation method and the app's usage frequency. The type of gesture is dependent on three user

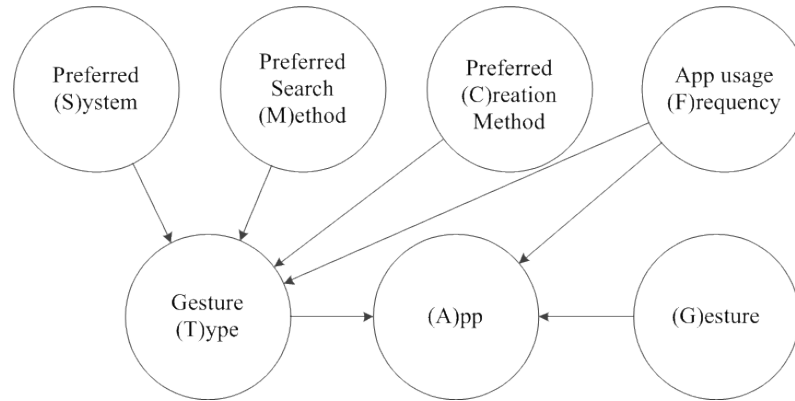


Figure 6.8: Bayesian network model for intelligent launcher.

preference factors and the app’s usage frequency. Every time the user draws a gesture, each app is assigned a score calculated as the conditional probability of the app that the user wants to access, according to equation (6.2). The probabilities of the gesture type are computed against the user’s preference and app’s usage frequency data (equation (6.3)). Based on our results, we have constructed a conditional probability table (CPT): see Table 6.1. The probability value on the right side of the equation (6.4) can be found from our CPT.

The higher the score  $A$ , the more the probability that the user wants to access this app.

$$Score(A) = P(A|T, F, G) = \frac{P(A, T, F, G)}{P(T, F, G)} \quad (6.2)$$

$$P(A, T, F, G) = P(F) \times P(G) \times P(T|F) \times P(A|F, T, G) \quad (6.3)$$

$$Score(A) = \frac{P(T|F) \times P(A|F, T, G)}{P(G|S, M, C, F)} \quad (6.4)$$

## 6.5 Conclusion

In chapter 5, the Intelligent Launcher provided an alternative approach to infer a target app via the users’ gesture. The novelty of it is two-fold. First, it has expanded the vocab-

ulary of handwriting gesture shortcuts, allowing users employing a *Letter* gesture based on either an app's name or function keyword to access the app. This provides flexibility to the user and makes it more complete. Second, it infers which app the user wants to access with a degree of similarity between the *Drawing* gesture and the apps' icon. Using the icons of apps installed on users' smartphones rather than a template created by other people makes the method highly customised. However, whether it is possible to predict a gesture type a user has chosen based on their profile, and how to do this, remains unclear.

This chapter has collated and analysed the users' profile, gestures and the reasons behind the choices of gestures for accessing target apps. We have found a relationship between the user's profile and their choice of gesture. Based on this, we have proposed an innovative gesture-based target app prediction method.

The predictor does not need to ask the reason why the user created the gesture each time it is used, and it can infer the most likely app the user wants to access. The participants in our experiment created their own gesture shortcuts for app access. After they had created a gesture and associated it with a target app, they explained why they had created the gesture. We recorded the explanations and gestures. The experiment was repeated 24 hours and 7 days following their initial creation. The gesture creation **consistency** did not significantly drop following both the short interval (24 hours) and the relatively long interval (7 days). The gesture shapes tended to be different, based on personal preferences and the contextual information.

Based on our results, the Intelligent Launcher provided (1) a Bayesian network algorithm for the gesture shape prediction, and (2) a mixed gesture recognition method for future intelligent app predictors. An example of "target app prediction" which these methods apply, is also demonstrated in this chapter.

This chapter contributes in advancing the field of HCI and gesture-based interfaces specifically by answering two important research questions: (1) does the type of gesture users choose relate to their profile information? and (2) how can these relationships and the reasons behind the gesture creation to construct a target app predictor? Future research should aim to check the accuracy of this predictor, as well as to compare its accuracy and speed with the PGS method (Zhang et al. 2016).

Overall, the preliminary findings in this study demonstrate a promising future for the proposed method. It could be offered as another option alongside existing solutions. We hope the proposed method will greatly speed up app access and improve smartphone user experience.

## Chapter 7

# Conclusion and Future Direction

### 7.1 Conclusion

This thesis proposed the "PGS" and "Intelligent Launcher" methods for app access tasks on smartphones.

Chapter 3 and 4 followed previous research to explore approaches for saving app look-up time. Two experiments were conducted to evaluate the PGS method, which included an initial study with 33 participants and 15 frequent used apps and an exploratory study with 36 participants for both frequent and infrequent used apps. This thesis proposed a gesture shortcut definition strategy and investigated its influence on gesture recall accuracy. This thesis also demonstrated the process of sampling apps of different usage frequency for user studies. Agile Search was developed to test this accuracy. Agile Gesture was developed to test both the recall accuracy and app access speed.

The preliminary findings regarding the PGS method demonstrated a promising future for the proposed method. Overall performances on app look-up time showed that using the PGS method can significantly improve ( $P < 0.05$ ) the speed of looking up the target app rather than using a traditional graphic interface (GUI). One drawback of this proposed method is that this method relies on the users' memorisation of their self-defined gestures. Our work provided evidence that this limitation would not extend to different re-



call intervals; at least the recall accuracy after the first day and after seven days following the initial gesture definition did not show any differences.

However, Chapter 4 also found, based on the natural differences of *frequent* and *infrequent* used apps, only relying on the PGS method may lead to several challenges. For example, the great advantage of finding infrequent used apps may be limited by the users' ability to memorize. A way to eliminate this limitation on memorisation was explored and described in the following Chapter.

The Intelligent Launcher provides an alternative way of using touch gesture for accessing apps. To explore how to design the Intelligent Launcher, Chapter 5 collected the gesture shortcuts 36 users created for accessing their target apps. A data collection tool was developed for collecting these data. Based on the analyses of these data, the Intelligent Launcher was developed. An evaluation study involving 22 participants was conducted to test the Intelligent Launcher's accuracy.

The Intelligent Launcher (1) improved the prediction accuracy of the Drawing gesture by personalizing the target-matching template, (2) extended the prediction of the Letter gesture by adding to the "Name Library" a "Function Library" based on Google Play, and (3) filled in the gap of *Abstract* gesture predictions by using the App Usage Frequency history information. Our evaluation study showed that the Intelligent Launcher was able to achieve a high level of prediction accuracy (96%) and a satisfactory user experience.

To improve the correctness of the predictions, Chapter 6 focused on exploring the relationship between the factors and the gesture shape, and propose a mixed sources-gesture-app prediction method based on our findings. This was motivated by Gesture Marks, which predicts the target app for handwriting and non-handwriting gestures via different sources (handwriting recogniser vs. predefined template). This confirms that before prediction, it is important to attribute the gesture to a particular (accurate) source. Differing from Gesture Marks (Heuwing et al. 2015), which infers the gesture relying on another user's gesture template or on a handwriting- app's name recognition, Chapter 6 proposed the approach of predicting the gesture by coordinating sources, such as the app's name, icon or function. In particular, to determine the source used for prediction, we examined these factors and how they might influence the gesture.

Both the "PGS" and the "Intelligent Launcher" opened a new way of quickly accessing

apps on mobile devices. This thesis has contributed to the body of HCI research by (1) conducting an experiment to test the users' performance with the proposed methods; (2) providing new gesture data collection methods (i.e., Agile Search and Agile Gesture); (3) analysing the data and proposing guidance towards improving current touch gesture shortcut methods; (4) proposing innovative AI methods to predict target apps for fast access (i.e., the Intelligent Launcher); and (5) demonstrating a way of improving target app prediction by considering a user's preference.

## 7.2 Future direction

The main challenge of PGS is to deal with gesture recall errors. To understand how accessing failures occurred in the gesture recall tests, we identified and classified the main sources of recall failures in our data. This should provide useful information for future improvements to the proposed gesture method. For example, because gestures classified as *Abstract* are more likely to result in poor memorisation, gesture-based methods that discourage users from using *Abstract* gestures as shortcut gestures could be developed.

Future research should aim to improve the accuracy of gesture recall. This could be done by offering informed feedback (i.e., the gesture creation reason) at the stage of gesture personalization, as well as by developing more flexibility (e.g., allowing users to choose their gesture creation reason rather than predict it from the users' profile) at the stage of gesture recall.

The main challenge of the intelligent app predictor we have proposed is how well it deals with app prediction accuracy. The predictor does not need to endure the setback of recall accuracy on predefined gestures. However, it inevitably has to suffer from a prediction error problem.

A common feature in most of these prediction approaches is that they rely on a gesture-app template. Gesture Marks allows for gestures to be used without having to go through the initial learning stage and extending the gesture-app template through the "wisdom of the crowd" (Ouyang and Li 2012). It builds the template based on a pre-collected gesture-app library from the population of users combined with the handwriting recogniser (Bahlmann et al. 2002; Connell and Jain 2001; Hu et al. 2000). There-

fore, it can infer the meaning of gestures that a user has never defined themselves based on what other users have defined or what is in the handwriting recogniser. Even though, among a corrected average predicting a total of 77%, the handwriting recogniser contributed (44%) more than inferring from the non-handwriting template (33%), non-handwriting gestures were used more for accessing an app (40%) than a website (14%). This means the figures for correct prediction for accessing apps should be lower. One possible reason for the low correctness is that the diversity app icons, which users are constantly exposed to, have inspired them to use non-handwriting gestures on an individual basis, and only 21% of the gestures could be predicted by using another users' template (Ouyang and Li 2012). Another possible reason is that the handwriting gestures are not always created based on an app's name; sometimes they are created because of an app's function (Zhang et al. 2016).

In addition to this, several pieces of research have confirmed that users use different shaped gestures to access an app (Wobbrock et al. 2009; Nacenta et al. 2013; Poppinga et al. 2014). What causes this difference in users' choice? Previous work has pointed out that users defined their gestures differently based on corresponding features: for example, a letter-like gesture created based on an app's name (Lü et al. 2014a) or a drawing-shaped gesture created based on an app's icon (Zhang 2016). Users create different shapes of gestures for a variety of reasons (Poppinga et al. 2014) and they do not always create the gesture for the same reasons (Anthony et al. 2013).

In Chapter 6, the prediction accuracy was improved by learning from the user's profile. Chapter 6 predicted a link between the gesture creation reason and the gesture's shape. Knowing that means we can tell the predictor to predicate based on corresponding sources and for certain shapes. For example, the gesture "m" is a letter-shaped gesture which may rely on: (1) the *name*- the predictor should consider finding the target app among apps those with a letter "m" in their name, such as "Mail", "Messenger", etc.; (2) the *function*- the predictor should consider finding the target app among those apps which include the letter "m" in their function, such as, "Waze" (the map function), "Skype" (the message function), etc. However other information, such as last used app, users' age, etc. which have not been investigated, may be helpful for target app prediction as well. Therefore, it would be ideal if the future intelligent app predictor could get a better

understanding of the user.

Overall, it is obvious that both the PGS and our proposed methods can provide users with alternative ways of accessing apps. Future work can (1) check the accessing speed and users' satisfaction of both methods. Perhaps we can find better, more suitable solutions for each user with an app accessing problem; (2) extend the proposed methods to other mobile devices, such as tablets and smartwatches; (3) improve the accuracy of PGS recall by offering informed feedback (i.e., gesture creation reasons); (4) improve the Intelligent Launcher's predictions through a better understanding of how and why users create gestures in different shapes.

# Bibliography

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. et al., 2016. Tensorflow: a system for large-scale machine learning. *OSDI*, volume 16, 265–283.
- Acerio, A., Bernstein, N., Chambers, R., Ju, Y.-C., Li, X., Odell, J., Nguyen, P., Scholz, O. and Zweig, G., 2008. Live search for mobile: Web services by voice on the cellphone. *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, IEEE, 5256–5259.
- Ahlström, D., Hasan, K. and Irani, P., 2014. Are you comfortable doing that?: acceptance studies of around-device gestures in and for public settings. *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*, ACM, 193–202.
- Ahmad, B. I., Langdon, P. M., Godsill, S. J., Hardy, R., Dias, E. and Skrypchuk, L., 2014. Interactive displays in vehicles: Improving usability with a pointing gesture tracker and bayesian intent predictors. *Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, ACM, 1–8.
- Ahrens, W. H., Cox, D. J. and Budhwar, G., 1990. Use of the arcsine and square root transformations for subjectively determined percentage data. *Weed Science*, 38 (4-5), 452–458.
- Al Kabary, I. and Schuldt, H., 2014. Using hand gestures for specifying motion queries in sketch-based video retrieval. *European Conference on Information Retrieval*, Springer, 733–736.

- Albinsson, P.-A. and Zhai, S., 2003. High precision touch screen interaction. *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, 105–112.
- Alpern, M. and Minardo, K., 2003. Developing a car gesture interface for use as a secondary task. *CHI'03 extended abstracts on Human factors in computing systems*, ACM, 932–933.
- Android, G., 2019a. Android developer guide. <https://developer.android.com/training/gestures/multi.html>. [Online; accessed 19-July-2019].
- Android, G., 2019b. Android studio. <https://developer.android.com/studio>. [Online; accessed 19-July-2019].
- Anthony, L., Vatavu, R.-D. and Wobbrock, J. O., 2013. Understanding the consistency of users' pen and finger stroke gesture articulation. *Proceedings of Graphics Interface 2013*, Canadian Information Processing Society, 87–94.
- Anthony, L. and Wobbrock, J. O., 2010. A lightweight multistroke recognizer for user interface prototypes. *Proceedings of Graphics Interface 2010*, Canadian Information Processing Society, 245–252.
- Appert, C. and Zhai, S., 2009. Using strokes as command shortcuts: cognitive benefits and toolkit support. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2289–2298.
- Apple, 2015. Use Search on your iPhone, iPad, or iPod touch 7 perfection for your app. <https://support.apple.com/en-gb/HT201285>. [Online; accessed 19-July-2016].
- Apple, 2017. Use Search on your iPhone, iPad, or iPod touch. <https://support.apple.com/en-us/HT201285>. [Online; accessed 10-Oct-2017].
- Apple, 2018. Search with Spotlight. <https://support.apple.com/en-us/HT204014>. [Online; accessed 10-Aug-2018].
- Apple-siri, 2016. Use Siri on your iPhone, iPad, or iPod touch. <https://support.apple.com/en-gb/HT204389>. [Online; accessed 19-July-2016].
- Appsfire, 2015. Infographic: iOS Apps vs. Web apps. <http://blog.appsfire.com/infographic-ios-apps-vs-web-apps/>. [Online; accessed 19-July-2016].

- Arefin Shimon, S. S., Lutton, C., Xu, Z., Morrison-Smith, S., Boucher, C. and Ruiz, J., 2016. Exploring non-touchscreen gestures for smartwatches. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ACM, 3822–3833.
- Ashcraft, M. H., 1989. *Human memory and cognition..* Scott, Foresman & Co.
- Avery, J., Malacria, S., Nancel, M., Casiez, G. and Lank, E., 2018. Introducing transient gestures to improve pan and zoom on touch surfaces. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ACM, 25.
- Baeza-Yates, R., Jiang, D., Silvestri, F. and Harrison, B., 2015. Predicting the next app that you are going to use. *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, ACM, 285–294.
- Bahlmann, C., Haasdonk, B. and Burkhardt, H., 2002. Online handwriting recognition with support vector machines—a kernel approach. *Frontiers in handwriting recognition, 2002. proceedings. eighth international workshop on*, IEEE, 49–54.
- Baudisch, P. and Chu, G., 2009. Back-of-device interaction allows creating very small touch devices. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 1923–1932.
- Bell, N. and Lindamood, P., 1991. *Visualizing and verbalizing: For language comprehension and thinking*. Academy of Reading Publications Paso Robles, CA.
- Benenson, Z., Gassmann, F. and Reinfelder, L., 2013. Android and ios users' differences concerning security and privacy. *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, ACM, 817–822.
- Bhuiyan, M. and Picking, R., 2009. Gesture-controlled user interfaces, what have we done and what's next. *Proceedings of the Fifth Collaborative Research Symposium on Security, E-Learning, Internet and Networking (SEIN 2009), Darmstadt, Germany*, 25–29.
- Bi, X. and Zhai, S., 2016. Ijqwerty: What difference does one key change make? gesture typing keyboard optimization bounded by one key position change from qwerty.

- Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ACM, 49–58.
- Billingshurst, S. S. and Vu, K.-P. L., 2015. Touch screen gestures for web browsing tasks. *Computers in Human Behavior*, 53, 71–81.
- Blackler, A., Popovic, V. and Mahar, D., 2010. Investigating users' intuitive interaction with complex artefacts. *Applied ergonomics*, 41 (1), 72–92.
- Blasius, J. and Greenacre, M., 2014. *Visualization and verbalization of data*. CRC Press.
- Blodgett, J. G. and Anderson, R. D., 2000. A bayesian network model of the consumer complaint process. *Journal of Service Research*, 2 (4), 321–338.
- Böhmer, M., Hecht, B., Schöning, J., Krüger, A. and Bauer, G., 2011. Falling asleep with angry birds, facebook and kindle: a large scale study on mobile application usage. *Proceedings of the 13th international conference on Human computer interaction with mobile devices and services*, ACM, 47–56.
- Böhmer, M. and Krüger, A., 2013. A study on icon arrangement by smartphone users. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2137–2146.
- Bolt, R. A., 1980. *"Put-that-there": Voice and gesture at the graphics interface*, volume 14. ACM.
- Bolt, R. A. and Herranz, E., 1992. Two-handed gesture in multi-modal natural dialog. *Proceedings of the 5th annual ACM symposium on User interface software and technology*, ACM, 7–14.
- Boser, B. E., Guyon, I. M. and Vapnik, V. N., 1992. A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory*, ACM, 144–152.
- Bowden<sup>12</sup>, R., Zisserman, A., Kadir, T. and Brady, M., 2003. Vision based interpretation of natural sign languages.



- Bowen, K. and Pistilli, M. D., 2012. Student preferences for mobile app usage. *Research Bulletin*(Louisville, CO: EDUCAUSE Center for Applied Research, forthcoming), available from <http://www.educause.edu/ecar>.
- Bozkurt, E., Erzin, E. and Yemez, Y., 2015. Affect-expressive hand gestures synthesis and animation. *2015 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 1–6.
- Bragdon, A., Nelson, E., Li, Y. and Hinckley, K., 2011. Experimental analysis of touch-screen gesture designs in mobile environments. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 403–412.
- Bulwer, J., 1975. *Chirologia, or the natural language of the hand*.
- Cano, R., Sordo, C. and Gutiérrez, J. M., 2004. Applications of bayesian networks in meteorology. *Advances in Bayesian networks*, Springer, 309–328.
- Carlson, N., 2010. And Boy Have We Patented It. <http://www.businessinsider.com/and-boy-have-we-patented-it-2010-3?IR=T>. [Online; accessed 19-July-2016].
- Carrascal, J. P. and Church, K., 2015. An in-situ study of mobile app & mobile search interactions. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ACM, 2739–2748.
- Chaudhary, A., Raheja, J. L., Das, K. and Raheja, S., 2013. Intelligent approaches to interact with machines using hand gesture recognition in natural way: a survey. *arXiv preprint arXiv:1303.2292*.
- Chen, C.-C., 2015. User recognition and preference of app icon stylization design on the smartphone. *International Conference on Human-Computer Interaction*, Springer, 9–15.
- Chen, Q., Georganas, N. D. and Petriu, E. M., 2007. Real-time vision-based hand gesture recognition using haar-like features. *2007 IEEE Instrumentation & Measurement Technology Conference IMTC 2007*, IEEE, 1–6.
- Childers, T. L. and Houston, M. J., 1984. Conditions for a picture-superiority effect on consumer memory. *Journal of consumer research*, 11 (2), 643–654.

- Chong, M. K. and Marsden, G., 2009. Exploring the use of discrete gestures for authentication. *IFIP Conference on Human-Computer Interaction*, Springer, 205–213.
- Chong, M. K., Marsden, G. and Gellersen, H., 2010. Gesturepin: using discrete gestures for associating mobile devices. *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, ACM, 261–264.
- Connell, S. D. and Jain, A. K., 2001. Template-based online character recognition. *Pattern Recognition*, 34 (1), 1–14.
- de Cristo, M. A. P., Calado, P. P., Da Silveira, M. d. L., Silva, I., Muntz, R. and Ribeiro-Neto, B., 2003. Bayesian belief networks for ir. *International Journal of Approximate Reasoning*, 34 (2-3), 163–179.
- Cuccurullo, S., Francese, R., Murad, S., Passero, I. and Tucci, M., 2012. A gestural approach to presentation exploiting motion capture metaphors. *Proceedings of the International Working Conference on Advanced Visual Interfaces*, ACM, 148–155.
- Dancu, A., Vechev, V., Ünlüer, A. A., Nilson, S., Nygren, O., Eliasson, S., Barjonet, J.-E., Marshall, J. and Fjeld, M., 2015. Gesture bike: examining projection surfaces and turn signal systems for urban cycling. *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*, ACM, 151–159.
- Dengel, A., Agne, S., Klein, B., Ebert, A. and Deller, M., 2006. Human-centered interaction with documents. *Proceedings of the 1st ACM international workshop on Human-centered multimedia*, ACM, 35–44.
- Derpanis, K. G., 2004. A review of vision-based hand gestures. *Unpublished. Feb.*
- Dingler, T., Rzayev, R., Shirazi, A. S. and Henze, N., 2018. Designing consistent gestures across device types: Eliciting rsvp controls for phone, watch, and glasses. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ACM, 419.
- Dittmar, T., Krull, C. and Horton, G., 2015. A new approach for touch gesture recognition: Conversive hidden non-markovian models. *Journal of Computational Science*, 10, 66–76.

- Ecker, R., Broy, V., Butz, A. and De Luca, A., 2009. pietouch: a direct touch gesture interface for interacting with in-vehicle information systems. *Proceedings of the 11th international Conference on Human-Computer interaction with Mobile Devices and Services*, ACM, 22.
- Efron, D., 1941. Gesture and environment.
- eMarket, 2015. How Many Apps Do Smartphone Owners Use? <https://www.emarketer.com/Article/How-Many-Apps-Do-Smartphone-Owners-Use/1013309>. [Online; accessed 10-Oct-2017].
- ENDERLE, R., 2012. Pixelsense: The Not-So-Secret Advantage Of The Microsoft Surface Tablet. <http://www.digitaltrends.com/opinion/pixelsense-the-not-so-secret-advantage-of-the-microsoft-surface-tablet/>. [Online; accessed 19-July-2016].
- Epps, J., Lichman, S. and Wu, M., 2006. A study of hand shape use in tabletop gesture interaction. *CHI'06 extended abstracts on human factors in computing systems*, ACM, 748–753.
- Erol, A., Bebis, G., Nicolescu, M., Boyle, R. D. and Twombly, X., 2007. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding*, 108 (1), 52–73.
- Escalera, S., Athitsos, V. and Guyon, I., 2017. Challenges in multi-modal gesture recognition. *Gesture Recognition*, Springer, 1–60.
- Fahn, C.-S. and Sun, H., 2005. Development of a data glove with reducing sensors based on magnetic induction. *IEEE Transactions on Industrial Electronics*, 52 (2), 585–594.
- Falaki, H., Mahajan, R., Kandula, S., Lymberopoulos, D., Govindan, R. and Estrin, D., 2010. Diversity in smartphone usage. *Proceedings of the 8th international conference on Mobile systems, applications, and services*, ACM, 179–194.
- Fang, Y., Wang, K., Cheng, J. and Lu, H., 2007. A real-time hand gesture recognition method. *2007 IEEE International Conference on Multimedia and Expo*, IEEE, 995–998.

- Fehske, A., Fettweis, G., Malmodin, J. and Biczok, G., 2011. The global footprint of mobile communications: The ecological and economic perspective. *IEEE Communications Magazine*, 49 (8), 55–62.
- Fels, S. S. and Hinton, G. E., 1993. Glove-talk: A neural network interface between a data-glove and a speech synthesizer. *IEEE transactions on Neural Networks*, 4 (1), 2–8.
- Felt, A. P., Finifter, M., Chin, E., Hanna, S. and Wagner, D., 2011. A survey of mobile malware in the wild. *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, ACM, 3–14.
- Feng, T., Yang, J., Yan, Z., Tapia, E. M. and Shi, W., 2014. Tips: Context-aware implicit user identification using touch screen in uncontrolled environments. *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, ACM, 9.
- Findlater, L., Lee, B. and Wobbrock, J., 2012. Beyond qwerty: augmenting touch screen keyboards with multi-touch gestures for non-alphanumeric input. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2679–2682.
- Fitz-Walter, Z., Jones, S. and Tjondronegoro, D., 2008. Detecting gesture force peaks for intuitive interaction. *Proceedings of the 5th Australasian Conference on Interactive Entertainment*, ACM, 2.
- Fitzmaurice, G. W., Zhai, S. and Chignell, M. H., 1993. Virtual reality for palmtop computers. *ACM Transactions on Information Systems (TOIS)*, 11 (3), 197–218.
- Foley, J., van Dam, A., Feiner, S. and Hughes, J., 1996. The form and content of user-computer dialogues. *Computer Graphics: Principles and Practice*, 392–395.
- Freeman, W. T. and Roth, M., 1995. Orientation histograms for hand gesture recognition. *International workshop on automatic face and gesture recognition*, volume 12, 296–301.
- Frisch, M., Heydekorn, J. and Dachsel, R., 2009. Investigating multi-touch and pen gestures for diagram editing on interactive surfaces. *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ACM, 149–156.

- Furbach, U. and Maron, M., 2013. Nui-based floor navigation—a case study. *International Conference of Design, User Experience, and Usability*, Springer, 270–279.
- Gao, Y., Bianchi-Berthouze, N. and Meng, H., 2012. What does touch tell us about emotions in touchscreen-based gameplay? *ACM Transactions on Computer-Human Interaction (TOCHI)*, 19 (4), 31.
- Garg, P., Aggarwal, N. and Sofat, S., 2009. Vision based hand gesture recognition. *World Academy of Science, Engineering and Technology*, 49 (1), 972–977.
- Gerpott, T. J., Thomas, S. and Weichert, M., 2013. Characteristics and mobile internet use intensity of consumers with different types of advanced handsets: An exploratory empirical study of iphone, android and other web-enabled mobile users in germany. *Telecommunications Policy*, 37 (4-5), 357–371.
- Gessler, S. and Kotulla, A., 1995. Pdas as mobile www browsers. *Computer networks and ISDN Systems*, 28 (1), 53–59.
- Gillespie, M., James, A. N., Federmeier, K. D. and Watson, D. G., 2014. Verbal working memory predicts co-speech gesture: Evidence from individual differences. *Cognition*, 132 (2), 174–180.
- Good, M. D., Whiteside, J. A., Wixon, D. R. and Jones, S. J., 1984. Building a user-derived interface. *Communications of the ACM*, 27 (10), 1032–1043.
- Google, 2015a. Google Now. The right information at just the right time. <https://www.google.co.uk/landing/now/>. [Online; accessed 19-July-2016].
- Google, 2015b. Select a category for your app or game. <https://support.google.com/googleplay/android-developer/answer/113475?hl=en-GB>. [Online; accessed 19-July-2016].
- Google, 2018a. Google Photos. <https://play.google.com/store/apps/details?id=com.google.android.apps.photos>. [Online; accessed 10-Aug-2018].
- Google, 2018b. Human interface guidelines. <https://developer.apple.com/ios/human-interface-guidelines/overview/themes/>. [Online; accessed 10-Feb-2018].

- Google, 2018c. Material Design. <https://material.io/guidelines/>. [Online; accessed 10-Feb-2018].
- Google, 2018d. Select a category for your app or game. <https://support.google.com/googleplay/android-developer/answer/113475?hl=en>. [Online; accessed 10-Aug-2018].
- Gordon, M., Ouyang, T. and Zhai, S., 2016. Watchwriter: tap and gesture typing on a smartwatch miniature keyboard with statistical decoding. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ACM, 3817–3821.
- Gould, J. D. and Salaun, J., 1987. Behavioral experiments on handmarkings. *ACM Transactions on Information Systems (TOIS)*, 5 (4), 358–377.
- Grainger, J., Bertrand, D., Lété, B., Beyersmann, E. and Ziegler, J. C., 2016. A developmental investigation of the first-letter advantage. *Journal of Experimental Child Psychology*, 152, 161–172.
- Guy, I., 2016. Searching by talking: Analysis of voice queries on mobile web search. *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, ACM, 35–44.
- Han, J., 2006. The radical promise of the multi-touch interface.
- Hasanuzzaman, M., Ampornaramveth, V., Zhang, T., Bhuiyan, M., Shirai, Y. and Ueno, H., 2004. Real-time vision-based gesture recognition for human robot interaction. *Robotics and Biomimetics, 2004. ROBIO 2004. IEEE International Conference on*, IEEE, 413–418.
- Heo, S., Gu, J. and Lee, G., 2014. Expanding touch input vocabulary by using consecutive distant taps. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2597–2606.
- Heo, S. and Lee, G., 2011. Forcetap: extending the input vocabulary of mobile touch screens by adding tap gestures. *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, ACM, 113–122.

- Herkenrath, G., Karrer, T. and Borchers, J., 2008. Twend: twisting and bending as new interaction gesture in mobile devices. *CHI'08 Extended Abstracts on Human Factors in Computing Systems*, ACM, 3819–3824.
- Herzog, D., Massoud, H. and Wörndl, W., 2017. Routeme: A mobile recommender system for personalized, multi-modal route planning. *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, ACM, 67–75.
- Heuwing, B., Köller, I., Schanz, V. and Mandl, T., 2015. Usability of gesture-based mobile applications for first-time use. *Mensch und Computer 2015–Proceedings*.
- Hodgkins, K., 2015. Average iPhone owner has 108 apps, spends 84 minutes a day using them. <https://www.engadget.com/2011/01/28/average-iphone-owner-spends-84-minutes-a-day-using-the-108-apps/>. [Online; accessed 19-July-2016].
- Hopwood, S., 2017. Report: How Many Mobile Apps are Actually Used? <https://www.apptentive.com/blog/2017/06/22/how-many-mobile-apps-are-actually-used/>. [Online; accessed 10-Oct-2017].
- Hu, J., Lim, S. G. and Brown, M. K., 2000. Writer independent on-line handwriting recognition using an hmm approach. *Pattern Recognition*, 33 (1), 133–147.
- Huang, D.-Y., Hu, W.-C. and Chang, S.-H., 2009. Vision-based hand gesture recognition using pca+ gabor filters and svm. *Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP'09. Fifth International Conference on*, IEEE, 1–4.
- Huang, K., Zhang, C., Ma, X. and Chen, G., 2012. Predicting mobile application usage using contextual information. *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ACM, 1059–1065.
- IDC, 2017. Worldwide smartphone OS Market share. <https://www.idc.com/promo/smartphone-market-share/os>. [Online; accessed 10-Oct-2017].
- Johnson, E., 1965. Touch display? a novel input/output device for computers. *Electronics Letters*, 8 (1), 219–220.

- Johnson, E., 1967. Touch displays: a programmed man-machine interface. *Ergonomics*, 10 (2), 271–277.
- Kane, S. K., Bigham, J. P. and Wobbrock, J. O., 2008. Slide rule: making mobile touch screens accessible to blind people using multi-touch interaction techniques. *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*, ACM, 73–80.
- Karam, M. et al., 2005. A taxonomy of gestures in human computer interactions.
- Kearl, M., 2016. 99 Need-To-Know Stats On The Mobile Customer. <https://www.braze.com/blog/mobile-customer-99-stats/>. [Online; accessed 19-July-2016].
- Kendon, A., Sebeok, T. A. and Umiker-Sebeok, J., 1981. *Nonverbal communication, interaction, and gesture: selections from Semiotica*, volume 41. Walter de Gruyter.
- Kettebekov, S. and Sharma, R., 2000. Understanding gestures in multimodal human computer interaction. *International Journal on Artificial Intelligence Tools*, 9 (02), 205–223.
- Kılıboz, N. Ç. and Güdükbay, U., 2015. A hand gesture recognition technique for human-computer interaction. *Journal of Visual Communication and Image Representation*, 28, 97–104.
- Kim, J.-H., Thang, N. D. and Kim, T.-S., 2009. 3-d hand motion tracking and gesture recognition using a data glove. *2009 IEEE International Symposium on Industrial Electronics*, IEEE, 1013–1018.
- Kim, J.-W., Nam, T.-J. and Park, T., 2017. Compositegesture: Creating custom gesture interfaces with multiple mobile or wearable devices. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 11 (1), 77–82.
- Kipp, M., Neff, M., Kipp, K. H. and Albrecht, I., 2007. Towards natural gesture synthesis: Evaluating gesture units in a data-driven approach to gesture synthesis. *International Workshop on Intelligent Virtual Agents*, Springer, 15–28.
- Koons, D. B. and Sparrell, C. J., 1994. Iconic: speech and depictive gestures at the human-machine interface. *Conference companion on Human factors in computing systems*, ACM, 453–454.



- Kray, C., Nesbitt, D., Dawson, J. and Rohs, M., 2010. User-defined gestures for connecting mobile phones, public displays, and tabletops. *Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, ACM, 239–248.
- Krueger, M. W., 1983. Artificial reality.
- Kubo, Y., Shizuki, B. and Tanaka, J., 2016. B2b-swipe: Swipe gesture for rectangular smartwatches from a bezel to a bezel. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ACM, 3852–3856.
- Kuribara, T., Yoshikawa, T., Shizuki, B. and Tanaka, J., 2014. Handyscope: a remote control technique using pull-out gesture. *CHI'14 Extended Abstracts on Human Factors in Computing Systems*, ACM, 175–176.
- Lavid Ben Lulu, D. and Kuflik, T., 2013. Functionality-based clustering using short textual description: helping users to find apps installed on their mobile device. *Proceedings of the 2013 international conference on Intelligent user interfaces*, ACM, 297–306.
- Lee, H., 2016. Smartui: Resource-saving and editable smartphone user interface for fast remote pc control. *IEICE TRANSACTIONS on Information and Systems*, 99 (7), 1852–1861.
- Lemort, A. and Vales, S., 2013. Continuous recognition of multi-touch gestures. US Patent 8,390,577.
- Lewe, J.-H., 2002. A spotlight search method for multicriteria optimization problems. *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, GA*.
- Li, Y., 2010a. Gesture search: a tool for fast mobile data access. *Proceedings of the 23rd annual ACM symposium on User interface software and technology*, ACM, 87–96.
- Li, Y., 2010b. Protractor: a fast and accurate gesture recognizer. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2169–2172.
- Li, Y., 2012. Gesture-based interaction: a new dimension for mobile user interfaces. *Proceedings of the International Working Conference on Advanced Visual Interfaces*, ACM, 6–6.

- Liao, C., Guimbretière, F., Hinckley, K. and Hollan, J., 2008. Papiercraft: A gesture-based command system for interactive paper. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 14 (4), 18.
- Liao, Z.-X., Lei, P.-R., Shen, T.-J., Li, S.-C. and Peng, W.-C., 2012. Mining temporal profiles of mobile applications for usage prediction. *Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on*, IEEE, 890–893.
- Lin, S.-Y., Shie, C.-K., Chen, S.-C. and Hung, Y.-P., 2013. Airtouch panel: a re-anchorable virtual touch panel. *Proceedings of the 21st ACM international conference on Multimedia*, ACM, 625–628.
- Linjama, J. and Kaaresoja, T., 2004. Novel, minimalist haptic gesture interaction for mobile devices. *Proceedings of the third Nordic conference on Human-computer interaction*, ACM, 457–458.
- Lipsman, A., 2017. Mobile Matures as the Cross-Platform Era Emerges. <http://www.comscore.com/Insights/Blog/Mobile-Matures-as-the-Cross-Platform-Era-Emerges>. [Online; accessed 10-July-2017].
- Liu, Y., Li, F., Guo, L., Shen, B. and Chen, S., 2013. A comparative study of android and ios for accessing internet streaming services. *International Conference on Passive and Active Network Measurement*, Springer, 104–114.
- Lü, H., Fogarty, J. A. and Li, Y., 2014a. Gesture script: recognizing gestures and their structure using rendering scripts and interactively trained parts. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 1685–1694.
- Lü, H. and Li, Y., 2011. Gesture avatar: a technique for operating mobile user interfaces using gestures. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 207–216.
- Lü, H. and Li, Y., 2012. Gesture coder: a tool for programming multi-touch gestures by demonstration. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2875–2884.

- Lu, H. and Li, Y., 2015. Gesture on: Enabling always-on touch gestures for fast mobile access from the device standby mode. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ACM, 3355–3364.
- Lü, H., Negulescu, M. and Li, Y., 2014b. Gesturermote: interacting with remote displays through touch gestures. *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*, ACM, 325–328.
- Lulu, D. L. B. and Kuflik, T., 2015. Wise mobile icons organization: Apps taxonomy classification using functionality mining to ease apps finding. *Mobile Information Systems*.
- Madhvanath, S., Mandalapu, D., Madan, T., Rao, N. and Kozhissery, R., 2012. Gecco: Finger gesture-based command and control for touch interfaces. *Intelligent Human Computer Interaction (IHCI), 2012 4th International Conference on*, IEEE, 1–6.
- Malima, A., Ozgur, E. and Çetin, M., 2006. A fast algorithm for vision-based hand gesture recognition for robot control. *2006 IEEE 14th Signal Processing and Communications Applications*, IEEE, 1–4.
- Malloch, J., Griggio, C. F., McGrenere, J. and Mackay, W. E., 2017. Fieldward and pathward: Dynamic guides for defining your own gestures. *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ACM, 4266–4277.
- Mäntyjärvi, J., Kela, J., Korpipää, P. and Kallio, S., 2004. Enabling fast and effortless customisation in accelerometer based gesture interaction. *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, ACM, 25–31.
- Margaritis, D., 2003. Learning bayesian network model structure from data. Technical report, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science.
- Marshall, J. and Benford, S., 2011. Using fast interaction to create intense experiences. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 1255–1264.
- McCafferty, S. G. and Stam, G., 2009. *Gesture: Second Language Acquisition and Classroom Research*. Routledge.

- Mcdougall, S. J., Curry, M. B. and de Bruijn, O., 1999. Measuring symbol and icon characteristics: Norms for concreteness, complexity, meaningfulness, familiarity, and semantic distance for 239 symbols. *Behavior Research Methods, Instruments, & Computers*, 31 (3), 487–519.
- McNeill, D., 1992. *Hand and mind: What gestures reveal about thought*. University of Chicago press.
- McNeill, D., 2000. *Language and gesture*, volume 2. Cambridge University Press.
- Mehta, N., 1982. A flexible machine interface. *MA Sc. Thesis, Department of Electrical Engineering, University of Toronto supervised by Professor KC Smith*.
- Minsky, M. R., 1984. Manipulating simulated objects with real-world gestures using a force and position sensitive screen. *ACM SIGGRAPH Computer Graphics*, ACM, volume 18, 195–203.
- Mitra, S. and Acharya, T., 2007. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37 (3), 311–324.
- Morrel-Samuels, P. and Krauss, R. M., 1992. Word familiarity predicts temporal asynchrony of hand gestures and speech. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 18 (3), 615.
- Morris, M. R., Danielescu, A., Drucker, S., Fisher, D., Lee, B., Wobbrock, J. O. et al., 2014. Reducing legacy bias in gesture elicitation studies. *interactions*, 21 (3), 40–45.
- Morris, M. R., Wobbrock, J. O. and Wilson, A. D., 2010. Understanding users' preferences for surface gestures. *Proceedings of graphics interface 2010*, Canadian Information Processing Society, 261–268.
- Moyle, M. and Cockburn, A., 2003. The design and evaluation of a flick gesture for 'back' and 'forward' in web browsers. *Proceedings of the Fourth Australasian user interface conference on User interfaces 2003-Volume 18*, Australian Computer Society, Inc., 39–46.
- Murthy, G. and Jadon, R., 2009. A review of vision based hand gestures recognition.

- International Journal of Information Technology and Knowledge Management*, 2 (2), 405–410.
- Nacenta, M. A., Kamber, Y., Qiang, Y. and Kristensson, P. O., 2013. Memorability of pre-designed and user-defined gesture sets. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 1099–1108.
- Naumann, A., Hurtienne, J., Israel, J. H., Mohs, C., Kindsmüller, M. C., Meyer, H. A. and Hußlein, S., 2007. Intuitive use of user interfaces: defining a vague concept. *International Conference on Engineering Psychology and Cognitive Ergonomics*, Springer, 128–136.
- Ng, C. W. and Ranganath, S., 2002. Real-time gesture recognition system and application. *Image and Vision computing*, 20 (13), 993–1007.
- Nielsen, M., Störring, M., Moeslund, T. B. and Granum, E., 2003. A procedure for developing intuitive and ergonomic gesture interfaces for hci. *International Gesture Workshop*, Springer, 409–420.
- North, C., Dwyer, T., Lee, B., Fisher, D., Isenberg, P., Robertson, G. and Inkpen, K., 2009. Understanding multi-touch manipulation for surface computing. *IFIP Conference on Human-Computer Interaction*, Springer, 236–249.
- Obama, B., 2008. A more perfect union. Video. URL <http://video.google.com/videoplay?docid=6528042696351994555>.
- Ofcom, 2015. Ofcom 2015 Communications Market Report. [http://stakeholders.ofcom.org.uk/binaries/research/cmr/cmr15/CMR\\_UK\\_2015.pdf](http://stakeholders.ofcom.org.uk/binaries/research/cmr/cmr15/CMR_UK_2015.pdf). [Online; accessed 19-July-2016].
- Oh, U. and Findlater, L., 2013. The challenges and potential of end-user gesture customization. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 1129–1138.
- Okpo, J., Masthoff, J., Dennis, M., Beacham, N. and Ciocarlan, A., 2017. Investigating the impact of personality and cognitive efficiency on the selection of exercises for learners.

- Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, ACM, 140–147.
- Ouyang, T. and Li, Y., 2012. Bootstrapping personal gesture shortcuts with the wisdom of the crowd and handwriting recognition. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2895–2904.
- Ouyang, T. Y. and Davis, R., 2009. A visual approach to sketched symbol recognition. *IJCAI*, volume 9, 1463–1468.
- Paek, T., Thiesson, B., Ju, Y.-C. and Lee, B., 2008. Search vox: Leveraging multimodal refinement and partial knowledge for mobile voice search. *Proceedings of the 21st annual ACM symposium on User interface software and technology*, ACM, 141–150.
- Paivio, A. and Csapo, K., 1973. Picture superiority in free recall: Imagery or dual coding? *Cognitive psychology*, 5 (2), 176–206.
- Palomares, J., Loaiza, M. and Raposo, A., 2014. Study and evaluation of separability techniques and occlusion in multitouch surfaces. *International Conference on Human-Computer Interaction*, Springer, 293–304.
- Parate, A., Böhmer, M., Chu, D., Ganesan, D. and Marlin, B. M., 2013. Practical prediction and prefetch for faster access to applications on mobile phones. *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, ACM, 275–284.
- Pavlovic, V. I., Sharma, R. and Huang, T. S., 1997. Visual interpretation of hand gestures for human-computer interaction: A review. *IEEE Transactions on pattern analysis and machine intelligence*, 19 (7), 677–695.
- Perez, S., 2017. Report: Smartphone owners are using 9 apps per day, 30 per month. <https://techcrunch.com/2017/05/04/report-smartphone-owners-are-using-9-apps-per-day-30-per-month/>. [Online; accessed 10-Oct-2017].
- Pirhonen, A., Brewster, S. and Holguin, C., 2002. Gestural and audio metaphors as a

- means of control for mobile devices. *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, 291–298.
- Plamondon, R. and Srihari, S. N., 2000. Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on pattern analysis and machine intelligence*, 22 (1), 63–84.
- Play, G., 2018. Gesture Magic. <https://play.google.com/store/apps/details?id=com.gesture.action>. [Online; accessed 20-July-2018].
- Pol, M., 2015. *App icon preferences: The influence of app icon design and involvement on quality and intention to download*. Master's thesis, University of Twente.
- Poppinga, B., Sahami Shirazi, A., Henze, N., Heuten, W. and Boll, S., 2014. Understanding shortcut gestures on mobile touch devices. *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*, ACM, 173–182.
- Poupyrev, I. and Maruyama, S., 2003. Tactile interfaces for small touch screens. *Proceedings of the 16th annual ACM symposium on User interface software and technology*, ACM, 217–220.
- Quek, F., McNeill, D., Bryll, R., Duncan, S., Ma, X.-F., Kirbas, C., McCullough, K. E. and Ansari, R., 2002. Multimodal human discourse: gesture and speech. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 9 (3), 171–193.
- Quek, F. K., 1994. Toward a vision-based hand gesture interface. *Virtual Reality Software and Technology Conference*, volume 94, 17–29.
- Raptis, G. E., Katsini, C., Belk, M., Fidas, C., Samaras, G. and Avouris, N., 2017. Using eye gaze data and visual activities to infer human cognitive styles: method and feasibility studies. *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, ACM, 164–173.
- Rautaray, S. S. and Agrawal, A., 2015. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, 43 (1), 1–54.

- Rekik, Y., Vatavu, R.-D. and Grisoni, L., 2014. Understanding users' perceived difficulty of multi-touch gesture articulation. *Proceedings of the 16th International Conference on Multimodal Interaction*, ACM, 232–239.
- Rekimoto, J., 2001. Gesturewrist and gesturepad: Unobtrusive wearable interaction devices. *Wearable Computers, 2001. Proceedings. Fifth International Symposium on*, IEEE, 21–27.
- Rekimoto, J., 2002. Smartskin: an infrastructure for freehand manipulation on interactive surfaces. *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, 113–120.
- Ren, Y., Li, Y. and Lank, E., 2014. Inkanchor: enhancing informal ink-based note taking on touchscreen mobile phones. *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, ACM, 1123–1132.
- Robinson, S., Eslambolchilar, P. and Jones, M., 2009. Sweep-shake: finding digital resources in physical environments. *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, ACM, 12.
- Rodriguez-Feliz, J. R. and Roth, M. Z., 2012. The mobile technology era: potential benefits and the challenging quest to ensure patient privacy and confidentiality. *Plastic and reconstructive surgery*, 130 (6), 1395–1397.
- Roudaut, A., Lecolinet, E. and Guiard, Y., 2009. Microrolls: expanding touch-screen input vocabulary by distinguishing rolls vs. slides of the thumb. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 927–936.
- Roy, Q., Malacria, S., Guiard, Y., Lecolinet, E. and Eagan, J., 2013. Augmented letters: mnemonic gesture-based shortcuts. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2325–2328.
- Rubine, D., 1992. Combining gestures and direct manipulation. *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, 659–660.
- Ruiz, J., Li, Y. and Lank, E., 2011. User-defined motion gestures for mobile interaction.



- Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 197–206.
- Russell, S. J. and Norvig, P., 2016. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.
- Rusňák, V., Appert, C., Chapuis, O. and Pietriga, E., 2018. Designing coherent gesture sets for multi-scale navigation on tabletops. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ACM, 142.
- Ryall, K., Morris, M. R., Everitt, K., Forlines, C. and Shen, C., 2006. Experiences with and observations of direct-touch tabletops. *First IEEE International Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP'06)*, IEEE, 8–pp.
- Sae-Bae, N., Memon, N., Isbister, K. and Ahmed, K., 2014. Multitouch gesture-based authentication. *IEEE transactions on information forensics and security*, 9 (4), 568–582.
- Sahami Shirazi, A., Henze, N., Dingler, T., Pielot, M., Weber, D. and Schmidt, A., 2014. Large-scale assessment of mobile notifications. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 3055–3064.
- Sanna, A., Lamberti, F., Paravati, G. and Manuri, F., 2013. A kinect-based natural interface for quadrotor control. *Entertainment Computing*, 4 (3), 179–186.
- Sanna, A., Lamberti, F., Paravati, G., Ramirez, E. A. H. and Manuri, F., 2011. A kinect-based natural interface for quadrotor control. *International Conference on Intelligent Technologies for Interactive Entertainment*, Springer, 48–56.
- Scaltritti, M. and Balota, D. A., 2013. Are all letters really processed equally and in parallel? further evidence of a robust first letter advantage. *Acta psychologica*, 144 (2), 397–410.
- Schalkwyk, J., Beeferman, D., Beaufays, F., Byrne, B., Chelba, C., Cohen, M., Kamvar, M. and Strobe, B., 2010. “your word is my command”: Google search by voice: A case study. *Advances in speech recognition*, Springer, 61–90.

- Schauerte, B. and Fink, G. A., 2010. Focusing computational visual attention in multimodal human-robot interaction. *International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction*, ACM, 6.
- Schoeffmann, K., 2014. The stack-of-rings interface for large-scale image browsing on mobile touch devices. *Proceedings of the 22nd ACM international conference on Multimedia*, ACM, 1097–1100.
- Schuler, D. and Namioka, A., 1993. *Participatory design: Principles and practices*. CRC Press.
- Scientist, J., 2009. The fountain of youth. Patent No. 12345, Filed July 1st., 2008, Issued Aug. 9th., 2009.
- Shahzad, M., Liu, A. X. and Samuel, A., 2013. Secure unlocking of mobile touch screen devices by simple gestures: you can see it but you can not do it. *Proceedings of the 19th annual international conference on Mobile computing & networking*, ACM, 39–50.
- Shen, S.-T., Owen, D. and Meen, T., 2016. People and their smartphones—mapping mobile interaction in the modern connected world. *Engineering Computations*, 33 (6).
- Shin, C., Hong, J.-H. and Dey, A. K., 2012. Understanding and prediction of mobile application usage for smart phones. *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, ACM, 173–182.
- Singh, I., Krishnamurthy, S. V., Madhyastha, H. V. and Neamtiu, I., 2015. Zapdroid: managing infrequently used applications on smartphones. *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, ACM, 1185–1196.
- Siong-Hoe, L. and Hui, L. T., 2012. Understanding the user acceptance of gesture-based human-computer interactions. *Computers & Informatics (ISCI), 2012 IEEE Symposium on*, IEEE, 7–10.
- Slideshare.net, 2017. Mobile Users Reach to Phone. [https://www.slideshare.net/kleinerperkins/kpcb-internet-trends-2013/52-Mobile\\_Users\\_Reach\\_to\\_Phone](https://www.slideshare.net/kleinerperkins/kpcb-internet-trends-2013/52-Mobile_Users_Reach_to_Phone). [Online; accessed 10-Oct-2017].

- Song, H., Benko, H., Guimbretiere, F., Izadi, S., Cao, X. and Hinckley, K., 2011. Grips and gestures on a multi-touch pen. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 1323–1332.
- Song, Y., Demirdjian, D. and Davis, R., 2012. Continuous body and hand gesture recognition for natural human-computer interaction. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2 (1), 5.
- Sonnhammer, E. L., Von Heijne, G., Krogh, A. et al., 1998. A hidden markov model for predicting transmembrane helices in protein sequences. *Ismb*, volume 6, 175–182.
- Statista, 2015a. Number of available apps in Apple app store from July 2008 to June 2015. <http://www.statista.com/statistics/263795/number-of-available-apps-in-the-apple-app-store/>. [Online; accessed 19-July-2016].
- Statista, 2015b. Number of available apps in Apple app store from July 2008 to June 2015. <http://www.statista.com/statistics/263795/number-of-available-apps-in-the-apple-app-store/>. [Online; accessed 19-July-2016].
- Statista, 2016a. Smart phone Users Worldwide 2014-2019. <http://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/>. [Online; accessed 19-July-2016].
- Statista, 2016b. Smart phone Users Worldwide 2014-2019. <http://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/>. [Online; accessed 19-July-2016].
- Stern, H. I., Wachs, J. P. and Edan, Y., 2008. Designing hand gesture vocabularies for natural interaction by combining psycho-physiological and recognition factors. *International Journal of Semantic Computing*, 2 (01), 137–160.
- Strachan, S., Mazoin, B. and Gimeno, A., 2010. Pot à musique: Tangible interaction with digital media. *CHI'10 Extended Abstracts on Human Factors in Computing Systems*, ACM, 3313–3318.

- Sturman, D. J. and Zeltzer, D., 1994. A survey of glove-based input. *IEEE Computer graphics and Applications*, 14 (1), 30–39.
- Sunburst, S., 2014. Samsung Sunburst. <http://www.cnet.com/products/samsung-sunburst-at-t/>. [Online; accessed 19-July-2016].
- Tan, C., Liu, Q., Chen, E. and Xiong, H., 2012. Prediction for mobile application usage patterns. *Nokia MDC Workshop*, volume 12.
- Tarchanidis, K. N. and Lygouras, J. N., 2003. Data glove with a force sensor. *IEEE Transactions on Instrumentation and Measurement*, 52 (3), 984–989.
- Thalheimer, W., 2010. How much do people forget. *Work-Learning Research, Inc. Somerville, MA*.
- Tochihara, N., Sato, T. and Koike, H., 2016. A remote pointing method with dynamic cd ratio during a pinching gesture for large tabletop systems. *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, ACM, 553–559.
- ToolkiT, G., 2009. Beyond pinch and flick: Enriching mobile gesture interaction.
- Tsianos, N., Germanakos, P., Lekkas, Z., Mourlas, C. and Samaras, G., 2009. Eye-tracking users' behavior in relation to cognitive style within an e-learning environment. *Advanced Learning Technologies, 2009. ICALT 2009. Ninth IEEE International Conference on*, IEEE, 329–333.
- Tu, H., Ren, X. and Zhai, S., 2012. A comparative evaluation of finger and pen stroke gestures. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 1287–1296.
- Ubhi, H. K., Kotz, D., Michie, S., van Schayck, O. C. P. and West, R., 2017. A comparison of the characteristics of ios and android users of a smoking cessation app. *Translational Behavioral Medicine*, 7 (2), 166–171. URL <http://dx.doi.org/10.1007/s13142-016-0455-z>.
- Ujima, K., Kadomura, A. and Siio, I., 2014. U-remo: projection-assisted gesture control

- for home electronics. *Proceedings of the extended abstracts of the 32nd annual ACM conference on Human factors in computing systems*, ACM, 1609–1614.
- Vaidyanathan, V. and Rosenberg, D., 2014. “will use it, because i want to look cool” a comparative study of simple computer interactions using touchscreen and in-air hand gestures. *International Conference on Human-Computer Interaction*, Springer, 170–181.
- Vatavu, R.-D., Anthony, L. and Wobbrock, J. O., 2014. Gesture heatmaps: Understanding gesture performance with colorful visualizations. *Proceedings of the 16th International Conference on Multimodal Interaction*, ACM, 172–179.
- Vatavu, R.-D., Vogel, D., Casiez, G. and Grisoni, L., 2011. Estimating the perceived difficulty of pen gestures. *IFIP Conference on Human-Computer Interaction*, Springer, 89–106.
- Vickers, A. J., 2005. Parametric versus non-parametric statistics in the analysis of randomized trials with non-normally distributed data. *BMC medical research methodology*, 5 (1), 35.
- Von Zezschwitz, E., De Luca, A., Brunkow, B. and Hussmann, H., 2015. Swipin: fast and secure pin-entry on smartphones. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ACM, 1403–1406.
- Wachs, J. P., Kölsch, M., Stern, H. and Edan, Y., 2011. Vision-based hand-gesture applications. *Communications of the ACM*, 54 (2), 60–71.
- Wang, C.-Y., Hsiu, M.-C., Chiu, P.-T., Chang, C.-H., Chan, L., Chen, B.-Y. and Chen, M. Y., 2015. Palmgesture: Using palms as gesture interfaces for eyes-free input. *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, ACM, 217–226.
- Werner, S., 2014. The steering wheel as a touch interface: Using thumb-based gesture interfaces as control inputs while driving. *Adjunct Proceedings of the 6th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, ACM, 1–4.

- Westerman, W., 1999. *Hand tracking, finger identification, and chordic manipulation on a multi-touch surface*. Ph.D. thesis, University of Delaware.
- Westerman, W., Elias, J. G. and Hedge, A., 2001. Multi-touch: A new tactile 2-d gesture interface for human-computer interaction. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, SAGE Publications, volume 45, 632–636.
- Wexelblat, A., 1995. An approach to natural gesture in virtual environments. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 2 (3), 179–200.
- Wigdor, D., Forlines, C., Baudisch, P., Barnwell, J. and Shen, C., 2007. Lucid touch: a see-through mobile device. *Proceedings of the 20th annual ACM symposium on User interface software and technology*, ACM, 269–278.
- Wigdor, D. and Wixon, D., 2011. *Brave NUI world: designing natural user interfaces for touch and gesture*. Elsevier.
- wikipedia, 2016. Recall (Memory). [https://en.wikipedia.org/wiki/Recall\\_\(memory\)](https://en.wikipedia.org/wiki/Recall_(memory)). [Online; accessed 19-July-2016].
- Wobbrock, J. O., Aung, H. H., Rothrock, B. and Myers, B. A., 2005. Maximizing the guessability of symbolic input. *CHI'05 extended abstracts on Human Factors in Computing Systems*, ACM, 1869–1872.
- Wobbrock, J. O., Morris, M. R. and Wilson, A. D., 2009. User-defined gestures for surface computing. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 1083–1092.
- Wobbrock, J. O., Wilson, A. D. and Li, Y., 2007. Gestures without libraries, toolkits or training: a \$1 recognizer for user interface prototypes. *Proceedings of the 20th annual ACM symposium on User interface software and technology*, ACM, 159–168.
- Wolf, C. G., 1986. Can people use gesture commands? *ACM SIGCHI Bulletin*, 18 (2), 73–74.
- Wolf, C. G. and Morrel-Samuels, P., 1987. The use of hand-drawn gestures for text editing. *International Journal of Man-Machine Studies*, 27 (1), 91–102.

- Wortham, J., 2013. Digital Diary: Are We Suffering From Mobile App Burnout? <https://bits.blogs.nytimes.com/2013/02/15/digital-diary-are-we-suffering-from-mobile-app-burnout/>. [Online; accessed 10-Oct-2017].
- Xia, H., Grossman, T. and Fitzmaurice, G., 2015. Nanostylus: Enhancing input on ultra-small displays with a finger-mounted stylus. *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, ACM, 447–456.
- Xu, Q., Eрман, J., Gerber, A., Mao, Z., Pang, J. and Venkataraman, S., 2011. Identifying diverse usage behaviors of smartphone apps. *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, ACM, 329–344.
- Xu, Y., Lin, M., Lu, H., Cardone, G., Lane, N., Chen, Z., Campbell, A. and Choudhury, T., 2013. Preference, context and communities: a multi-faceted approach to predicting smartphone app usage patterns. *Proceedings of the 2013 International Symposium on Wearable Computers*, ACM, 69–76.
- Yahoo, 2014. Android users have an average of 95 apps installed on their phones. <http://www.androidcentral.com/yahoo-aviate-data-shows-95-apps-are-installed-average-android-device>. [Online; accessed 19-July-2016].
- Yamato, J., Ohya, J. and Ishii, K., 1992. Recognizing human action in time-sequential images using hidden markov model. *Proceedings 1992 IEEE Computer Society conference on computer vision and pattern recognition*, IEEE, 379–385.
- Yan, T., Chu, D., Ganesan, D., Kansal, A. and Liu, J., 2012. Fast app launching for mobile devices using predictive user context. *Proceedings of the 10th international conference on Mobile systems, applications, and services*, ACM, 113–126.
- Yang, Y., Clark, G. D., Lindqvist, J. and Oulasvirta, A., 2016. Free-form gesture authentication in the wild. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ACM, 3722–3735.
- Yu, C., Sun, K., Zhong, M., Li, X., Zhao, P. and Shi, Y., 2016. One-dimensional hand-

- writing: Inputting letters and words on smart glasses. *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, ACM, 71–82.
- Zeiß, S., Marinc, A., Braun, A., Große-Puppendahl, T. and Beck, S., 2014. A gesture-based door control using capacitive sensors. *International Conference on Distributed, Ambient, and Pervasive Interactions*, Springer, 207–216.
- Zenebe, A., Zhou, L. and Norcio, A. F., 2010. User preferences discovery using fuzzy models. *Fuzzy Sets and Systems*, 161 (23), 3044–3063.
- Zhai, S., Kristensson, P. O., Appert, C., Andersen, T. H. and Cao, X., 2012. Foundational issues in touch-screen stroke gesture design-an integrative review. *Foundations and Trends in Human-Computer Interaction*, 5 (2), 97–205.
- Zhang, C., 2016. Agile search: a tool for searching apps easily on mobile by gesture. *Proceedings of the 30th International BCS Human Computer Interaction Conference: Fusion!*, BCS Learning & Development Ltd., 12.
- Zhang, C., 2017. Improving app look up speed on mobile via user-defined touch gesture. *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, ACM, 196–201.
- Zhang, C., Ding, X., Chen, G., Huang, K., Ma, X. and Yan, B., 2012. Nihao: A predictive smartphone application launcher. *International Conference on Mobile Computing, Applications, and Services*, Springer, 294–313.
- Zhang, C., Jiang, N. and Tian, F., 2016. Accessing mobile apps with user defined gesture shortcuts: An exploratory study. *Proceedings of the 2016 ACM on Interactive Surfaces and Spaces*, ACM, 385–390.
- Zhang, H. and Li, Y., 2014. Gestkeyboard: enabling gesture-based interaction on ordinary physical keyboard. *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, ACM, 1675–1684.
- Zhao, X., Feng, T. and Shi, W., 2013. Continuous mobile authentication using a novel graphic touch gesture feature. *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*, IEEE, 1–6.



Zimbra, D., Sarangee, K. R. and Jindal, R. P., 2017. Movie aspects, tweet metrics, and movie revenues: The influence of ios vs. android. *Decision Support Systems*, 102, 98–109.

Zong, S., Kveton, B., Berkovsky, S., Ashkan, A. and Wen, Z., 2017. Get to the bottom: Causal analysis for user modeling. *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, ACM, 256–264.

Zou, X., Zhang, W., Li, S. and Pan, G., 2013. Prophet: What app you wish to use next. *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, ACM, 167–170.