Fingertip-Specific Mobile Interaction Through Camera-Based Fingertip Identification

Liang Chen, Dongyi Chen* and Zhizhong Tan

University of Electronic Science and Technology of China, No.2006, Xiyuan Ave, West Hi-Tech Zone, Chengdu, China.
Email: liangchen0513@gmail.com, dychen@uestc.edu.cn
*Corresponding Author Email: dychen@uestc.edu.cn

Abstract. In this paper, we propose to utilize camera-based fingertip identification to enable fingertip-specific mobile interaction without hand occlusion. The core idea of the proposed fingertip identification method is to utilize features, whether innate or got up, on a fingertip to recognize its identity rather than utilize hand information to infer the fingertip’s identity. A fighter game prototype is presented to demonstrate the interaction idea. Possible ways to realize fingertip identification and how to utilize fingertip identification to enable richer interactions on mobile devices are also discussed.

1. Introduction
Direct touch has been widely used on off-the-shelf mobile devices for the past decade, enabling users with natural and intuitive touch gestures to manipulate mobile interfaces. Nevertheless, it also brings in usability problems, such as the occlusion problem [1] and fat finger problem [2], which hamper user experience. Many approaches have been investigated to alleviate the negative effects of the abovementioned problems. Back-of-device interaction [3], which employs the input unit(s) on the rear for manipulation, is deemed to be one of the most promising approaches. Among the input units used in prior research literature [4-6], the back-facing cameras are the few input units that have already become standard on mobile devices. Hence, it is reasonable to consider making use of the input from a back-facing camera to realize additional interaction on a mobile touch device when touch interaction is not suitable to use.

In fact, it is not new to utilize camera-based input on mobile devices. There are tons of mobile AR (augmented reality) studies leveraging the inbuilt camera as the input unit. Since our focus is not to augment the surroundings or interact with virtual objects placed in physical world, most mobile AR studies are not directly related to our study. To use the rear-facing camera input for manipulating ordinary mobile UIs (user interfaces), there are mainly two types of interaction styles [7]: moving the holding device [8,9] and moving an object in front of the rear-facing camera. One drawback of the first type of interaction style is that the user’s eyes or even head has to follow the device’s moving direction in order to keep awareness of the contents on screen, which may cause fatigue and even increase mental load.

The second type of interaction style can address this issue. Hachet et al. [10] introduced a camera-based interaction method which a user operated the mobile UI by moving an object -- a card with special designed color patterns -- in front of the rear-facing camera. In this practice, a user does not need to move the device as well as his/her eyes or head during manipulation. Gallo et al. [11] went one step further to get rid of the cumbersome holding card by directly using a finger for manipulation. The results from [7] also indicated that the participants performed better in pointing tasks by using the
interaction style of moving the finger than the interaction style of moving the device itself. Baldaulf et al. [12] and Song et al. [13] respectively presented techniques which provided gesture input by two or more fingertips. One common problem of the fingertip detection methods used in [11-13] and similar methods used on stationary or mobile devices to enable single- or multi-finger interaction [14-16] is that their fingertip detection approaches usually include two steps -- first detect the hand(s) and then estimate the fingertip(s) belonging to each detected hand. That is, in order to detect the fingertip(s), at least part of the hand, including part of the palm and finger(s), must appear in the camera’s FOV. The fingertip(s) will not be detected if only the finger(s) or fingertip(s) is in the camera’s FOV. Therefore, the user has to move the manipulating hand to a certain distance away from the rear camera, which is more likely to cause fatigue and mistakes.

Hirobe et al. [17] made use of a template-based matching method to track a fingertip. Makino [18] employed a Haar-like features classifier for detecting the thumbnail and utilized it for HMD (head-mounted display) manipulation. Although their approaches can detect fingertip(s) without first detecting the palm, they cannot identify a specific fingertip among detected fingertips or can only detect one fingertip without knowing its identity.

Touch manipulation augmented by finger(tip) identification has become a research hotspot [19,20]. Unlike traditional touch manipulation, it maps fingers to different functionalities. Take DualKey [20], a virtual keyboard for miniature screens, for example, each key contains two letters or symbols (e.g. ‘as’, ‘df’); a letter ‘a’ will be entered by tapping the ‘as’ key with the index finger while a letter ‘s’ will be entered by tapping the same key with the middle finger. Although the practice of mapping fingers to different functionalities is deemed positively from users, it is prone to cause more hand occlusion during manipulation on small touchscreens since at least two fingers involved in interaction are hovering over the display.

To avoid hand occlusion, we argue that mobile manipulation through camera-based fingertip identification should be explored. In this paper, we present a game prototype which utilizes fingertip identification, realized by utilizing the rear-facing camera input, for game manipulation, thereby enabling fingertip(s)-specific interactions without hand occlusion. The core idea of the fingertip identification method is to directly utilize either innate or got-up features on fingertips for recognizing their identities rather than indirectly infer their identities through the detected hand information. In this way, as long as the specific fingertip(s), e.g. the tip of index finger or the tips of index and middle fingers, appear(s) in the camera’s field of view (FOV), the functionality associated with the fingertip(s) will be triggered.

![Figure 1. The workflow of the game prototype.](image_url)

2. The Game Prototype
The game prototype was developed in Java and Android SDK. It mainly contains three modules: the fingertip identifier, the command recognizer, and the fighter game. The fingertip identifier analyzes each incoming image from the rear-facing camera and extracts useful parameters of each detected
fingertip for the next step of command recognition. The command recognizer utilizes the extracted parameters of each fingertip to confirm which command(s) the user has just conducted. The fighter game periodically accepts command input, calculates the current state of each item, e.g. enemy plane, enemy bullet, and so on, in the game according to the game logic, and renders game graphics on the display. The workflow of the game prototype is illustrated in Figure 1.

2.1. The Fingertip Identifier
The fingertip identifier is in charge of turning detected fingertip(s) into corresponding parameter(s) for the next step of command recognition. As our main intention is to explore mobile manipulation through camera-based back-of-device interaction via fingertip identification rather than develop complex computer vision techniques, we make use of fingertip-worn markers in different colors for simplifying the complexity of distinguishing different fingertips. The current practice of our prototype is to use a blue marker to represent the tip of the index finger and a green marker for the middle fingertip.

The fingertip identifier, which utilizes a thresholding algorithm similar to the one used in [21], detects each color marker appeared in the camera frame and then calculates each marker’s center point’s coordinates. As a result, we obtain the following parameters of each detected fingertip: its identity (the tip of index or middle finger) and the coordinates of its center point. We use IF, IF, MF, and MF to respectively indicate whether a specific fingertip is detected or not, as shown in Table 1.

After the extracted parameters are obtained, they will be sent to the command recognizer.

| Abbreviation | Description                     |
|--------------|---------------------------------|
| IF           | The tip of the index finger is detected. |
| IF           | The tip of the index finger is not detected. |
| MF           | The tip of the middle finger is detected. |
| MF           | The tip of the middle finger is not detected. |

2.2. The Command Recognizer
The command recognizer is responsible for identifying the command(s) which a gamer inputs through the analysis of the extracted parameters obtained from the fingertip identifier. The core of the command recognizer is the mapping of gamer actions (inferred from the extracted parameters) to game commands. This mapping is related to the interaction design of the fighter game.

In current game, there are four commands respectively for controlling navigation, bullet, missile, and shield. Since mechanically keeping the connections between user actions and functions in mind would increase the cost of learning, we tried to link them in a relatively natural and intuitive manner when designing the interactions.

Navigation Manipulation: Since the index finger is naturally used for pointing things in daily life, it is rather intuitive to be utilized for fighter navigation. Therefore, we make use of the coordinates of the center of its tip for positioning the fighter on display.

Bullet Manipulation: In order to eliminate more enemies, bullet-firing often accompanies with fighter navigation. Therefore, adding up another fingertip adjacent to the index fingertip for realizing bullet-firing is intuitive and easy to operate. In the current prototype, the bullet-firing function is enabled by showing up both index and middle fingertips.

Missile Manipulation: Since a missile is a massive destructive weapon, which can eliminate all enemies and their weapons in its way, the fighter is safe at the launching position. Thus, the index fingertip for navigation is unneeded when the fighter launches a missile. As a result, we make use of showing up only the middle fingertip for launching a missile.

Shield Manipulation: Similar to missiles, the shield keeps the fighter safe for a short while during which the fighter does not need to move its position. Therefore, we design to trigger the shield by showing up no fingertips.
The mapping of extracted parameters from gamer actions to game commands is listed in Table 2. To summarize, in our interaction design, the index fingertip is for navigation, the middle fingertip is related to firing assault weapons, and no fingertip triggers the defensive weapon. The command recognizer utilizes the mapping to identify game commands and then sends them to the fighter game.

Table 2. The mapping of extracted parameters to game commands.

|   | MF                  | MF                  |
|---|---------------------|---------------------|
| IF | Navigation and bullet-firing | Navigation |
| IF | Missile-launching   | Shield triggering |

2.3. The Fighter Game

In our fighter game, a series of enemy aircrafts continuously fly out from the top of the display towards its bottom and fire bullets during the process. The gamer controls the fighter to destroy the enemy aircrafts via different assault weapons and avoid being hit by enemies. Figure 2 illustrates several views of how to manipulate the fighter during game. Figure 2(b) and 2(d) demonstrate that different functions (navigation and missile-launching) are triggered by different fingertips (index or middle). Figure 2(c) shows how a fingertip combination can achieve two functions (navigation and bullet-firing) simultaneously. Note that, only showing up the fingertip(s) in the camera's FOV is enough to trigger the function(s).

Figure 2. The fighter game manipulation. The gamer (a) makes a fist to trigger the shield, (b) shows up the tip of the index finger for navigation, (c) shows up the index and middle fingertips for firing bullets while navigating, (d) only shows up the tip of the middle finger to launch a missile.
3. Discussions and Future Work
In current study, to reduce the complexity of development, we made use of color markers for simplifying fingertip identification. This practice allowed us to focus on conveying the interaction idea rather than paying too much attention to resolving complex computer vision problems. However, computer vision researches on making use of natural fingertip features, e.g. natural patterns on human nails, for identification are of great importance. To the best of our knowledge, there is no research in this area yet. With the recent rapid development, computer vision researchers can open up a new research area to explore distinguishing different fingertips through various approaches, say, deep learning [22]. Certainly, got-up features on fingertips are much easier to be used for realizing fingertip identification, e.g. on-nail decorations (patterns or colors). Although male users are less likely to wear conspicuous on-nail decorations, they may accept handwritten symbols for temporary use. Another possible way is to use certain special coatings, which are transparent to humans but can be detected and differentiated by computer vision techniques, on nails.

In the current game prototype, we make use of two fingertips for manipulation. Shall we make use of more fingertips to realize more interactions? Apparently, the first idea across our mind is that the more fingertips involve in interaction, the more interactive commands can be generated. However, this does not simply mean the more fingertips involve the better the manipulation will be. On one hand, many fingertip combinations are not suitable to use due to physiological limitations of human hands. On the other hand, as interaction fingertips increase, user's mental cost for grasping and using the commands generated by the fingertip combinations will also increase. Moreover, when interacting with a mobile UI by camera-based input, you should always keep all of the manipulating fingertips within the camera's FOV, otherwise the desired control will not be achieved. Therefore, we will do further researches to get a more complete understanding of camera-based interaction, e.g. which fingertips and fingertip combinations are more suitable to use.

Utilizing more features of each fingertip can further increase interaction expressiveness. In current implementation, we merely make use of two parameters: the identity of each fingertip (index or middle) and the position information (the coordinates). In the future, other features can be added to generate richer interactions. For example, the depth information can be used. Even with an ordinary camera, the depth information can be coarsely inferred from the size of the fingertip: the closer/farther the fingertip from the camera, the bigger/smaller the fingertip in the camera frame. In our game prototype, the depth information can be used for adjusting the firing rate.

4. Conclusion
In this paper, we propose to make use of camera-based fingertip identification, directly utilizing the features (whether innate or got-up) on a fingertip, to realize occlusion-free and fingertip-specific mobile interaction. We present a fighter game prototype to further demonstrate our interaction idea. In addition, we discuss possible approaches to realize fingertip identification, and future studies on human computer interaction perspectives. The proposed interaction method can also be applied to many other mobile applications, not just games. It may serve as a supplementary interaction technique that can be used when direct touch is not suitable to use.

5. Acknowledgments
This work is supported by the National Key R&D Program of China (No. 2016YFB1001401).

6. References
[1] Wigdor D, Forlines C, Baudisch P, Barnwell J, Shen C. Lucid-Touch: a see-through mobile device. In Proceedings of the 20th annual ACM symposium on User interface software and technology, 2007, 269-278.
[2] Siek K, Rogers Y, Connelly K. Fat finger worries: how older and younger users physically interact with PDAs. In Proceedings of the 2005 IFIP TC13 international conference on Human-Computer Interaction, 2005, 267-280.

[3] Baudisch P, Chu G. Back-of-device interaction allows creating very small touch devices. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2009, 1923-1932.

[4] Scott J, Izadi S, Rezai L, Ruszkowski D, Bi X, Balakrishnan R. Reartype: text entry using keys on the back of a device. In Proceedings of the 12th international conference on Human computer interaction with mobile devices and services, 2010, 171-180.

[5] Chen L, Chen D, Chen X. BackAssist: Augmenting Mobile Touch Manipulation with Back-of-Device Assistance. IEICE Trans. Inf. & Syst., 2018, E101–D(6), 1682-1685.

[6] Granell E, Leiva L A. Less Is More: Efficient Back-of-Device Tap Input Detection Using Built-in Smartphone Sensors. In Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces, 2016, 5-11.

[7] Chen L, Chen D. An empirical study of interaction styles of mobile camera-based cursor manipulation. J Comput Theor Nanos, 2016, 13(1), 989-992.

[8] Rohs M. Real-world interaction with camera-phones. In Proceedings of the 2nd International Symposium on Ubiquitous Computing Systems, 2004, 74-89.

[9] Wang J, Zhai S, Canny J. Camera phone based motion sensing: Interaction techniques, applications and performance study. In Proceedings of the 19th annual ACM symposium on User interface software and technology, 2006, 101-110.

[10] Hachet M, Poudroux J, Guittton P. A camera-based interface for interaction with mobile handheld computers. In Proceedings of the 2005 symposium on Interactive 3D graphics and games, 2005, 65-72.

[11] Gallo O, Arteaga S M, Davis J E. Camera-based pointing interface for mobile devices. In Proceedings of the 15th IEEE International Conference on Image Processing, 2008, 1420 -1423.

[12] Baldauf M, Zambanini S, Fröhlich P, Reich P. Markerless visual fingertip detection for natural mobile device interaction. In Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services, 2011, 539-544.

[13] Song J, Soros G, Pece F, Fanello S R, Izadi S, Keskin C, Hilliges O. In-air gestures around unmodified mobile devices. In Proceedings of the 27th annual ACM symposium on User interface software and technology, 2014, 319-329.

[14] Lee T, Hollerer T. Handy AR: markerless inspection of augmented reality objects using fingertip tracking. In Proceedings of the 11th IEEE International Symposium on Wearable Computers, 2007, 83-90.

[15] Krejnov P, Bowden R. Multi-touchless: Real-time fingertip detection and tracking using geodesic maxima. In Proceedings of the 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition, 2013, 1-7.

[16] Fanello S R, Keskin C, Izadi S, Kohli P, Kim D, Sweeney D, Criminisi A, Shotton J, Kang S, Paek T. Learning to be a depth camera for close-range human capture and interaction. Acm T Graphic, 2014, 33(4), 1-11.

[17] Hirobe Y, Niikura Y, Watanabe Y, Komuro T, Ishikawa M. Vision-based input interface for mobile devices with high-speed fingertip tracking. In Adjunct proceedings of the 22nd annual ACM symposium on User interface software and technology, 2009, 7-8.

[18] Makino, Y. Thumbnail input for head-mounted display. Haptic Interaction, LNEE, 2015, 277, 197-200.

[19] Gil H, Lee D, Im S, Oakley I. TriTap: identifying finger touches on smartwatches. In Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, 2017, 3879-3890.

[20] Gupta A, Balakrishnan R. DualKey: miniature screen text entry via finger identification. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, 2016, 59-70.
[21] Chen L, Chen D, Chen X. MoCamMouse: Mobile camera-based mouse-like interaction. In Proceedings of the 4th International Conference on Systems and Informatics, 2017, 132-136.
[22] Lecun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521, 436-444.