AirFlip: A Double Crossing In-Air Gesture using Boundary Surfaces of Hover Zone for Mobile Devices

Hiroyuki Hakoda, Takuro Kuribara, Keigo Shima, Buntarou Shizuki, and Jiro Tanaka

University of Tsukuba, Japan.
{hakoda,kuribara,keigo,shizuki,jiro}@iplab.cs.tsukuba.ac.jp

Abstract. Hover sensing capability provides richer interactions on mobile devices. For one such exploration, we show a quick double crossing in-air gesture for mobile devices, called *AirFlip*. In this gesture, users move their thumb into the hover zone from the side, and then move it out of the hover zone. Since this gesture does not conflict with any touch gestures that can be performed on mobile devices, it will serve as another gesture on mobile devices with touchscreens where only a limited input vocabulary is available. We implemented two applications based on AirFlip. In this paper, we show the results of a comparative user study that we conducted to identify the performance of AirFlip. We also discuss the characteristics of AirFlip on the basis of the results.

Keywords: hover gesture, mobile, input method, in-air gesture.

1 INTRODUCTION

Mobile devices with hover sensing capability have recently emerged such as ELUGA P P-03E and AQUOS PHONE ZETA SH-06E. This capability provides richer interactions on mobile devices. For example, it allows users to unlock a pattern lock without touching the touchscreen, accordingly enabling secure authentication because users do not leave their fingerprints on the touchscreen. Moreover, the capability can be used to detect a finger's movement above the touchscreen, i.e., in-air gestures on mobile devices. However, few studies have explored in-air gestures in comparison with touch gestures on mobile devices.

For one such exploration, we show a quick double crossing in-air gesture for mobile devices, called *AirFlip*, which uses side boundary surfaces of the hover zone. In this gesture, users move their thumb into the hover zone from the side, and then move it out of the hover zone (Fig. 1). Since this gesture does not conflict with any touch gestures that can be performed on mobile devices, it will serve as another gesture on mobile devices with touchscreens where only a limited input vocabulary is available. In this study, we conducted a comparative user study with only touch and Bezel Swipe [1] to identify the performance of AirFlip.

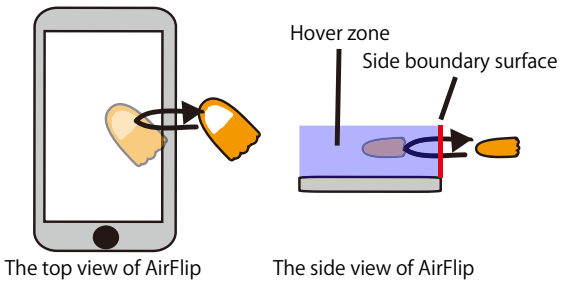

Fig. 1. Overview of AirFlip.

2 RELATED WORK

In-air gestures on various input devices have been explored. For example, ThickPad [2] is a touchpad that can sense hover gestures with proximity-sensors. Similarly, Taylor et al. [3] presented a keyboard that senses in-air gestures on the keyboard. These studies presented in-air gestures on conventional input devices such as a touchpad and a keyboard. In contrast, we explore in-air gestures above the touchscreen of mobile devices.

In-air gestures above tabletops have been explored. Interactions in the Air [4] and Continuous Interaction Space [5] focused on the space above tabletops. Han and Park [6] proposed hover based zooming interaction above tabletops. Pointable [7] is an in-air pointing technique on tabletops. Pyryeskin et al. [8] proposed a system that senses a user's hand above multi-touch surfaces using only a diffused surface illumination device. In contrast, we focus on the space above mobile devices.

In-air gestures above mobile devices have also been explored. Air+Touch [9] is a synthesis of touch and in-air gestures using an additional depth camera. Kratz et al. [10] showed a detection algorithm for in-air gestures and the design space. While these studies utilized hovering in the hover zone, AirFlip utilizes boundary surfaces of the hover zone. Han et al. [11] proposed Push-Push utilizing the pressed state and the hover state that does not conflict with a drag operation. Hover Widgets [12] utilizes movements of a pen above a screen. In contrast, AirFlip utilizes movements of a finger above a screen.

Crossing has been explored intensively especially to enrich interactions [13–18]. For example, Bezel Swipe [1] is a drag gesture starting from the bezel of mobile devices. Nakamura et al. [19, 20] proposed a double crossing gesture for hand gesture interfaces that crosses a target twice. In contrast, AirFlip is a double crossing in-air gesture that crosses a side boundary surface of the hover zone twice.

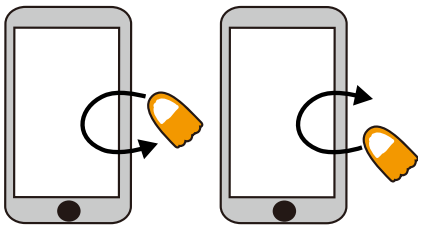

Fig. 2. Rotation gesture by twirling user's thumb.

3 DESIGN OF AIRFLIP

AirFlip is a quick double crossing gesture using the boundary surfaces of the hover zone, and users perform it with the thumb of their holding hand. Fig. 1 illustrates AirFlip. Users move the thumb into the hover zone from the side, and then move it out of the hover zone quickly. While current in-air gestures on mobile devices with hover sensing capability utilize motions such as keeping or moving their finger *within* the hover zone, AirFlip utilizes the motion that crosses the boundary surfaces of the hover zone. Moreover, AirFlip adopts a double crossing gesture because a single crossing gesture may be incorrectly recognized when users touch the screen. Due to these designs, AirFlip does not conflict with conventional touch and in-air gestures.

AirFlip has two variations: just flipping the thumb (Fig. 1) and twirling the thumb (Fig. 2). The former is suitable as a trigger of a single action and thus can be used as a button; the latter is suitable to adjust a continuous value such as a rotational angle of a map.

4 IMPLEMENTATION

We implemented AirFlip as an Android application that monitors hover events. Currently, sensing capability in Android devices begins to generate hover events when a user's finger enters the hover zone and continues to generate them until the user's finger leaves the zone. Therefore, AirFlip is recognized when hover events begin to appear and then disappear quickly (600 ms in our current implementation).

However, AirFlip is incorrectly recognized in naïve implementation for the following two reasons. First, AirFlip is recognized when users tap the screen because hover events occur before and after a tap. To address this problem, AirFlip is ignored when a touch event occurs within 50 ms after hover events disappear. Second, AirFlip is recognized when users are searching for a target to touch because their thumb tends to enter and leave the top boundary of the hover zone frequently in this context. To address this problem, AirFlip is ignored when their thumb leaves the hover zone more than 600 ms after their thumb has entered it. These realize stable recognition of both AirFlip and conventional touch gestures.

5 APPLICATION

We present two applications of AirFlip. To test these application, we used ELUGA P P-03E (Android 4.2.2) as a mobile device with hover sensing capability.

5.1 Rotating a map in map applications

We implemented a map viewer that adopts AirFlip. In this application, users can rotate a map by using AirFlip (Fig. 3); users can change the direction of rotation by changing the direction of twirl. Note that in conventional map applications, users touch an area of a map with two fingers and drag both fingers in a circular motion to rotate it. In contrast, users can rotate a map by using only one hand: i.e., users hold a device with one hand and perform AirFlip using the thumb of that hand.

5.2 Switching tabs in web browsers

Users can use AirFlip to switch tabs to the next (Fig. 4). In conventional web browsers, users need to open a list and choose a tab from it. In contrast, users can switch tabs (i.e., go to the next tab and go back to the previous tab) quickly in this application, because AirFlip is only a double crossing gesture. Users can change the direction of switching by changing the direction of twirl.

6 EVALUATION

We conducted a user study to measure the speed and usability of AirFlip. The user study is designed to measure the above metrics under the assumption that users browse web pages by selecting links and switch tabs in a web browser repeatedly.

6.1 Participants

Fourteen participants took part in the experiment as volunteers. However, we eliminated the data of two participants because we failed to collect their experimental data correctly. As the result, we used the data of 12 participants (eight males and four females) aged from 20 to 25 (mean = 22.7; = 1.29). They all used their mobile devices on a daily basis and were all right-handed. They had been using mobile devices for 11 to 99 months (mean = 34.8; SD = 24.9).

6.2 Apparatus

We used a mobile device (ELUGA P P-03E, OS: Android 4.2.2, size: height 132 mm × width 65 mm × thick 10.9 mm) with an approximately 4.7 inch touchscreen (resolution: 1080 × 1920 pixels).

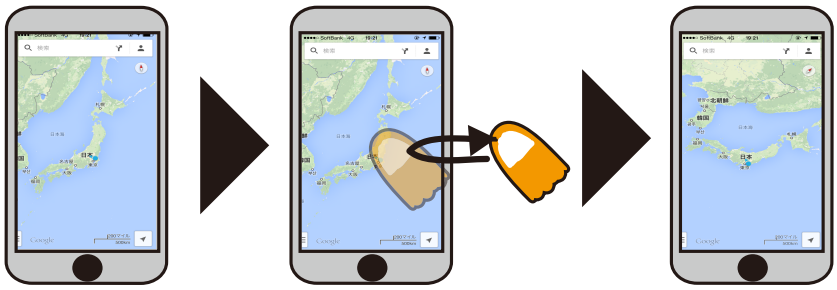

Fig. 3. Rotating a map in map applications.

Fig. 4. Switching tabs in web browsers.

6.3 Methods

We compared the performance of the following three methods for switching tabs:

AirFlip The participants switch tabs by AirFlip. They move their thumb into the hover zone from the right side, and then move it out of the hover zone quickly. A hover trajectory is displayed on the display of the device as visual feedback when users perform AirFlip.

Bezel Swipe [1] The participants switch tabs by Bezel Swipe. They start a swipe gesture from the right bezel to the left. A touch trajectory is displayed on the display of the device as visual feedback when users perform Bezel Swipe.

Touch The participants switch tabs by tapping one of the tabs.

6.4 Procedure

We asked the participants to sit on a chair and hold a mobile device in their right hand. To control the experimental conditions between participants, we also asked the participants to hold the device without supporting it by using a desk or their bodies. We asked the participants to perform this user study as accurately and rapidly as possible.

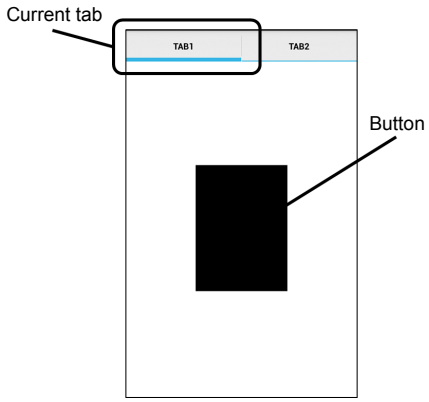

Fig. 5. Overview of the application for the user study. In this user study, there are two tabs in the web browser.

Each participant was told the goals of the user study. We also explained how to perform the three methods. Then a participant practiced each method for more than one minute. The user study started when the participant pressed the "Start" button displayed on the device's touchscreen. First, she touched a button randomly displayed on a cell in the 3 × 3 grid (Fig. 5). After that, she switched tabs by using one of the methods. In each session, she performed 18 trials (9 places × 2 tabs). She completed three sessions for each method. Thus, she performed 162 trials (3 sessions × 3 methods × 9 places × 2 tabs) in this user study.

The order of methods was counter-balanced across participants. After all trials were finished, we asked the participants to complete a questionnaire: they answered four five-point Likert scale questions (1 = strongly disagree, 5 = strongly agree) and gave reasons for their scores. The participants took about 20 minutes to complete this user study.

6.5 Results and Analysis

Fig. 6 shows task completion time of all methods, which is defined as the elapsed time between pressing a button and switching tabs. As this figure shows, the fastest method was Touch (476 ms) and the second fastest was Bezel Swipe (479 ms). AirFlip was 912 ms, approximately 1.9 times slower than the other methods.

Fig. 7 shows the results of questionnaires. Interestingly, while the accuracy of AirFlip is subjectively evaluated as the lowest, the participants felt AirFlip to be a quick gesture because its quickness is evaluated positively (4.0). On the other hand, by taking into account that AirFlip is rated the same as Touch in terms of easiness and preference, the participants were not considered to be

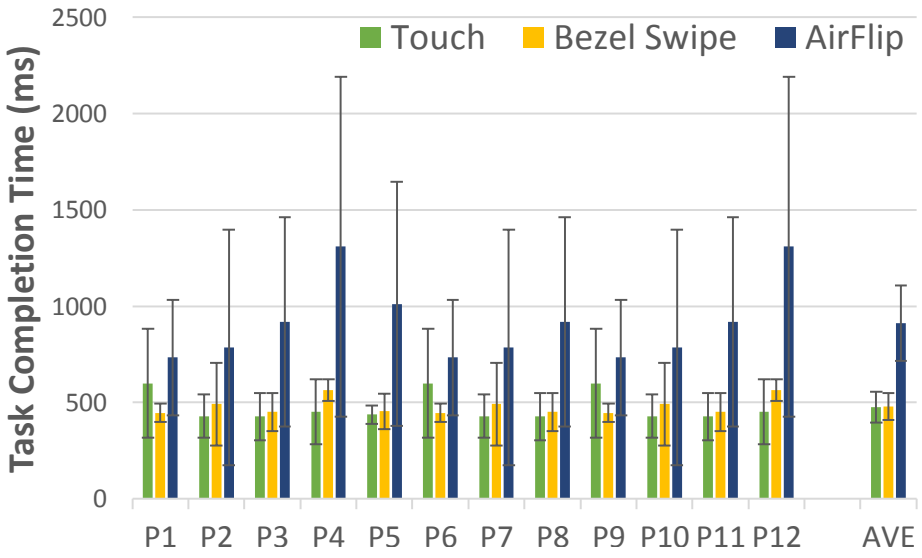

Fig. 6. Task completion time for all methods.

satisfied with AirFlip. We analyzed the comments from the participants and found that this was due to too much incorrect recognition of AirFlip (that is, the system failed to recognize AirFlip while users believed they performed the gesture correctly). Eleven participants mentioned this problem. Moreover, high variance of task completion time would be caused by this problem. Consequently, if we can reduce such errors of AirFlip, its performance may improve.

7 DISCUSSION

While we found much incorrect recognition lowered the performance of AirFlip in the evaluation, we considered that two factors of this problem can be addressed to improve the performance.

7.1 Accidental touching

We observed that participants often touched the touchscreen accidentally while performing AirFlip. This problem may have been caused by the hover zone being too narrow: hover zone is so low that hovering a finger above the screen may be difficult for users because they need to keep hovering their thumb in the hover zone when they perform AirFlip. In the questionnaire, the participants commented that the height of the hover zone is difficult to determine, and too low to perform AirFlip. Therefore, the performance of AirFlip will be improved if the hover sensing capability of mobile devices is improved to sense user's fingers at higher positions.

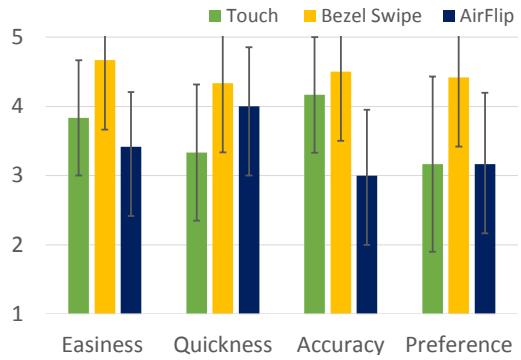

Fig. 7. The questionnaire results for the three methods (5-point Likert scale).

Moreover, we plan to attach a protective case to a mobile device shown in Fig. 9. This case is designed so that its side is higher than the surface of the device. With this case, users will be able to use AirFlip by flipping the side of the case.

7.2 Incorrectly thinking one's thumb has moved out of the hover zone

We also observed that participants incorrectly perceived that they had moved their thumb out of the hover zone (Fig. 8a) to perform AirFlip although the thumb stayed within the hover zone (Fig. 8b). In this case, AirFlip was not recognized because hover events continued to occur. To address this problem, we plan to provide users with feedback such as vibration when users move their thumb out of the hover zone. Accordingly, users can be made aware of the boundary surfaces of the hover zone and thus can perform AirFlip stably.

8 CONCLUSION

We presented a quick double crossing in-air gesture for mobile devices called AirFlip. We conducted a user study to measure its performance. From the results, AirFlip is slower than the other methods. The data and the participants' comments suggest this result is caused by incorrect recognition of AirFlip due to an inability to sense user's fingers in high positions. For immediate future work, we plan to incorporate a haptic feedback and measure the performance of AirFlip using a protective case whose side is higher than the surface of the device. Furthermore, we also plan to implement a mobile device with hover sensing capability that can sense user's fingers in higher positions by using a vision-based approach.

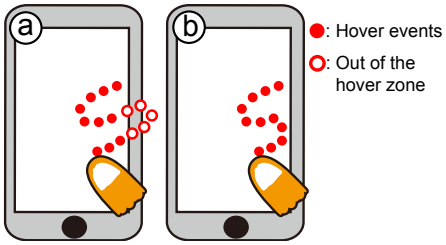

Fig. 8. Incorrect perception of thumb's position. Red circles show positions of hover events when users incorrectly thought that they had moved their thumb out of the hover zone; red rings show required trajectory to perform AirFlip.

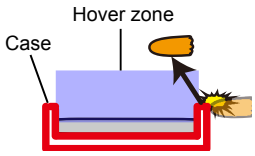

Fig. 9. A mobile device in a protective case. Users move their thumb into the hover zone by flipping the side of the case.

References

1. Volker Roth and Thea Turner. Bezel Swipe: Conflict-free scrolling and multiple selection on mobile touch screen devices. In *Proc. CHI '09*, pp. 1523–1526.
2. Sangwon Choi, Jiseong Gu, Jaehyun Han, and Geehyuk Lee. Area gestures for a laptop computer enabled by a hover-tracking touchpad. In *Proc. APCHI '12*, pp. 119–124.
3. Stuart Taylor, Cem Keskin, Otmar Hilliges, Shahram Izadi, and John Helmes. Type-hover-swipe in 96 bytes: A motion sensing mechanical keyboard. In *Proc. CHI '14*, pp. 1695–1704.
4. Otmar Hilliges, Shahram Izadi, Andrew D. Wilson, Steve Hodges, Armando Garcia-Mendoza, and Andreas Butz. Interactions in the Air: Adding further depth to interactive tabletops. In *Proc. UIST '09*, pp. 139–148.
5. Nicolai Marquardt, Ricardo Jota, Saul Greenberg, and Joaquim A. Jorge. The Continuous Interaction Space: Interaction techniques unifying touch and gesture on and above a digital surface. In *INTERACT (3)*, Vol. 6948 of *Lecture Notes in Computer Science*, pp. 461–476, 2011.
6. Seungju Han and Joonah Park. A study on touch & hover based interaction for zooming. In *CHI EA '12*, pp. 2183–2188.
7. Amartya Banerjee, Jesse Burstyn, Audrey Girouard, and Roel Vertegaal. Pointable: An in-air pointing technique to manipulate out-of-reach targets on tabletops. In *Proc. ITS '11*, pp. 11–20.
8. Dmitry Pyryeskin, Mark Hancock, and Jesse Hoey. Comparing elicited gestures to designer-created gestures for selection above a multitouch surface. In *Proc. ITS '12*, pp. 1–10.
9. Xiang 'Anthony' Chen, Julia Schwarz, Chris Harrison, Jennifer Mankoff, and Scott E. Hudson. Air+Touch: Interweaving touch & in-air gestures. In *Proc. UIST '14*, pp. 519–525.
10. Sven Kratz and Michael Rohs. HoverFlow: Expanding the design space of around-device interaction. In *Proc. MobileHCI '09*, pp. 4:1–4:8.
11. Jaehyun Han, Sunggeun Ahn, and Geehyuk Lee. Push-Push: A two-point touchscreen operation utilizing the pressed state and the hover state. In *Proc. UIST'14 Adjunct*, pp. 103–104.

12. Tovi Grossman, Ken Hinckley, Patrick Baudisch, Maneesh Agrawala, and Ravin Balakrishnan. Hover Widgets: Using the tracking state to extend the capabilities of pen-operated devices. In *Proc. CHI '06*, pp. 861–870.
13. Stuart Pook, Eric Lecolinet, Guy Vaysseix, and Emmanuel Barillot. Control menus: Excecution and control in a single interactor. In *CHI EA '00*, pp. 263–264.
14. François Guimbretiére and Terry Winograd. FlowMenu: Combining command, text, and data entry. In *Proc. UIST '00*, pp. 213–216.
15. Johnny Accot and Shumin Zhai. More than dotting the i's — foundations for crossing-based interfaces. In *Proc. CHI '02*, pp. 73–80.
16. Pierre Dragicevic. Combining crossing-based and paper-based interaction paradigms for dragging and dropping between overlapping windows. In *Proc. UIST '04*, pp. 193–196.
17. Yuexing Luo and Daniel Vogel. Crossing-based selection with direct touch input. In *Proc. CHI '14*, pp. 2627–2636.
18. Chen Chen, Simon T. Perrault, Shengdong Zhao, and Wei Tsang Ooi. BezelCopy: An efficient cross-application copy-paste technique for touchscreen smartphones. In *Proc. AVI '14*, pp. 185–192.
19. Takashi Nakamura, Shin Takahashi, and Jiro Tanaka. Double-Crossing: A new interaction technique for hand gesture interfaces. In *Computer-Human Interaction*, Vol. 5068 of *Lecture Notes in Computer Science*, pp. 292–300, 2008.
20. Takashi Nakamura, Shin Takahashi, and Jiro Tanaka. The selection technique of hand gesture in large screen environment—proposal of double-crossing and comparison with other techniques—. *The Institute of Electronics, Information and Communication Engineers Transactions*, Vol. J96-D, No. 4, pp. 978–988, 2013. (In Japanese).