Improving App Look up speed on Mobile via User-defined Touch Gesture

Chi Zhang
Department of Creative
Technology
Bournemouth University
Bournemouth, BH12 5BB, UK
czhang@bournemouth.ac.uk

Abstract

Nowadays, users rely on the smartphones to do all sort of things more often. On average, a normal user accesses to his or her smartphone 150 times per day. These include accessing varieties of apps for different tasks. Accessing these apps via the traditional smartphone graphic interface, the user needs to navigate the UI hierarchy to look up the target app. However, with the increasing amount of apps installed, the looking up process takes a lot of time. Using gesture shortcuts directly to active an app can largely simplify the apps' look up process but its performances were not investigated by previous research. This study evaluated the speed and accuracy of accessing target app using the user-defined gesture shortcuts approach and the traditional graphic interface. The results suggest the user-defined gesture shortcuts approach can improve the app look up speed. Additionally, detail breakdowns of speed differences, as well as the advantages of each approach, were analyzed.

Author Keywords

Mobile Interface; User-Defined Gestures; App Finding; Touch Gestures

ACM Classification Keywords

H.5.m. Information interfaces and presentation (e.g., HCI): User Interfaces

Figure 1: Examples of normal smartphone screen that content 16 normal size app icons.

Introduction

The popularity of using variety of apps to support people's everyday life promotes the increasing of the installed apps' number in people's smartphones. On average there are around a hundred installed apps in people's smartphones [12]. According to the report given by Statista there are 2.1 billion smartphone users in 2016 [13] and half of them have forty to seventy installed apps in their smartphones [9]. However, a normal smartphone (take Android phone as example) can hold up to 16 normal size apps' icon in one single screen (Figure 1). That means more than one billion smartphone users (with more than 16 installed apps) may face the problem of navigating through screens to find out the target app as shown in Figure 2. Obviously, in case of only relying on the traditional smartphone graphic interface mentioned above to find a target app is very time-consuming and inconvenient [7] [17]. Moreover, the requirement of finding the target app is very often. Based on KPCB's report the smartphone users access installed apps 150 times per day [10]. Thus, it becomes crucial to provide new acceptable approach to save user's average app look up time.

Past approaches which addressed improving the app look up speed mainly included: 1) using machine learning methods to classify installed apps into functional-based categories [3]; 2) using context information to infer the next app that users may want to access [14][6][8]; 3) using crowd sourcing and personalized hand writing recognizer to predict a user's target app via arbitrage gesture shortcuts [15]. Such approaches provide a better way to help users find apps. Users can find the target app from a reduced apps' list that only content the apps that predictor thought most possible to be used by the machine learning algorithm. However, the effectiveness of these approaches all relies on the machine learning method's qualification. It would become less effective when the prediction algorithm could not figure out the candidate app correctly.

Recent research on touch screen gestures has acknowledged the convenience of using touch gesture shortcuts to access system functions as well as installed apps [4] [16]. Moreover, the gesture-app mapping rule is determined by system but not machine learning algorithm. The effectiveness is not influenced however by the qualification of the machine learning method. However, these approaches are akin to use touch gestures to input instead of inputting the corresponding character via keyboard. The input is limited to character-like gestures and the gesture-app mapping rule is determined by the system but not users. The users can only benefit from these approaches when they can at least clearly remember some of the textual information related to the target they want to access. For example, if the user wants to access the Gmail via Gesture Search or Gesture On, he or she needs to draw a character such as "G" or "M" etc., which is included in the word of "Gmail". Otherwise, if the user only has a vague impression on the target app's name and input a character which was not included in it, these approaches turn out to be less effective. It can be seen that two of primary critical problems which influence these approaches' effectiveness are 1) the users' memorability on the gestures and 2) the limitation of the gestures' format. One previous research confirmed that user-defined gesture shortcuts are easier to remember than system-defined ones with 24% higher recall rate [11]. Other research focused on understanding users' gesture for fast accessing commonly used system functions or apps shown that in addition to character-like gestures, drawing shape gestures were also adopted by users [1] [5].

Taking into account of the advantages and limitations of all these previous studies, this paper adopted the approach of using user-defined gestures shortcuts to access the target app. The approach gives the freedom to the user to define whatever gesture as shortcuts they like for the target app. Based on the previous research, there are benefits from at least two perspectives: on the one hand, user-defined gestures has better memorability performance than system pre-defined gestures, on the other hand the gesture that is defined by user is no longer limited to character format.

That said, previous research has already looked into the effectiveness of using user-defined gestures to fast access the frequently used apps [2]. Nevertheless, whether the approach will still be effective for infrequently used apps were not inspected. This paper addresses the following questions:

- Can the approach of using user-defined gesture shortcuts save the interaction time of looking up the target app on the mobile?

- Given the opportunity and freedom to access the app via self-defined gestures, can users find the apps that they frequently and infrequently use within the similar time and achieve the similar accuracy?

Methodology

The existing gesture-based approaches for helping users to find their required apps do not distinguish the apps based on usage frequency. The time users spend on recalling the gesture defined for frequently and infrequently used apps as well as the target app's look up time may differ due to the apps' using frequency and the time past since the initial definition. In order to get a convincing, reliable result, we tested the performance for accessing frequently and infrequently used apps separately after 24 hours and the following 7 days of the first execution.

Apparatus

The hardware is a Nexus 5 designed by Google (2.3 GHz processor with 2 GB of RAM). Agile Gesture is installed on the device for data collection. Agile Gesture is a standalone Android app that allows the user to define gesture shortcuts to launch associated apps. The gesture and the corresponding app are recorded by Agile Gesture. Therefore, after the gesture for the particular app is well defined by the user, he /she can access the target app by drawing the corresponding gesture. The Agile Gesture was developed using java for Android. The development environment was Apple's Android Studio 2.2. Figure 2 shows the work flow of Agile Gesture.

Participant

Thirty-six paid volunteer participants (15 female, 21 male) were recruited from the local university campus. Participants ranged from 18 to 36 years. All were daily users of smart-phones, reporting 0.5 to 6 hours usage per day (mean = 4.5, SD = 4.3), have 50 to 131 the third party installed apps at the time the experiment was conducted (mean = 84, SD = 21.7). Participants had no prior experience of similar approaches for launching installed apps.

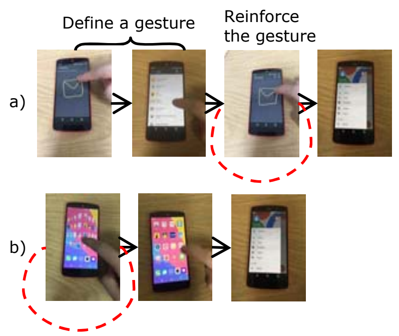

Figure 2: The work flow of finding the target app (Gmail) via a) Agile Gesture and b) Traditional graphic interface. The actions circled by the red dotted line needs to repeat several times.

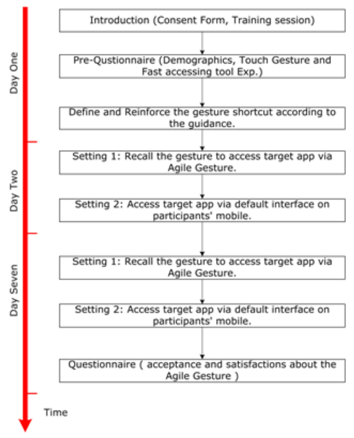
Introduction (Consent Form, Training session)
Pre-Qustionnaire (Demographics, Touch Gesture and Fast accessing tool Exp.)
Define and Reinforce the gesture shortcut according to the guidance.
Setting 1: Recall the gesture to access target app via Agile Gesture.
Setting 2: Access target app via default interface on participants' mobile.
Setting 1: Recall the gesture to access target app via Agile Gesture.
Setting 2: Access target app via default interface on participants' mobile.
Questionnaire (acceptance and satisfactions about the Agile Gesture)
Day One
Day Two
Day Seven
Time

Figure 3: Overall experiment procedure

	Order of tasks for gesture definition and recall	
Group A	Frequently used target app	Infrequently used target app
Group B	Infrequently used target apps	Frequently used target apps

Table 1: Group details for counterbalance

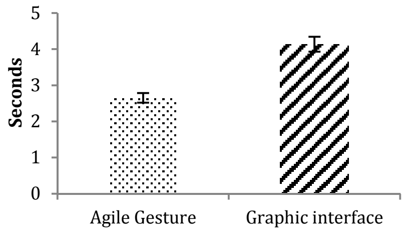

Figure 4: Overall app look up time cross different approaches

Experiment Design
The experiment was a 2*2 within-subjects design. There were two independent variables:

1. App usage frequency (Frequent used app, Infrequent used app)
2. Gesture recall interval (24 hours, 7 days)

The app usage frequency conditions were counterbalanced by switching the order of trials for frequently used apps and infrequently used apps. The experiments were conducted under two different settings: 1) accessing the target app via Agile Gesture and 2) via traditional graphic interface. The dependent variables were time spent for completing the accessed target app's task under both settings and the accuracy of the gesture recall tasks. The amount of trial was 2 app usage frequency* 2 gesture recall interval * 2 conditions* (6+7+8+9+10) number of testing app * 6 participants/number of testing app = 2160 trials.

Procedure
Eligible students for the experiment were asked to provide their screen-shots of their installed apps' list and their frequently used apps to the experimenter via email. Those who had been selected for the experiments were emailed and asked to come to the dedicated lab. 5 pounds were paid each. All participants were randomly divided into two groups, the details of the groups and the associated order for gesture definition and recall process shows in Table 1.

The experiment began with a training session. The training session included: 1) telling the participants how Agile gesture works; 2) giving participants time to familiarize themselves with the Agile Gesture. After the participants reported they were familiar with Agile Gesture they were then asked to complete the questionnaire which asked participants' demographics, gesture and app fast accessing tool's using experience. The first days' tasks were to define and reinforce the gesture shortcut according to the experimenter's guidance. The second and seventh day's tasks were to access the target apps via 1) the gesture they had defined in the gesture definition task by Agile Gesture; 2) system default interface by (see Figure 3). At last, participants' acceptance and satisfactions on Agile Gesture were collected via a questionnaire.

Results and Discussions

Overall performances on app look up time
The results of paired samples t-Test shown that overall using user-defined gesture shortcuts (M=2.65, SD=1.26) can significantly improve (P<0.05) the speed for looking up the target app than using traditional graphic interface (M=4.13, SD=2.89) (Figure 4).

Look up sped performances on different target apps across Agile Gesture and Graphic interface
The detail breakdowns of the speed performance for frequently and infrequently used apps are shown in Figure 5. According to the paired samples t-Test, there were no speed differences between using Agile Gesture and traditional graphic interface to access the target app. This may because the frequently used apps usually been put on the relative front page of screens, so that user do not need to transfer long to find them. However, for accessing the infrequently used apps the speed performance shown significant differences in using Agile Gesture and traditional graphic interface (P<0.05). Agile Gesture (M=2.95, SD=1.37) showed great advantages than graphic interface (M=5.49,

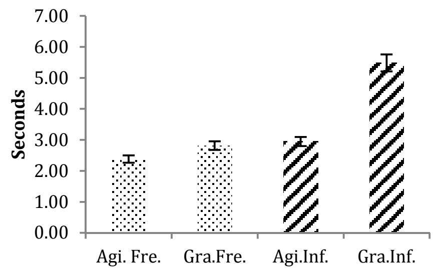

Figure 5: Detail breakdowns of looking up speed. In the figure Agi.Fre. is short for using Agile Gesture for frequently used target apps, Gra. is short for graphic interface and Inf. is short for infrequently used target apps.

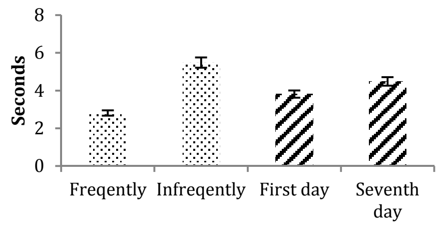

Figure 6: Detail breakdowns of looking up speed via Graphic interface

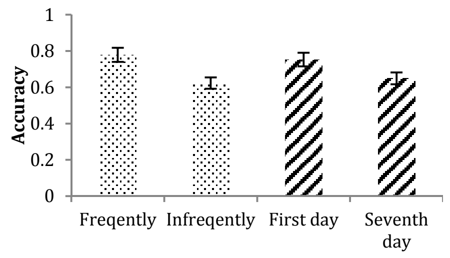

Figure 7: Gesture recall accuracy performance on Agile Gesture.

SD=3.25) since majority users need to flip back and forward several times till they finally find the app.

Look up speed performance on using graphic interface
According to paired samples t-Test results, Figure 6 shown that frequently used target apps (M= 2.82, SD=1.67) are significant easier for user to find than infrequently used apps (M= 5.49, SD=3.35) via graphic interface (P=0.003). On the other side, the Figure shows that whether participants look up the target app via graphic interface for the first day or the seventh day of the experiments, the interacting time tend to be similar. That is to say that, the speed of looking up target app via graphic interface may stay at a quite stable level unless new techniques emerge.

Gesture recall accuracy on using Agile Gesture
One important thing this paper wanted to investigate was whether the memorability on user-defined gestures would decrease when the target app were not frequently used by users. Figure 7 shown the results of paired samples t-Test, can see that the recall accuracy on frequently used target (M= 2.95, SD=1.37) apps were significant (P=0.023) higher than on infrequently used target apps (M= 2.38, SD= 1.06). It can infer that Agile Gesture's performance may be limited by user's memorability. This limitation won't extend to different recall intervals, at least the recall accuracy after the first day and seven days of the initial gesture definition did not show differences.

Conclusions and Future work

This paper followed the previous work to explore the approach for saving the app's look up time. The experiments confirmed that adopting user defined gesture shortcut for fast accessing target app can achieve better speed performance than graphic interface. However, the study also found, based on the natural differences of frequently and infrequently used apps, only relying on user-defined gestures may face several challenges. As described in the results analysis, the great advantage shows in finding infrequently used apps may be limited by the users' memorability. How to eliminate this limitation on memorability will be explored in the future.

Acknowledgements

I would like to thank my supervisor, Dr. Feng Tian, Dr. Nan Jiang and Prof. Changhong Liu for their generous support. I would also want to say thank you for all the participants who took part in the study.

References

1. Benjamin Poppinga, Alireza Sahami Shirazi, Niels Henze, Wilko Heuten, and Susanne Boll. 2014. Understanding shortcut gestures on mobile touch devices. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services* (MobileHCI '14), 173-182. http://dx.doi.org/10.1145/2628363.2628378
2. Chi Zhang, Nan Jiang, and Feng Tian. 2016. Accessing Mobile Apps with User Defined Gesture Shortcuts: An Exploratory Study. In *Proceedings of the 2016 ACM on Interactive Surfaces and Spaces* (ISS '16). 385-390. https://doi.org/10.1145/2992154.2996786
3. David Lavid Ben Lulu and Tsvi Kuflik. 2013. Functionality-based clustering using short textual description: helping users to find apps installed on their mobile device. In *Proceedings of the 2013 international conference on Intelligent user interfaces* (IUI '13). 297-306. http://dx.doi.org/10.1145/2449396.2449434

4. Hao Lu and Yang Li. 2015. Gesture On: Enabling Always-On Touch Gestures for Fast Mobile Access from the Device Standby Mode. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (CHI '15), 3355-3364. http://dx.doi.org/10.1145/2702123.2702610
5. Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. 2009. User-defined gestures for surface computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '09), 1083-1092. http://dx.doi.org/10.1145/1518701.1518866
6. Ke Huang, Chunhui Zhang, Xiaoxiao Ma, and Guanling Chen. 2012. Predicting mobile application usage using contextual information. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing* (UbiComp '12).1059-1065. http://dx.doi.org/10.1145/2370216.2370442
7. Li Yang. 2016. Enabling New Input Dimensions for Mobile Interaction. Video. (5 December 2016.). Retrieved Jan 9, 2017 from https://www.youtube.com/watch?v=G0iplOd9JXI
8. Li-Yu Tang, Pi-Cheng Hsiu, Jiun-Long Huang, and Ming-Syan Chen. 2013. iLauncher: an intelligent launcher for mobile apps based on individual usage patterns. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing* (SAC '13). 505-512. http://dx.doi.org/10.1145/2480362.2480461
9. Mary Kearl. 2016. 99 need to know stats on the mobile customer. Retrieved Jan 9, 2017 from https://www.appboy.com/blog/mobile-customer-99-stats/
10. Mary Meeker, Liang Wu. 2013. 2013 Internet Trends. Retrieved Jan 9, 2017 from http://www.kpcb.com/blog/2013-internet-trends
11. Miguel A. Nacenta, Yemliha Kamber, Yizhou Qiang, and Per Ola Kristensson. 2013. Memorability of pre-designed and user-defined gesture sets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '13), 1099-1108. http://dx.doi.org/10.1145/2470654.2466142
12. Paul Sawers. 2014. Yahoo active data. Retrieved Jan 9, 2017 from http://thenextweb.com/apps/2014/08/26/android-users-average-95-apps-installed-phones-according-yahoo-aviate-data/
13. Statista. 2015. Number of smartphone users worldwide from 2014 to 2020 (in billions). Retrieved Jan 9, 2017 from https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/
14. Tingxin Yan, David Chu, Deepak Ganesan, Aman Kansal, and Jie Liu. 2012. Fast app launching for mobile devices using predictive user context. In *Proceedings of the 10th international conference on Mobile systems, applications, and services* (MobiSys '12). 113-126. http://dx.doi.org/10.1145/2307636.2307648
15. Tom Ouyang and Yang Li. 2012. Bootstrapping personal gesture shortcuts with the wisdom of the crowd and handwriting recognition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (CHI '12). http://dx.doi.org/10.1145/2207676.2208695
16. Yang Li. 2010. Gesture search: a tool for fast mobile data access. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology* (UIST '10), 87-96. http://dx.doi.org/10.1145/1866029.1866044
17. Yang Li. 2012. Gesture-based interaction: a new dimension for mobile user interfaces. In *Proceedings of the International Working Conference on Advanced Visual Interfaces* (AVI '12), http://dx.doi.org/10.1145/2254556.2254560