Int. J. Human–Computer Studies 179 (2023) 103105

Contents lists available at ScienceDirect

International Journal of Human - Computer Studies

journal homepage: www.elsevier.com/locate/ijhcs

Around-device finger input on commodity smartwatches with learning guidance through discoverability

Marium-E- Jannat [a,*], Xing-Dong Yang [b], Khalad Hasan [a]

[a] The University of British Columbia - Okanagan, 3333 University Way, Kelowna, BC V1V 1V7, Canada
[b] Simon Fraser University, 8888 University Dr, Burnaby, BC V5A 1S6, Canada

ARTICLE INFO

Keywords:
Smartwatch
Finger posture
Finger gesture
Around-device input
Mid-air
Back of the palm
Learning guidance
Discoverability

ABSTRACT

User interactions with smartwatches are limited due to one-finger usage on small touch screens. However, smartwatch interaction capabilities can be extended beyond the screen into the around-device space by leveraging multi-finger interactions. In this paper, we explore suitable finger postures and gestures in mid-air and on the back of the palm to extend interaction capabilities on the unmodified commodity smartwatch while ensuring their learnability through discoverability. We conduct a design study to find a set of finger postures and gestures suitable for interacting with off-the-shelf smartwatches. An application is then designed to detect the posture-gesture set with on-device dual cameras while examining their classification accuracy. To facilitate learnability through discoverability of the gesture-posture set, we explore 'context-aware-hints' that shows a sub-set of gesture-posture and their associated actions based on the usage context. Results from a user study show that the guidance helps users to discover and learn them quickly and accurately.

1. Introduction

People commonly use one finger to interact with smartwatches — while leaving the other fingers idle. This is primarily due to the devices' limited touch input space, making it challenging to use multiple fingers on the screen. Prior research highlighted that this form of interaction limits the overall input bandwidth on the device (Van Vlaenderen et al., 2015; Yeo et al., 2016; Han et al., 2017; Schirra and Bentley, 2015; Knibbe et al., 2014). Further, small screens on smartwatches exaggerate many commonly-known issues on smart devices (e.g., smartphones), such as fat finger (Boring et al., 2012; Siek et al., 2005) and occlusion (Xia et al., 2015). To mitigate these limitations, researchers (Seyed et al., 2016; Knibbe et al., 2014; Van Vlaenderen et al., 2015; Yeo et al., 2016; Han et al., 2017; Ahlström et al., 2018) explored ways to extend smartwatch interaction capabilities beyond the watch screen. They showed hand gestures and postures leveraging around device space as a promising solution to extend the interaction space. For instance, Sridhar et al. (2017) developed a smartwatch prototype capable of detecting finger activities on and above the back of the palm with an external depth camera placed on the forearm. Chen et al. (2014) demonstrated detecting finger gestures in mid-air with the help of an external depth camera. All these solutions primarily depend on additional hardware or instrumentation (e.g., 3D depth camera Sridhar et al., 2017; Chen et al., 2014, magnetometer McIntosh et al., 2019; Park et al., 2020) for tracking hand activities, which limits the adoption of the solutions for everyday use (Yeo et al., 2016). In addition, gestural interactions are not self-revealing (Baudel and Beaudouin-Lafon, 1993) and users need to explicitly discover, learn, and memorize them (Bau and Mackay, 2008; Fennedy et al., 2021) — which further impose additional challenges to users.

In this paper, we extend the interaction capability of an unmodified commodity smartwatch by enabling finger input tracking (e.g., gestures and postures) around the device while ensuring that users can easily discover and learn the gestures and postures. More specifically, we leverage dual cameras on an off-the-shelf smartwatch (LEMFO LEM14) to track users' finger activities in mid-air above the watch (with the front camera) and on the back of the palm (with the side camera). Fig. 1 provides the overview of the works described in this paper. We started our exploration with a design study to find suitable and user-preferred finger gestures and postures around the device for interacting with the smartwatch. We selected a set of finger gestures and postures with a suitable delimiter based on the results and further investigated how accurately they can be classified with deep-learning techniques. As discussed above, prior research demonstrated around-device interaction with the help of additional instrumentation to the smartwatch which is not practical for everyday use. Moreover, to our

* Corresponding author.
E-mail addresses: marium.jannat@ubc.ca (M. Jannat), xingdong_yang@sfu.ca (X. Yang), khalad.hasan@ubc.ca (K. Hasan).

https://doi.org/10.1016/j.ijhcs.2023.103105
Received 20 December 2022; Received in revised form 20 June 2023; Accepted 3 July 2023
Available online 7 July 2023

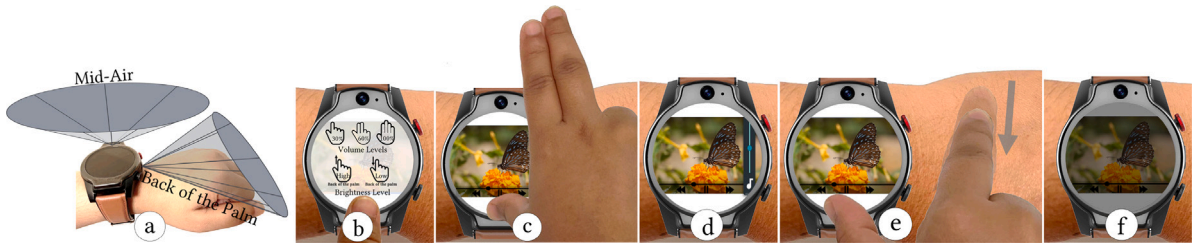


Fig. 1. Our system extends smartwatch input space to leverage (a) mid-air and the back of the palm space to extend interaction capabilities on unmodified commodity smartwatches. We explored (b) context-aware-hints that guide users to show available postures and gestures they can use with a delimiter (i.e., thumb touch on screen). (c) a user can use finger posture and (e) gesture on the extended interaction space to perform tasks on smartwatches. Here, (d) the user changes the volume level to 60% with (c) a straight two-finger posture in mid-air and (f) lowers the screen brightness level with (e) one finger gesture on the back of the palm space.

best knowledge, no prior work has leveraged both mid-air and the back of the palm space with user-preferred finger postures and gestures accompanied by delimiters on an unmodified commodity smartwatch to extend the device's input capabilities. Therefore, exploring around-device finger input accompanied by suitable delimiters for unmodified commodity smartwatch can open new possibilities for extending overall input capabilities of commercial smartwatches. Results showed that the selected gesture set could be classified with high accuracy (average 93.27% with Mask-RCNN), while some postures were less accurate due to multi-finger occlusions. As the gesture and postures are not self-revealing like buttons, we next investigated ways to assist users in discovering and learning the set of gestures and postures to interact with smartwatches which has never been explored before. We thus explore 'context-aware-hints' — a learning guidance technique that shows available postures or gestures and associated tasks with on-screen overlays based on the application context when touching the screen. With a user study, we provide evidence that (1) showing only application-relevant gestures improves learning and (2) additional steps (i.e., scrolling) can detract from learning.

Our main contributions include:

- An exploration of user-preferred suitable finger gestures and postures that can be used to interact with an unmodified commodity smartwatch
- An proof-of-concept prototype application to detect finger gestures and postures both in mid-air and on the back of the palm with a high accuracy
- An exploration of a guidance techniques for learning through discoverability of the postures and gestures

2. Related works

We briefly review previous works that inspired us to design our interaction technique. This section is organized into two sections: the first section focuses on around-device interaction techniques for smartwatches, and the second section discusses the discoverability and learnability of new interactions.

2.1. Around-device interaction

This section includes reviewing the prior works focusing on around-device interaction capabilities and around-device sensing techniques for smartwatches.

2.1.1. Around-device interaction capabilities

Researchers have explored ways to extend smartwatch interaction space beyond its tiny screen into the around device space (Knibbe et al., 2014; Zhou et al., 2016; McIntosh et al., 2019; Zhang et al., 2016b; Sridhar et al., 2017; Jannat et al., 2022; Pietroszek et al., 2017). For instance, Lim et al. (2015) demonstrated tap, swipe and scrolling on smartwatches by detecting the index finger gesture on the back of the palm. Knibbe et al. (2014) demonstrated single-finger tap, swipe gesture and two-finger pinch-to-zoom gesture on the back of the palm to support around-device interaction on smartwatches. Arefin Shimon et al. (2016) explored a set of 31 hand gestures for smartwatches, including motion gestures, gestures on bezel and touch gestures. However, Most of the gestures (e.g., motion gestures of the wearing hand, touch on the bezel) on their list are not identifiable with the on-device's cameras. Also, they did not explore any delimiter options for gesture-based input which is very crucial for differentiate a valid intentional gesture from an unintentional one. During our exploration, we considered the gestures (e.g., hovering over the watch face) from their list that are recognizable through cameras. McIntosh et al. (2019) presented tracking users' index finger around the watch to support tap and hovering. In a recent work, Hayashi et al. (2021) presented RadarNet, where a miniature radar is capable of tracking and recognizing five swiping gestures around the smartphone space by analyzing the electromagnetic waves emitted by the radar.

A few other prior works (Zhou et al., 2016; Park et al., 2020) also investigated finger gestures in around-device space to control applications on smartwatches. In a recent work, Sridhar et al. (2017) used a combination of mid-air and multi-touch input on and above the back of the palm for controlling games, scrolling menus and zooming maps. Several prior works (Xu et al., 2015; Gupta and Balakrishnan, 2016; Gong et al., 2018) demonstrated writing on the smartwatch using finger gestures in mid-air space. Gil et al. (2017) identified thumb, index and middle fingers in mid-air to use the on-screen keyboard on the smartwatch efficiently. Nascimento and Soares (2020) demonstrated controlling videos on smartwatches by detecting finger gestures in mid-air. Nascimento et al. (2019) did the same to control Netflix movies on a smartwatch. Mid-air finger gestures were also used for mode switching or triggering commands (Kim et al., 2007; Sun et al., 2017). Moreover, there are prior works (Perrault et al., 2012; Zhang et al., 2016b; Klamka et al., 2020) leveraging different parts (e.g., strap, knob, bezel) of the watch as around-device interaction space. For instance, Perrault et al. (2012) explored touch gestures on watchstrap while Zhang et al. (2016b) used both bezel and strap as interaction space. In a recent work, Klamka et al. (2020) presented watch straps as both input and output spaces with multi-touch and pressure-touch inputs. Further, researchers (Gong et al., 2016; Reyes et al., 2018; Zhu et al., 2018b; Wen et al., 2016) demonstrated using hand gestures on the around-device space to support one-handed input on smartwatches.

In summary, over the past few years, researchers explored different input methods (e.g., gestures and motions) on different input spaces (e.g., mid-air, back of the palm) to extend the interaction capabilities. However, none of these prior works explored delimiters for the around-device input. Our work explores a large set of user-preferred finger postures and gestures along with a delimiter that leverages both mid-air above the watch face and the back of the palm as input space.

2.1.2. Around-device sensing

Prior research investigated different sensing mechanisms to detect gestures and postures around the device. For instance, Lim et al. (2015) detected finger gestures using Infrared (IR) sensors in the around-device

space of smartwatches. Similarly, Knibbe et al. (2014) tracked multi-finger activity on the back of the palm using multiple infrared (IR) sensors. Zhang et al. (2016a) presented TapSkin, which recognizes tap gestures at eleven locations (ten on the back of the palm and one on the forearm) on the hand skin around the watch. They used inertial sensors and microphones attached to the smartwatch to classify the gestures. They claimed to achieve higher classification accuracy for all the gestures. Hayashi et al. (2021) presented solutions for tracking and recognizing five swiping gestures around the smartphone space by sensing the electromagnetic wave with a miniature radar (Trotta et al., 2021) developed under Google's project Soli. Several prominent research (McIntosh et al., 2019; Gil et al., 2017; Lyons, 2020; Park et al., 2020; Park and Lee, 2019; Reyes et al., 2018) investigated magnetic sensing techniques for tracking finger activities in around smartwatch space. Zhou et al. (2016) used electric field sensing through electrodes and antennas to capture finger activities in around-device space. Sridhar et al. (2017) captured multi-finger activities on and above the back of the palm through a depth camera attached to the forearm. Chan et al. (2015) presented tracking finger motion using a miniature fisheye camera attached to the edge of a ring. It could track seven hand gestures observing from the finger joint position through a fisheye perspective. Yang et al. (2015) presented around-device hand posture recognition for smartwatches with EMG sensors. Xu et al. (2015) investigated tracking around-device gestures by capturing the motion of the fingers through accelerometers and gyroscope sensors. Lim et al. (2018) explored scenarios where multiple fingers are used on smart devices, with one finger engaged in touch input while the remaining fingers perform mid-air gestures to extend touch-based input capabilities. They explored 20 Touch+Finger gestures using two ring-like devices on the finger that have IMU sensor attached to recognize the mid-air gestures and suggested 8 Touch+Finger gestures to extend touch-based input capabilities. Both Zhu et al. (2018b) and Gupta and Balakrishnan (2016) identified around-device finger gestures by built-in IMU sensors in smartwatches. Wen et al. (2016) also recognized finger gestures using IMU sensors to support touch-less around-device smartwatch interaction.

Prior research investigated different sensing techniques for detecting around-device input that primarily depended on external sensors (e.g., IR sensor, mini radar, EMG sensor, depth camera) where additional instrumentation is needed. This paper investigates and presents a camera-based system on a commodity smartwatch that detects finger input in both mid-air above the watch face and back of the palm without any external sensors and additional instrumentation.

2.2. Discoverability of interaction

Any new and/or unconventional mode of interaction is difficult to discover (Baudel and Beaudouin-Lafon, 1993). For instance, gestural interactions are not self-revealing like graphical buttons and menus (Bau and Mackay, 2008). As a result, users sometimes remain unaware of the features that could make their tasks easier. It is important that the system exposes the new capabilities to users. Several prior research (Kurtenbach et al., 1994; Bau and Mackay, 2008; Anderson and Bischof, 2013; Delamare et al., 2016; Fennedy et al., 2021; Goguey et al., 2018) investigated design guidelines to enable the users to discover the available capabilities of a new interaction technique. For instance, Anderson and Bischof (2013) used a static learning guidance: "crib-sheet" to discover the new capabilities, which lists the tasks and visual illustrations of the associated gestures. In an earlier work, Kurtenbach et al. (1994) presented a combination of crib-sheet and contextual animation to let the users know which gestures are currently available, where a pop-up crib-sheet displayed the relevant subset of the gestures depending on the context. Grossman et al. (2006) demonstrated how users could learn mid-air pen gestures by hovering the pen above the screen to display all the possible gestures. Later in OctoPocus, Bau and Mackay (2008) presented a dynamic learning guidance technique that displays the whole or portion of the corresponding gesture paths on-screen and provides feedback after completing the gesture. In recent years, Delamare et al. (2016) investigated OctoPocus (Bau and Mackay, 2008) for 3D mid-air gestures and Fennedy et al. (2021) for virtual reality. Both of them reported that the guidance technique makes the gestures more discoverable; hence learnable. Sodhi et al. (2012) proposed a different visual guidance technique where the upcoming gesture direction is projected onto the user's hand as a spot or a 3D arrow.

As discussed, learning guidance can make the interaction self-learnable through discoverability. To our best knowledge, no prior work has explored the discoverability and learnability of postures and gestures on smartwatches. Therefore, we systemically explore the user-preferred finger gestures and postures along with a delimiter, develop a finger posture-gesture detection system supported by the built-in sensors and explore learning guidance techniques to support learning through discoverability.

3. Design space exploration

Prior work showed that extending the input space beyond the smartwatch screen opens opportunities to increase the input bandwidth on the device. However, they primarily relied on external sensors (e.g., depth cameras Sridhar et al., 2017) or instrumentation to the device (Lim et al., 2015; McIntosh et al., 2019) and/or attachment to hands (Park et al., 2020) — which might not be practical for everyday use (Schirra and Bentley, 2015; Yeo et al., 2016). We aimed to extend the smartwatch input space without relying on external instrumentation or attachments. While exploring commercial smartwatches, we found that many of them are now equipped with multiple (e.g., front and side facing) cameras (e.g., LEMFO LEM14, ZEBLAZE THOR 6, Kospet Prime S), enabling users to capture their finger activities in mid-air on top of the screen and on the back of the palm with the front and side cameras, respectively. However, prior work (Markussen et al., 2014) showed that around-device interaction should have an explicit delimiter to be activated. Therefore, we started our exploration by examining possible delimiters and finger activities (i.e., gestures and postures) that can be captured with the on-devices' cameras and used on the smartwatch.

3.1. Delimiters

Touch input has an implicit delimiter where a gesture begins with a finger touching the screen and ends with the finger lifted from the screen. However, no implicit delimiter is available for around-device gestures or postures with smartwatches. As users' finger movements around the watch can either be intentional or unintentional, we need a way to avoid detecting unintentional finger movements from users' intended inputs. Markussen et al. (2014) explored different delimiter options for a mid-air gestural keyboard and showed that touch on the screen could be a potential delimiter. Chen et al. (2014) also suggested touch on the screen as an intuitive delimiter to indicate a valid mid-air gesture input for smartphones. We are unaware of prior work exploring the delimiters for the around-device gesture input on smartwatches. Consequently, in this work, we considered touch on the screen as a delimiter while using gestures and postures with smartwatches.

We observed that people naturally use either their index finger or thumb to touch the watch screen while leaving the other fingers idle. Therefore, we decided to use the following two options as delimiters: (1) *Thumb On-screen as Delimiter* where users are required to touch the screen with the thumb to initiate a gesture or posture and lift off the finger to end it; and (2) *Index On-screen as Delimiter* is similar to the previous delimiter; however, users touch the screen with the index finger instead of the thumb. Note that touching the screen can potentially trigger an on-screen item. Therefore, we only considered a touch as a delimiter when the touch is detected outside any UI widgets (e.g., buttons, menus).

Fig. 2. Postures and gestures that we considered in our study. Images with arrows [↔, ↕] were used for gestures and postures, whereas images without arrows were only used for postures. Arrows also represent the directions of finger movements.

3.2. Finger interaction

While one finger, i.e., thumb or index, is being used as a delimiter, other fingers and their combinations can create many possibilities for postures and gestures. Hence, we reviewed prior works (Dim and Ren, 2014; Delamare et al., 2015; Seyed et al., 2016; Lu et al., 2020; Zhu et al., 2018; Ruiz et al., 2011; Arefin Shimon et al., 2016) that leverages finger postures and gestures for smart device interaction. None of these prior works explored the finger postures and gestures that can be detected by on-device cameras. We considered the finger postures and gestures from the prior works that could likely be recognizable through watch's cameras. For instance, Arefin Shimon et al. (2016) presented a list of non-touchscreen hand gestures for smartwatches and we have considered the gestures from their list that can be identified through built-in cameras of the watch. Based on our review, we prepared a list of finger postures and gestures that can be used with smart devices (e.g., smartwatches). We (two co-authors) further refined this set (i.e., gesture-posture set) that are not difficult to perform while touching the screen and can be captured by the device's front and side-facing cameras. The set is shown in Fig. 2.

3.2.1. Postures

For the posture-gesture set, we selected a total of nine mid-air postures with the thumb on the screen as delimiter. The postures are: straight index finger, straight middle finger, straight two fingers (index and middle), straight all fingers, V, hook index finger, hook middle finger, hook two fingers (index and middle), and hook all fingers (Fig. 2(a–i)). We also selected five possible mid-air postures with the index finger on the screen as a delimiter. The postures are: straight middle finger, straight all fingers (middle, ring and small finger), hook middle finger, hook all fingers (middle, ring and small finger and thumb) (Fig. 2(j–n)). We chose four possible postures on the back of the palm with the thumb on the screen as delimiter: index finger, middle finger, two finger (index and middle), and all finger tapping on the palm (Fig. 2(o–r)). For the index on the screen as a delimiter, there were three possible postures: middle finger, two finger (middle and ring finger) and all finger on the back of the palm (Fig. 2(s–u)).

3.2.2. Gestures

We selected eight possible mid-air gestures with the thumb on the screen as delimiter. The selected gestures are: one finger hovering with the index finger, one finger hovering with the middle finger, two finger (index and middle finger) hovering, all finger hovering, one finger crawling with the index finger, one finger crawling with the middle finger, two finger (index and middle finger) crawling, and all finger crawling (Fig. 2(a–d, f–i)). Next, we chose five possible mid-air gestures with the index finger on the screen as delimiter. These gestures are: one finger hovering with the middle finger, one finger hovering with the thumb, all finger than the thumb hovering, one finger crawling with middle finger, and all finger other than the thumb crawling (Fig. 2(j–n)). We chose four possible gestures on the back of the palm with the thumb on the screen as a delimiter: one finger sliding on back of the palm with the index finger, one finger sliding with the middle finger, two finger (index and middle) sliding, and all finger sliding (Fig. 2(o–r)). For the index on the screen as delimiter, there were three possible back of the palm gestures: one finger sliding with the middle finger, two finger (middle and ring finger) sliding and all finger other than the thumb sliding (Fig. 2(s–u)).

3.3. User study

We ran a user study to gather users' preferences on the gesture-posture set that we selected. We recruited 30 participants (21 male, 9 female) ages between 20 and 40 (M = 28.47, SD = 5.75) by posting the study advertisements on different online forums (e.g., Facebook) and advertising sites (e.g., Castanet). Due to the surge of the Covid-19 omicron variant, we conducted study sessions online via Zoom by inviting each participant individually. At the beginning of the session, we informed the participants of the project's goal and showed them the images for the initial set of all the postures and gestures from Fig. 2. As mentioned earlier, this set of postures and gestures was prepared for this study by reviewing prior works (Dim and Ren, 2014; Delamare et al., 2015; Seyed et al., 2016; Lu et al., 2020; Zhu et al., 2018; Ruiz et al., 2011; Arefin Shimon et al., 2016). We used PowerPoint slides, where each slide contained either gestures or postures for a delimiter option, and one interaction space (e.g., mid-air/back of the palm gestures/postures for the thumb on screen/index on screen delimiter). During the session, one co-author performed the gestures and postures to show to the participants and asked them to try these by themselves. Once they were done with all the gestures/postures presented on a slide, the author asked them to provide their preference on the gestures and postures based on if they would like to use the gesture in reality and the ease of performing the gesture with 5-point Likert scales on an online questionnaire hosted on Qualtrics. More specifically, the participants were asked two questions for each of the gestures and postures:

- One question collecting users' preference (i.e., *Would you prefer using the gesture in real life while interacting with the smartwatch? Provide your preference on a scale of 5 where 1 represents least preferred and 5 represents most preferred*)
- Another question on the ease of performing the gesture (i.e., *How easy it was to perform the gesture? Provide your rating on a scale of 5 where 1 represents very hard and 5 represents very easy*).

Besides providing their preference, participants were also asked to suggest potential gestures and postures that are not presented in the slides but are worth exploring. Further, participants provided feedback on other factors such as the number of fingers that are suitable for gestures/postures, their preferred interaction area, preferred delimiter, and a comparison between mid-air and back of the palm gestures space for such interaction. Each session took approximately 45 min to complete.

3.4. Results

We used Friedman tests with Wilcoxon tests for post-hoc pairwise comparisons to analysis the 5-point Likert scale data. Post-hoc pairwise comparisons were Bonferroni adjusted. The same tests were used for analyzing postures, gestures and other factors (e.g., delimiter).

3.4.1. Postures in mid-air

While analyzing the mid-air postures with the *Thumb On-screen as Delimiter*, we found statistically significant differences among the postures ($\chi^2(8, N = 30) = 80.01, p < 0.001$). Wilcoxon tests (Bonferroni: α-level 0.05 to 0.001) showed significant differences between the *Straight Index Finger* and all other postures, between *Straight Two Finger* and *Hook Middle* and between *Straight All Finger* and *Hook Middle Finger* (all $p < 0.001$). However, *Straight Index Finger* ($M = 4.73$), *Straight Two Finger* ($M = 3.4$), *Straight All Finger*, *V* ($M = 3.1$) and *Hook Index Finger* ($M = 3.0$) received higher mean ratings than other postures. Therefore, we included them for our future exploration. We next analyzed postures with the *Index On-screen as Delimiter* and found no statistically significant differences among the postures ($\chi^2(4, N = 30) = 6.31, p = 0.18$).

3.4.2. Postures on the back of the palm

We analyzed the data for the postures on the back of the palm with the *Thumb On-screen as Delimiter* and found statistical differences ($\chi^2(3, N = 30) = 41.23, p < 0.001$). Wilcoxon tests (Bonferroni: α-level 0.05 to 0.008) showed significant differences between *One Finger (Index)* ($M = 4.53$) and other postures and between *Two Finger* ($M = 4.20$) and other postures. Therefore, we included *One Finger (Index)*, *Two Finger* into our final posture-gesture set. We next analyzed the postures perform on the back of the palm with *Index On-screen as Delimiter* and found no statistical differences ($\chi^2(2, N = 30) = 8.598, p < 0.05$).

3.4.3. Gestures in mid-air

We next analyzed the mid-air gestures with the *Thumb On-screen as Delimiter*. A Friedman test showed statistically significant differences among the gestures ($\chi^2(7, N = 30) = 98.29, p < 0.001$). Bonferroni adjusted Wilcoxon tests (α-level 0.05 to 0.001) showed differences between *Straight Index Finger* and all other gestures, *Straight Two Finger* and all other gestures except for *Hook Index Finger* and between *Hook Index Finger* and other gestures except for *Straight Two Finger* and *Straight All Finger*. However, we included *Straight Index Finger* ($M = 4.53$), *Straight Two Finger* ($M = 3.6$) and *Hook Index Finger* ($M = 3.67$) for future exploration as they received higher mean ratings than other gestures. We then analyzed the gestures with the *Index On-screen as Delimiter* and found no statistical differences among the gestures ($\chi^2(4, N = 30) = 5.206, p = 0.267$).

3.4.4. Gestures on the back of the palm

We next analyzed the gestures on the back of the palm with the *Thumb On-screen as Delimiter* and noticed statistical differences ($\chi^2(3, N = 30) = 44.47, p < 0.001$). Bonferroni adjusted Wilcoxon tests (Bonferroni: α-level 0.05 to 0.008) showed that *One Finger (Index)* ($M = 4.73$) and *Two Finger* ($M = 4.33$) on the back of the palm were significantly different than other gestures (all $p < 0.001$). Therefore, we included them for further exploration. We then analyzed the gestures with the *Index On-screen as Delimiter* and observed no statistical differences ($\chi^2(2, N = 30) = 8.374, p = 0.015$).

3.4.5. Preferred delimiter and number of interacting fingers

We asked participants to rate two delimiter (*Thumb On-screen* and *Index On-screen*) on a 5-point Likert scale. A Friedman test showed statistical differences ($\chi^2(1, N = 30) = 9.00, p < 0.01$) where Wilcoxon tests revealed that using thumb is preferred over using the index finger. We next ask questions on the number of fingers that they prefer to use for performing postures or gestures. There were three options — using one finger (e.g., index), two fingers (e.g., index and middle) and all fingers (e.g., index, middle, ring and small). The Friedman test showed statistical differences ($\chi^2(2, N = 30) = 13.218, p < 0.001$) on their preference where Wilcoxon tests showed that using one finger was significance preferred over two fingers ($p < 0.001$).

3.4.6. Interaction space

Next the participants rated three interaction space: using mid-air, back of the palm and a combination of mid-air and back of the palm. A Friedman test showed significant differences ($\chi^2(2, N = 30) = 23.38, p < 0.001$). Wilcoxon tests (Bonferroni: α-level 0.05 to 0.017) showed significant differences between mid-air and other two interaction spaces. No other comparisons were significant.

3.5. Discussion

We asked participants to select gesture or posture as their preferred mode of interaction. We found that 13 participants selected gestures as their preferred mode of interaction, where 17 selected postures. Consequently, we decided to have both gestures and postures for our implementation. For delimiter, participants preferred to use a single finger with a preference for using the thumb. Therefore, we selected the *Thumb On-screen* as the delimiter for the postures and gestures. When selecting gestures and postures, we considered both back of the palm and mid-air spaces. Therefore, in our *Final Posture-Gesture Set*, there are five mid-air postures with the *Thumb On-screen* as the delimiter (letters beside the gesture-posture are from Fig. 2): (i) *Straight Index Finger*(a), (ii) *Straight Two Finger* (c), (iii) *Straight All Finger* (d), (iv) *Hook Index Finger* (f) and (v) *V* (e); three back of the palm postures with the *Thumb On-screen* as the delimiter: (i) *One Finger (Index)* (o), (ii) *Two Finger* (q) and (iii) *All Finger* (r). The *Final Posture-Gesture Set* contains three mid-air gestures where *Thumb On-screen* is the delimiter: (i) *Straight Index Finger* (a), (ii) *Straight Two Finger* (c) and (iii) *Hook Index Finger* (f); and two back of the palm gestures: (i) *One Finger (Index)* (o) and (ii) *Two Finger* (q).

4. Implementation

We developed an application to detect and classify finger postures and gestures on a smartwatch. LEMFO LEM14 smartwatch is used for implementation and testing which has built-in dual cameras (5MP front-facing camera and a 2MP side-facing camera) and runs Android 10 OS. The front-facing camera captures the mid-air space above the watch screen, and the side-facing camera captures the area on the back of the palm. We used Mask-RCNN (Mask-RCNN); a widely used machine learning model for classifying images based on instance-segmentation.

From the design study, we selected eight postures and five gestures. We observed that all five gestures are related to the postures where the postures have no finger movements. Therefore, once we can detect and classify the postures, corresponding gestures can be classified by detecting the finger movements of the corresponding postures over time. Thus, we first investigated how well the postures can be tracked and classified on smartwatches.

4.1. Participants, data collection and annotation

Due to the surge of the Covid-19 Omicron variant, we had limited options to collect data in a lab environment while ensuring all safety protocols. Consequently, we recruited eight participants (5 male, 3 female) ages between 25 and 31 (M = 27.86, SD = 2.09) and collected images related to the selected finger postures. We used the LEMFO LEM14 smartwatch to capture the images. One author showed the participants how to perform the postures on the smartwatch. Participants were then asked to wear the smartwatch and perform the postures. We showed them postures one-by-one on a PowerPoint slide. Once participants confirmed their finger positions for a posture, we took an image on the smartwatch with its front or side camera. We collected 15 images for each posture from each participant — resulting in a total of 960 images. The participant made new poses for each image of a given posture while 15 images were captured. Next, we next applied basic image manipulation-based data augmentation techniques (e.g., rotation(10°), cropping) (Shorten and Khoshgoftaar, 2019a), to increase the image set to 1024 images. Next, we manually annotated the dataset into JSON format using VGG Image Annotator (VIA) (Dutta et al., 2016). We randomly split the dataset into 70% (~720 images; 90 images per class) to train the model, 20% (~200 images; 25 images per class) to validate the model and 10% (~104 images; 13 images per class) to test the model. The random splitting of data was done based on participants so that the outcomes are cross-user results — which was also used in Wang et al. (2021), Khan et al. (2018), Bendersky et al. (2017) and Christoudias et al. (2006).

4.2. Posture detection

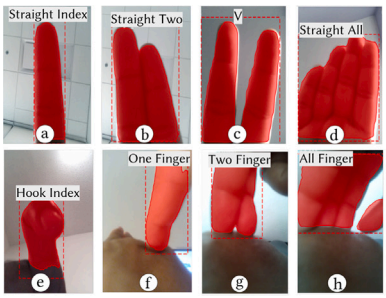
For classifying the postures, we used the official implementation of Mask-RCNN (Abdulla, 2017), which is the widely used version of this deep-learning model. This implementation is developed by following the original Mask-RCNN paper (Mask-RCNN) with a few deviations for the sake of generalization and coding simplicity. We started with the provided pre-trained weights for the Microsoft COCO dataset in the implementation as a starting point to train our dataset. We customized the system to best support our vision in an easier way while keeping the structure of the model unchanged. The original implementation was developed on TensorFlow 1. We updated the model into TensorFlow 2 to easily access the supporting APIs and libraries. We used Resnet (He et al., 2016) with 50 layers as the backbone. We resized all the images to the same size (i.e., 512 × 512px) to allow the training of multiple images per batch. We used the same learning rate of 0.02 as the original implementation as the model converged faster with the smaller learning rates. Note that we developed our Mask-RCNN model in Google Colab for faster training through accelerated run-time on Google Colab-provided GPUs. It took approximately two and a half hours to initially train and validate the model which can potentially vary based on the server load. Finally, we evaluated the model with the testing set on the Colab server. Fig. 3(i) shows the confusion matrix. We deployed the trained model on the Colab cloud server for further inference as the deep learning models are usually very computationally heavy for handheld devices. Therefore, no further overhead with training and computation while using it. The watch only needs to capture images and send the images to the cloud server and receive the numeric results back. Since Google Colab is an always available cloud-based service, there is no need for external instrumentation.

4.3. Results

We used 104 images (~10% of the dataset) – 13 images per class – to examine our model's performance. Fig. 3i shows the confusion matrix across eight postures and their classification accuracy. We observed that our model successfully classified 97 out of 104 images into the correct classes, yielding an overall accuracy of 93.27%. We found that the model achieved an accuracy of 98.45% for mid-air postures and 84.62% for the back of the palm postures. Among the mid-air postures, the model classified all the postures with 100% accuracy except for the *Straight All Finger* (Fig. 2d) posture (i.e., 92.31% accuracy). For the back of the palm postures, our model detects *One Finger (Index)* (Fig. 2o) posture with an accuracy of 100% while the other two postures *Two Finger* (Fig. 2q) posture (84.62%) and *All Finger* (Fig. 2r) (69.23%) showed lower accuracy. We believe this is primarily due to finger occlusions, where the index finger commonly occludes the other fingers on the back of the palm (e.g., *Two Finger* (Fig. 2q) and *All Finger* (Fig. 2r) — thus lowering the detection accuracy. Therefore, the system shows comparatively higher accuracy in mid-air postures than the back of the palm postures.

4.4. Implementation on smartwatch

We developed an android smartwatch app to continuously capture and send finger images to the Mask-RCNN model hosted on Google Colab. We changed the application on Google Colab so that the model could classify the image into a posture and sends the class id, class name and bounding box coordinates. We found this system can process 7 to 10 images per second (i.e., sending images to Colab and receiving results). The Android smartwatch application stored the class id and the bounding box coordinates and checked the difference between the bounding box coordinates with the previous frames stored in the last second. Once the application detects a difference higher than a threshold value (i.e., 50 pixels) for the same class ids, it is classified the user's finger movement as a gesture of the corresponding posture



(i)

		Mid-Air					Back of the Palm		
		Straight Index	Straight Two	Straight All	V	Hook Index	One Finger	Two Finger	All Finger
Mid-Air	Straight Index	100%	0	0	0	0	0	0	0
	Straight Two	0	100%	0	0	0	0	0	0
	Straight All	0	7.69%	92.31%	0	0	0	0	0
	V	0	0	0	100%	0	0	0	0
	Hook Index	0	0	0	0	100%	0	0	0
Back of the Palm	One Finger	0	0	0	0	0	100%	0	0
	Two Finger	0	0	0	0	0	15.38%	84.62%	0
	All Finger	0	0	0	0	0	0	30.77%	69.23%

Fig. 3. (a–h) Sample output of the posture detection from the testing image set; (i) Confusion matrix of classification results across 8 finger postures classes.

class as the movement of the posture is the corresponding gesture. We observed that the accuracy of gestures was similar to the accuracy of the corresponding posture as gesture detection relies on detecting the related posture.

4.5. Discussion

We acknowledge that a large set of images and advanced deep-learning algorithms would result in even higher classification accuracy. However, our current results showed that the model can still classify the set of finger postures with reasonably high accuracy (average 93.27%). We observed that finger postures on the back of the palm offer lower classification accuracy due to occlusion, which could potentially be increased through additional images with features. We also demonstrated that finger gestures could be detected from postures by leveraging the same model. As our primary goal is to demonstrate a deep learning-based proof-of-concept application suitable for detecting around-device finger postures and gestures on an unmodified commodity smartwatch while ensuring their learnability and discoverability, we next moved to explore guidance techniques to facilitate learnability and discoverability.

5. Discoverability and learnability of gestures and postures

Prior research highlighted concerns regarding the learnability and discoverability of gestures and postures on and off the devices as they are not self-revealing (Baudel and Beaudouin-Lafon, 1993). Here, users need to explicitly discover, learn, and memorize them to use with their associated activities (Bau and Mackay, 2008; Fennedy et al., 2021). Learnability and discoverability for gesture-based input on a smartwatch have never been explored before. Thus, we next explore gesture guidance techniques to help users to discover, learn and recall our selected postures and gestures and their associated tasks on smartwatches.

5.1. Guidance techniques

Prior research explored different guidance techniques to help users to discover, learn and memorize postures and gestures while interacting with devices. For instance, Crib-sheet (Kurtenbach et al., 1994; Anderson and Bischof, 2013), the most basic form of discoverability guidance technique, provides guidance by displaying the static list of all postures and gestures that can be used to interact with devices. Another well-explored approach is the context-aware guidance technique (Kurtenbach et al., 1994), which displays currently available postures and gestures with their corresponding commands based on their usage context in a desktop environment. Inspired by the prior works, we explored the following two guidance techniques in this paper to explore smartwatches.

5.1.1. Crib-sheet

Similar to the prior research (Bau and Mackay, 2008; Delamare et al., 2016; Fennedy et al., 2021), we used crib-sheet as the baseline for guidance techniques comparison. In our implementation, we initially showed the crib-sheet before starting any of the tasks with a static list of images displaying possible postures-gestures and the command associated with each gesture, as shown in Fig. 4(b). With crib-sheet, participants were required to touch the screen (on the empty space where there were no UI elements like buttons, icons or menus) with the thumb as a delimiter while performing a valid intentional posture/gesture to execute the displayed task. In addition, the crib-sheet was always available to participants, allowing them to access it at any time by tapping on the "Help" button at the top of the screen — mimicking the traditional ways to seek help in an application. The users required to scroll through the list to check all the postures-gestures.

5.1.2. Context-Aware-Hints

Our implementation of Context-Aware-Hints was inspired by the YouTube (Android App) where the app shows possible tasks (e.g., pause, next) with highlighted buttons when tapping on the screen. We adapted their design in our Context-Aware-Hints by showing a limited set of available postures or gestures and associated tasks based on the application context. Each of the tasks appeared at the top of the screen (Fig. 4(c)). Our design of context-aware hints was inspired by the recent work of Fennedy et al. (2021), where they provided visual guidance whenever users performed any gesture. Similar to their work, the context-aware hints technique only appears when users perform a gesture, allowing them to leverage the hints to guide their learning and performance of the correct gestures. To invoke the Context-Aware-Hints, participants had to touch the screen where there were no UI widgets, and the application displayed available postures and gestures based on the context with their corresponding tasks (4(d)) as an overlay on the screen for 4s. Based on a pilot study, we determined the 4-second time duration to keep the hints on display, allowing participants sufficient time to review the hints and perform the required postures/gestures. Additionally, automatically hiding the hints after 4 s eliminates the need for users to perform additional gestures. The users can wait for the overlay to fade away or issue postures and gestures immediately while the overlay is still there. Thus, the Context-Aware-Hints were visually hidden; however, were always available to become visible to users and guide them to recall and learn a gesture/posture to interact with the application.

5.2. Gestures-postures and tasks mapping

To mitigate the potential learning effect, we utilized two distinct sets of gestures and postures for the two guidance techniques. In this context, we also aimed to investigate any performance differences between the gesture-posture sets designed by expert and non-expert designers. Therefore, we opted to use two different sets of gesture-posture-task mappings to compare the performance of gesture-posture-task mappings designed by expert and non-expert designers, with the

Fig. 4. (a–b) Crib-Sheet displays all the postures, gestures, and associated tasks. A user can access them by pressing the "help" button; (c–d) Context-Aware-Hints shows available postures-gestures and associated tasks based on the application context. A user can trigger the guidance by touching on the screen; (e) Postures/gestures and corresponding task set for media player app (*Set 1*); (f) Postures/gestures and corresponding task set for a map app (*Set 2*).

findings from this comparison potentially providing insights for future directions in designing gestures and postures for specific tasks, taking into account the background of the designers. We selected a media player and a map app to investigate the two guidance techniques. To create mappings between gestures/postures and frequently used tasks with these apps, we reviewed a list of related prior works (Sridhar et al., 2017; Arefin Shimon et al., 2016; Ruiz et al., 2011; Ferron et al., 2019; Je et al., 2018) and observed the following frequently used tasks: (i) opening an app (e.g., map, media player), (ii) changing volume levels (e.g., 30%, 60%, 100%), (iii) navigate between media (e.g., next or previous song), (iv) changing brightness levels (e.g., 30%, 60%, 100%) and (v) workspace manipulation (e.g., zoom in, zoom out) etc. We also found that though there are studies exploring mappings between gestures and tasks (Ruiz et al., 2011; Ferron et al., 2019), none of the prior work explored how users' performance changes based on how the mappings were created (e.g., created by design expert or general user). To this end, we created two sets of mappings: (Set 1): created by a group of design experts and (Set 2): assigned by us.

5.2.1. Set 1

To create (Set 1), we conducted a design workshop (Je et al., 2018; Ali et al., 2018; Wobbrock et al., 2005) by recruiting eight Computer Science graduate students (3 PhD and 5 MSc; 5 male) aged 25–30 (M : 27.25, SD : 1.56) from four different universities. All the participants have a degree in design and took courses in Interface and/or interaction design. We conducted two zoom sessions where in the first session, we provided them with a PowerPoint file containing all gesture-posture images from the *Final Posture-Gesture Set* and a list of tasks that we discussed above. We then asked them to select a task and place a posture or gesture image that they believed was suitable for the task. This session lasted about 40 min.

After this session, we reviewed 35 mappings received from the participants and removed duplicate mappings that resulted in 24 unique mappings for eight activities. The second Zoom session was conducted with the same participants to refine the set further. In this session, we provided them with a Qualtrics form containing all the mapping. We asked them to provide their preference for the mappings with a 5-point Likert scale. We asked the participants to provide their preferences on the mapping between posture/gesture and eight activities as listed in Fig. 4(e). More specifically, they were asked to provide their preference ratings for the mapping between the posture/gesture and the corresponding activities — which were collected on a scale of 5, where 5 represented the most preferred and 1 represented the least preferred. This session took around 15 min.

We selected the patterns (e.g., single posture, sequence of postures and single gesture) with the highest mean ratings to generate one set of posture-gesture and task mappings. As this set mostly contains the posture-gesture and task mappings related to the media player app, we considered using those posture-gesture and task mappings for *Set 1*. Fig. 4(e) shows *Set 1*; the posture-gesture mappings for the media player app.

5.2.2. Set 2

This set contains 8 posture-gesture and task mappings which is done by two co-authors. Fig. 4(f) shows *Set 2* for a map application. During the assignment, we discussed tasks and possible gestures or posture that can be mapped. We assigned a gesture or postures once we mutually agreed on the posture or gestures for a task.

5.3. Participants and apparatus

We recruited 12 participants (7 male, 5 female) ages between 22 and 33 (M=26.67, SD=3.01). Each study session lasted approximately 45 min and participants received CA$15 for their participation. We used the smartwatch application that we developed to identify the posture/gesture while using the LEMFO LEM14 smartwatch to conduct this study. We invited the participants to come to a research lab and wear the smartwatch to perform the tasks for the study.

5.4. Tasks and procedure

Our tasks involved participants using two apps: a media player and a map and performing a set of tasks. We used the two posture/gesture-task *Set* that we discussed above, one for each app. Each *Set* contains eight postures/gestures-tasks pairs, totaling 16 postures/gestures-task combinations. *Set* 1 (Fig. 4(e)) holds 8 postures/gestures-task set for media player app and *Set* 2 (Fig. 4(f)) contains another 8 postures/gestures-task set for map app. We followed the procedure as used in prior work (Bau and Mackay, 2008; Delamare et al., 2016; Fennedy et al., 2021) where the study session was divided into four phases: pre-test, main-phase (a sequence of alternating training and testing trials), post-test and subjective feedback.

5.4.1. Pre-test

This phase aims to investigate if the posture/gesture-task mappings are known to the participants even before seeing them for the first time. This is done to establish a recall baseline. In this phase, the participants go through testing trials for each of the 16 postures/gestures without any guidance technique or feedback in random order and are not given any feedback on the correctness of posture and gestures. Since the participants are unaware of the postures/gestures-tasks mappings, the accuracy is 0% similar to the prior works (Bau and Mackay, 2008; Delamare et al., 2016; Fennedy et al., 2021). Note that all the gestures were different than what we traditionally use e.g., pinch gesture with index and thumb fingers for zooming in and out of a map. Instead, in our gestures set, users always had to keep the thumb on the smartwatch screen and use the remaining fingers for gestures. This creates a different gesture set than the traditional one. We use this accuracy value as the baseline to understand the recall rate better.

5.4.2. Main-phase

This phase aims to train the participants through alternating sequences of training and testing trials. There are three blocks of trials in this phase — each with a sequence of training trials followed by a sequence of testing trials. Each training sequence has 16 trials, eight postures/gestures, each with 2 repetitions, randomized within each block. Each testing sequence has eight trials — for each of the postures/gestures they are just trained, however, without any guidance techniques. During the main-phase trials, the participants completed tasks using a guidance technique, either with the crib-sheet or the context-aware-hints. While one set of postures/gestures (8 postures/gestures) was used with one guidance technique, another set of postures/gestures was used with another guidance technique alternatively that we managed through counterbalancing.

5.4.3. Post-test

This phase examines the overall learning and recall after all the main-phase trials. Similar to the pre-test phase, in this phase, participants completed test trials for all 16 postures/gestures in a random order without any guidance technique. This phase also measures participants' performance in recalling and performing tasks regarding execution time and accuracy.

After completing all the pre-test, main-phase, and post-test sessions, the participants provided subjective feedback for each guidance technique. They rated the techniques based on ease of learning, ease of recall, speed of learning, the accuracy of learning and recall, and comfort using a 7-point Likert scale in an online Qualtrics form. The participants also chose between the techniques they preferred most.

Participants started with a pre-test phase, followed by three blocks of sequential training and testing trials in the main phase; and finished with post-test trials. A trial starts when the participants touch the screen with the thumb and it ends when the participant releases the touch after performing the posture/gesture. The total execution time for each block denotes the time required to complete all the trials in that block. Accuracy for each block was measured by the number of times the participant performed the postures/gestures correctly while completing that block.

Participants' performance was evaluated in terms of execution time (i.e., measures total execution time for each block) and accuracy (i.e., measures the number of incorrect postures/gestures) for each guidance technique. For each trial in a block, the trial execution time includes the time for reading the task, performing the corresponding posture/gesture, checking the help from guidance techniques (in training trials) and recalling the posture/gesture (in testing trials).

In both training and testing trials, whenever the participant performed a correct posture/gesture, the task accomplishment message was shown as visual feedback and the next task appeared. In case of incorrect postures/gestures, "Incorrect gesture" was shown. During testing trials, participants completed the tasks without any guidance.

We used a $2 \times 2 \times 3$ within-subjects design for the factors *Guidance technique* (Crib-Sheet, Context-Aware-Hints), *Set* (Set1, Set2), and *Block* (Block1, Block2, Block3). The different set was paired with a different *Guidance technique* over alternating participants. We did this so to avoid any confounding effect of *Set* on *Guidance technique*. During the main phase, each participant completes a total of 96 training trials (2 *Guidance technique* $\times$ 3 *Block* $\times$ 8 postures/gestures $\times$ 2 repetitions) and 48 testing trials (2 *Guidance technique* $\times$ 3 *Block* $\times$ 8 postures/gestures $\times$ 1 repetition). In the post-test phase, each participant completes 16 testing trials (16 postures/gestures $\times$ 1 repetition). Fig. 5 shows the overall design for this discoverability and learability study.

5.5. Results

To analyze measures (e.g., execution time and accuracy), we used repeated-measures ANOVA with pairwise t-tests with Bonferroni corrections for post-hoc comparisons. We used Wilcoxon Signed-Rank tests to analyze subjective ratings.

5.5.1. Execution time

We examined task execution times of each block of both training and testing trials.

Training Time: Results from repeated-measure ANOVA showed *Guidance technique* has no main effect on Training Time ($F_{1,11} = 1.77, p = 0.21$, $\eta_P^2 = 0.14$). The mean training time across all blocks of Context-Aware-Hints ($M = 57.48$ s) was lower than Crib-Sheet was ($M = 65.12$ s) as shown in Fig. 6a. However, we found significant main effect of *Block* ($F_{2,22} = 10.41$,p< 0.001, $\eta_P^2 = 0.49$) on training times. Pairwise comparisons show that participants were significantly faster in Block3 ($M = 53.52$ s) compared to Block2 ($M = 62.19$ s, $\Delta = 8.67$ s, [1.24 s, 16.11 s]) and Block1 ($M = 68.16$ s, $\Delta = 14.64$ s, [2.89 s, 26.39 s]). No interaction between *Guidance technique* and *Block* was found. *Testing Time:* For testing trials, both *Guidance technique* ($F_{1,11} = 8.38, p = 0.02$, $\eta_P^2 = 0.43$) and Block ($F_{3,33} = 4.49, p = 0.01$, $\eta_P^2 = 0.29$) has main effect on execution time. The participants were 3.10s faster with Context-Aware-Hints $M = 26.39$ s) than Crib-Sheet ($M = 29.48$ s) across all the testing trials (Fig. 6(b)). Pairwise comparisons showed that participants were significantly faster in Block3 ($M = 24.05$ s) compared to Block2 ($M = 28.07$ s, $\Delta = 4.02$ s, [.65 s, 7.39 s]) and Block1 ($M = 29.55$ s, $\Delta = 5.05$ s, [.68 s, 10.33 s]). The accuracy increased significantly from the test Block3 ($M = 24.05$ s) to the post-test block ($M = 30.07$ s, $\Delta = 6.02$ s). We noticed the participants were taking a long time in recalling the postures/gestures in post-test trials. No interaction between *Guidance technique* and *Block* was found.

5.5.2. Execution accuracy

We examined the execution accuracy of each block of both training trials and testing trials.

Training Accuracy: During training trials, Context-Aware-Hints was more accurate than Crib-Sheet by 1.74% (Fig. 6(c)). Results from repeated-measure ANOVA showed that there was a statistical main effect of *Guidance technique* ($F_{1,11} = 5.85, p = 0.04$, $\eta_P^2 = 0.35$) on overall Training Accuracy, with Context-Aware-Hints ($M = 99.13\%$) being more accurate than Crib-Sheet ($M = 97.39\%, \Delta = -1.74\%$, $[-3.32\%, -0.16\%]$). We also found statistical main effect of *Block* ($F_{2,22} = 6.12, p = 0.01$, $\eta_P^2 = 0.36$) on training accuracy. Pairwise comparisons showed that participants were more significantly accurate in Block3 ($M = 99.74\%$) compared to Block2 ($M = 98.70\%, \Delta = -1.04\%$, $[-2.70\%, 0.62\%]$) and Block1 ($M = 96.35\%, \Delta = -3.39\%$, $[-6.72\%, -0.05\%]$). There was no interaction between *Guidance technique* and *Block*.

Testing Accuracy: Repeated-measure ANOVA showed that both *Guidance technique* ($F_{1,11} = 27.39$,p< 0.001, $\eta_P^2 = 0.71$) and *Block* ($F_{3,33} = 6.25, p = 0.002$, $\eta_P^2 = 0.36$) have significant main effect on accuracy. Pairwise comparisons showed that participants were significantly more accurate with Context-Aware-Hints ($M = 94.79\%$) than Crib-Sheet ($M = 88.28\%, \Delta = -6.51\%, [-9.25\%, -3.77\%]$) (Fig. 6(d)). In addition, their accuracy was significantly high in Block3 ($M = 94.79\%$) compared to Block2 ($M = 91.67\%, \Delta = -3.13\%, [-8.91\%, 2.66\%]$) and Block1 ($M = 85.94\%, \Delta = -8.85\%, [-16.84\%, -0.87\%]$). The accuracy did not change significantly between the test Block3 ($M = 94.79\%$) and the post-test block ($M = 93.75\%, \Delta = -1.04\%$). There was no interaction between *Guidance technique* and *Block*.

5.5.3. Subjective feedback

After completing all the training and testing phases, each participant provided feedback on a 7-point Likert scale. Fig. 6e presents the participants' responses to each subjective measure for each guide. Results from the Wilcoxon Signed Rank tests showed that *Guidance techniques* elicited a statistically significant effect on both Ease of Learning ($Z = -3.130, p = 0.002$) and Ease of Recall ($Z = -2.213, p = 0.027$). Participants rated Context-Aware-Hints more favorably than Crib-Sheet for ease of learning ($Mean = 6.15, Median = 6.00$ vs $Mean = 4.00, Median = 4.00$, respectively) and ease of recall ($Mean =$

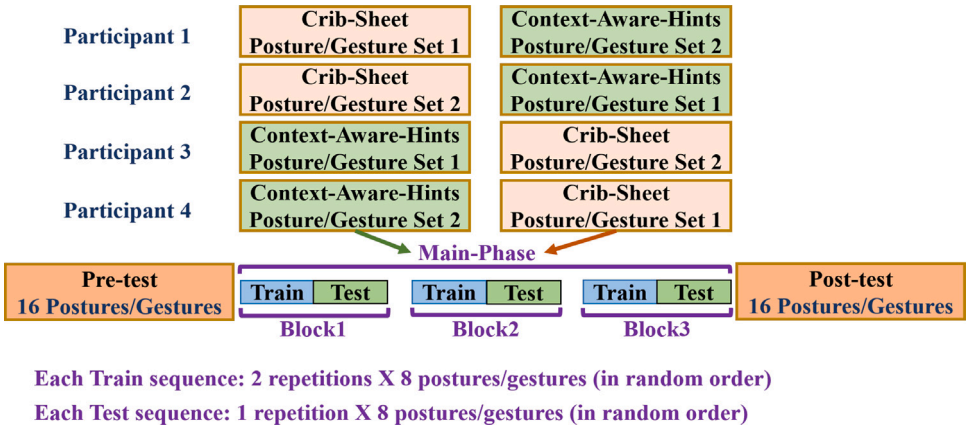


Fig. 5. Discoverability and Learnability study design.

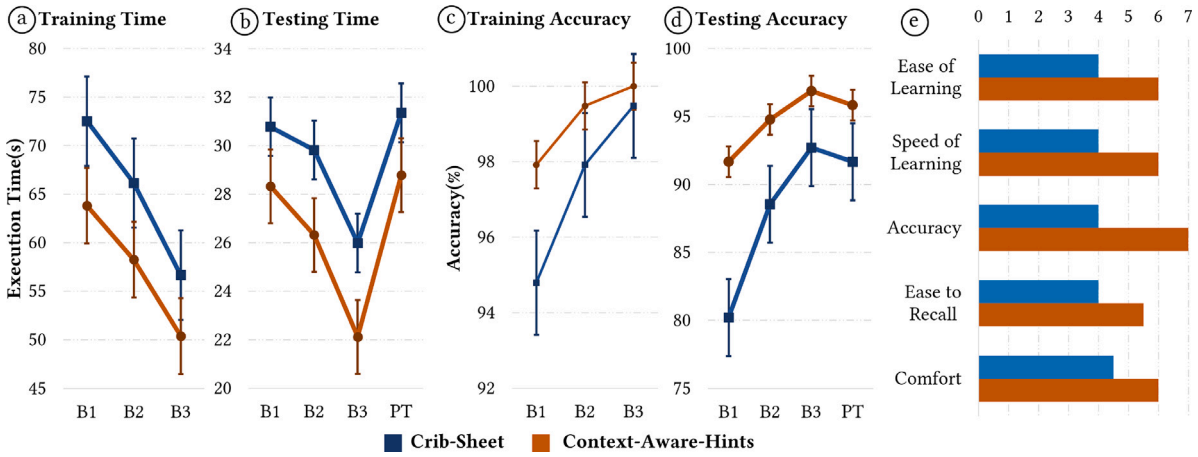


Fig. 6. Execution Times for (a) Training Blocks and (b) Testing Blocks, Execution Accuracy (c) Training Blocks and (d) Testing Blocks for each *Guidance technique*. (e) Subjective feedback (B = Block and PT = Post Test).

5.42, $Median = 5.50$ vs. $Mean = 3.75, Median = 4.00$, respectively). We also found that *Guidance techniques* also has a statistically significant effect on both Speed ($Z = -2.831, p = 0.005$) and Accuracy of learning ($Z = -3.097, p = 0.002$). Participants rated speed of learning with Context-Aware-Hints ($Mean = 5.50, Median = 6.00$) higher than Crib-Sheet ($Mean = 3.58, Median = 4.00$). Similarly, we observed a higher participant rating for the accuracy of learning with Context-Aware-Hints ($Mean = 6.58, Median = 7.00$) than Crib-Sheet ($Mean = 4.00, Median = 4.00$). Although the *Guidance techniques* did not show any statistically significance of Comfort ($Z = -1.437, p = 0.151$); the participants were in favor of Context-Aware-Hints ($Mean = 5.50, Median = 6.00$) than Crib-Sheet ($Mean = 4.67, Median = 4.50$). Finally, 9 out of 12 participants selected Context-Aware-Hints as their preferred technique than the Crib-sheet.

5.5.4. Discussion

We explore the crib-sheet and the context-aware hints based on prior works (Bau and Mackay, 2008; Delamare et al., 2016; Fennedy et al., 2021; Anderson and Bischof, 2013; Kurtenbach et al., 1994). However, there is a fundamental difference between these two techniques. The crib-sheet is more like traditional "help" feature that shows the list of all the postures-gestures regardless of the application context. Then the user requires additional clicks to open the list and scroll through it to see all the postures and gestures. On the other hand, the context-aware hints automatically show only the postures and gestures that are available based on the application's current context. Therefore, Crib-sheet requires users to take additional actions such as scrolling through the list compared to the context-aware hints. Therefore, both techniques guide users to discover and learn the available postures and gestures in two different ways. We conducted a study to explore which technique can better guide users in learning the postures and gestures quickly. In our discoverability and learnability study, participants learned through these techniques during the main phase, and in the post-test phase, they executed tasks without any guidelines based on their memory. Results from this study indicate that the participants learnt the postures and gestures with Context-Aware-Hints more quickly and accurately than Crib-Sheet. Moreover, the results of the post-test measures demonstrate better recall for Context-Aware-Hints than the Crib-sheet. Subjective feedback also showed better user ratings for Context-Aware-Hints than Crib-Sheet. Thus our study provides evidence that displaying only application-relevant postures-gestures improves learning while additional steps like scrolling can detract from learning.

6. Discussion and design guidelines

The small form factor of the smartwatch screen limits the available input space, resulting in limited interaction capabilities on the device (Van Vlaenderen et al., 2015; Yeo et al., 2016; Han et al., 2017; Schirra and Bentley, 2015; Knibbe et al., 2014). Consequently, researchers (Seyed et al., 2016; Knibbe et al., 2014; Van Vlaenderen et al., 2015; Yeo et al., 2016; Han et al., 2017; Ahlström et al., 2018) explored around device interaction for extending the interaction space beyond the watch – however, primarily through external instrumentation (McIntosh et al., 2019; Lim et al., 2015; Sridhar et al., 2017) – which might not be practical in many cases (e.g., outdoor) and limit

the applicability of these solutions for everyday use (Yeo et al., 2016). Leveraging users' finger input (e.g., finger gestures/postures) around the device to extend input capabilities on an unmodified commodity smartwatch can be a potential solution to this issue. Therefore, we started our exploration to find suitable user-preferred finger gestures and postures that can be detected through the smartwatch's built-in cameras on the commodity smartwatch.

We first reviewed several prior related works (Dim and Ren, 2014; Delamare et al., 2015; Seyed et al., 2016; Lu et al., 2020; Zhu et al., 2018; Ruiz et al., 2011; Arefin Shimon et al., 2016) that leverages finger postures and gestures for smart device interaction. However, we found that none of these prior works include any delimiter for the postures and gestures. Prior work (Markussen et al., 2014; Chen et al., 2014) showed that around-device interactions benefit from an explicit delimiter to avoid detecting any unintentional finger movements. Therefore, our work extended the prior work by incorporating delimiter options (e.g., index or thumb on the screen) into around-device finger postures and gestures that can be tracked with an unmodified commodity smartwatch.

We observed participants' positive attitudes towards using both mid-air and back-of-the-palm spaces, which guides us in leveraging both spaces for around-device interaction. Thus, we continued our exploration, further investigating user-preferred finger postures and gestures that are detectable by watches' built-in cameras into the spaces. Interestingly, we found that there were more postures and gestures (i.e., five postures, three gestures) than back-of-the-palm (i.e., three postures, two gestures) participants preferred with the Thumb On-screen as the delimiter. We believe this is primarily due to the large mid-air space available to participants where they can move their fingers to perform gestures and postures rather than limited the back-of-the-palm space. We also observed that finger postures and gestures on the back-of-the-palm showed a lower classification accuracy. When observing participants, we found that while positioning multiple fingers on the back of the palm, the finger at the front causes occlusion to the rest of the fingers and thus lowers the classification accuracy. We also observed similar issues with other camera-based systems (e.g., Leap Motion) where finger occlusions lower the gesture and posture detection accuracy (Sridhar et al., 2015; Sharp et al., 2015; Kim et al., 2012).

Another common concern with posture/gesture-based interaction is that they are not self-revealing like graphical buttons and menus (Baudel and Beaudouin-Lafon, 1993). Users need to explicitly discover, learn, and memorize those postures and gestures (Bau and Mackay, 2008; Fennedy et al., 2021). Therefore, a discoverability mechanism or gesture guidance is critical to help users quickly discover, learn and memorize the possible postures and gestures with their associated activities. To our best knowledge, no prior works investigated the discoverability and learnability of postures and gestures for around-device interaction on smartwatches. Therefore, we also explored gesture guidance techniques for around-device finger postures and gestures based interaction on smartwatches ensuring learning through discoverability on unmodified commodity smartwatches.

Inspired by prior work (Bau and Mackay, 2008; Delamare et al., 2016; Fennedy et al., 2021), we explored Context-Aware-Hints, where a limited set of postures and gestures and associated tasks are displayed based on the application contexts. We further compared it with Crib-Sheet which displays a static list of postures-gestures and associated commands. Similar to prior works (Bau and Mackay, 2008; Fennedy et al., 2021; Delamare et al., 2016), we showed evidence that showing only application-relevant gestures improves learning. Our study findings align with those of Fennedy et al. (2021), which suggest that context-aware hints are significantly more accurate than crib-sheet in training users to perform postures and gestures more precisely. Additionally, both Bau and Mackay (2008) and Fennedy et al. (2021) found that both training and recall time were lower for context-aware hints compared to crib-sheet. Our study results further confirm the lower training and recall time for context-aware hints compared to crib-sheet. We believe that the Context-Aware-Hints is always available to visually guide users based on the application context, thus minimizing the cognitive load for memorizing the postures-gestures and the associated tasks. Consequently, our results demonstrated that participants could easily discover, learn and recall the postures/gestures to interact with the smartwatch with Context-Aware-Hints.

Here we summarize our main findings and present the possible design considerations of our exploration of around-device interaction along with learning guidance on commodity smartwatches. The results from our studies offer the following guidelines to the designers of around-device smartwatch interaction:

Interaction space and Gesture-Posture: Our study results revealed that the participants rated both mid-air and the back of the palm favorably as smartwatch interaction spaces. Seventeen out of thirty participants rated mid-air as their preferred around-device input space while thirteen participants preferred the back of the palm. Therefore, we suggest considering both mid-air and the back of the palm as extended interaction space while designing around-device interaction on the smartwatch. As indicated in the paper, finger activities in mid-air can be captured with the front camera, whereas activities on the back of the palm can be captured with the side camera on the device. We found that participants had an equal preference for finger postures and gestures. Thus, finger gestures and posture should be considered as around-device smartwatch input.

Delimiter: Results from the design space exploration study indicated that the participants preferred touch on the screen over any posture and gesture as a delimiter. Prior research (Markussen et al., 2014; Chen et al., 2014) also suggested touch on the screen as an intuitive delimiter. Therefore, we considered touch on the screen with different fingers (e.g., index, thumb) as possible delimiter options while using gestures and postures with smartwatches for further investigation. Participants preferred touching the screen with the thumb as the delimiter to indicate the beginning of a valid intentional posture or gesture. Designers should consider this form of interaction as a potential delimiter option for starting any mid-air interaction.

Using comfortable gestures/postures: We found that some postures were not comfortable to perform due to awkward finger combinations and their positions. For instance, placing the index, middle and ring finger straight and the pinky finger bent while the thumb is touching the screen was very uncomfortable for most participants. We suggest designers only consider gestures or postures that are comfortable and not awkward. In addition, we need to ensure that comfortable finger gestures and postures are possible to track with the device's camera while avoiding finger occlusions.

Guidance technique: Crib-Sheet used a static list of possible postures-gestures and the command associated with each gesture (Fig. 4(b)) whereas Context-Aware-Hints only showed the available postures and gestures and their associated tasks based on the application context (Fig. 4(d)). Through our exploration of guidance techniques, we found that Context-Aware-Hints assisted users to learn the postures and gestures more accurately and quickly than Crib-Sheet. Additionally, Context-Aware-Hints assisted users by providing step-by-step visual guidance for tasks based on an application context, which eventually helps the users to discover and learn the postures/gestures naturally and more easily. Therefore, we suggest including Context-Aware-Hints as a guidance technique for gesture or posture-based interfaces to support learnability through discoverability.

7. Limitations and future work

In this paper we present a proof-of-concept solution for extending the input capabilities of commodity smartwatches with expressive finger posture/gesture on the back of the palm and in mid-air while

ensuring learning the interaction through discoverability. However, future work can address some existing limitations we discuss here. For instance, we found that delimiters for gesture-based interaction have not been explored thoroughly for smartwatches. Therefore, we used touch on the screen as a delimiter as suggested in Markussen et al. (2014) and Chen et al. (2014) as well as based on participants' subjective feedback during the first study. However, Future work can explicitly explore suitable delimiters for gesture-based interaction on smartwatches — while considering both users' performance and their feedback.

All the training data for the deep-learning classifier model were collected and tested in a lab environment. As our focus was to demonstrate proof-of-concept around-device interaction capabilities on an unmodified commodity smartwatch, we moved forward with the settings. We believe that a large set of training images captured in different conditions (e.g., outdoor, lighting) with any advanced deep learning algorithms will further show higher classification accuracy for different environments and usage scenarios. Future work can collect more data with different conditions (e.g., lighting, outdoor) to improve the model's overall accuracy. In addition, we did not explore how users' attributes (e.g., finger shape, skin color) can have an impact on the detection accuracy which can be investigated further. We have not investigated the performance of the system for different mobility conditions (i.e. standing, walking, and running). Future work can evaluate the design and performance of the system for different mobility conditions.

We implemented and deployed our deep-learning detection model in the Google Colab server as the smartwatches have limited processing capability. In addition, deep learning models are usually very computationally heavy for handheld devices. As standard smartphone and smartwatch applications leverage external servers for processing and data storage (e.g., Firebase), we deployed our deep-learning model on the Google Colab server. Future work can explore ways to incorporate the deep-learning model into smartwatches. Further investigation can explore lightweight deep-learning models (e.g., MobileNets Howard et al., 2017) to embed them into smartwatches. We noticed that placing more than one finger on back-of-palm occludes other fingers, resulting in lower posture detection accuracy. Participants also expressed that adjusting the fingers to avoid occlusion is a bit uncomfortable. Future work can explore ways to detect multi-finger posture/gesture on the back of the palm reliably using advanced deep-learning models.

Although results indicate that Context-Aware-Hints supported participants to learn the postures/gestures more accurately and quickly, Crib-sheet also showed comparable performance. A few participants found Context-Aware-Hints to be annoying as the hints popped up every time they attempted to perform a posture/gesture. They found crib-sheet as a simpler guidance technique. Therefore, future work can explore other potential simple guidance techniques like crib-sheet. For instance, we can explore ways to redesign buttons in an expressive way (e.g., embedding hand posture/gesture images into the button) to incorporate hints for the next hand actions. We acknowledge that our proposed system is suitable for smartwatches with built-in cameras and there are a limited number of smartwatches with the same feature (e.g., LEMFO LEM14, ZEBLAZE THOR 6, Kospet Prime S). However, the current trend of having multiple cameras on smartphones indicates that we will most likely see multiple cameras integrated into a vast range of smartwatches in the near future.

8. Conclusion

In this paper, we have presented a set of user-preferred around-device finger postures and gestures in both mid-air and back of the palm accompanied by a delimiter to be used for interacting with smartwatches. We have also demonstrated a camera-based proof-of-concept prototype system that is capable of detecting finger gestures and postures on around the device (mid-air and back of the palm) to extend the interaction capability of the commodity smartwatches with built-in cameras. The postures and gestures are not self-revealing and the users need to explicitly discover, learn, and memorize them to use further. Therefore, we have finally explored "Context-Aware-Hints"; through which the user can easily discover and learn those finger postures-gesture and their corresponding tasks easily and quickly. We believe the user-preferred finger posture-gesture-based interaction along with context-aware learning guidance can be an effective and unique solution to pushing the boundary of current around-device interactions for commodity smartwatches.

CRediT authorship contribution statement

Marium-E- Jannat: Conceptualization, Methodology, Implementation, Data-Analysis, Writing – original draft, Writing – review & editing . **Xing-Dong Yang:** Conceptualization, Supervision. **Khalad Hasan:** Conceptualization, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Xing-Dong Yang, the second author of this paper, is one of the Associate Editors of this journal.

Data availability

Data will be made available on request

Acknowledgments

We thank all the participants for their time and feedback. This research was funded by a Natural Sciences and Engineering Research Council (NSERC) Discovery Grant (RGPIN-2019-05211). This project was undertaken with the financial support of the Government of Canada. Ce projet a été réalisé avec l'appui financier du gouvernement du Canada.

References

Abdulla, W., 2017. Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. GitHub Repository, https://github.com/matterport/Mask_RCNN.

Ahlström, D., Hasan, K., Lank, E., Liang, R., 2018. TiltCrown: Extending input on a smartwatch with a tiltable digital crown. In: Proceedings of the 17th International Conference on Mobile and Ubiquitous Multimedia. In: MUM 2018, Association for Computing Machinery, New York, NY, USA, pp. 359–366. http://dx.doi.org/10.1145/3282894.3289726.

Ali, A.X., Morris, M.R., Wobbrock, J.O., 2018. Crowdsourcing similarity judgments for agreement analysis in end-user elicitation studies. In: Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology. UIST '18, Association for Computing Machinery, New York, NY, USA, pp. 177–188. http://dx.doi.org/10.1145/3242587.3242621.

Anderson, F., Bischof, W.F., 2013. Learning and performance with gesture guides. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '13, Association for Computing Machinery, New York, NY, USA, pp. 1109–1118. http://dx.doi.org/10.1145/2470654.2466143.

Arefin Shimon, S.S., Lutton, C., Xu, Z., Morrison-Smith, S., Boucher, C., Ruiz, J., 2016. Exploring non-touchscreen gestures for smartwatches. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems. CHI '16, Association for Computing Machinery, New York, NY, USA, pp. 3822–3833. http://dx.doi.org/10.1145/2858036.2858385.

Bau, O., Mackay, W.E., 2008. OctoPocus: A dynamic guide for learning gesture-based command sets. In: Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology. UIST '08, Association for Computing Machinery, New York, NY, USA, pp. 37–46. http://dx.doi.org/10.1145/1449715.1449724.

Baudel, T., Beaudouin-Lafon, M., 1993. Charade: remote control of objects using free-hand gestures. Commun. ACM 36 (7), 28–35.

Bendersky, M., Wang, X., Metzler, D., Najork, M., 2017. Learning from user interactions in personal search via attribute parameterization. In: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining. WSDM '17, Association for Computing Machinery, New York, NY, USA, pp. 791–799. http://dx.doi.org/10.1145/3018661.3018712.

Boring, S., Ledo, D., Chen, X.A., Marquardt, N., Tang, A., Greenberg, S., 2012. The fat thumb: Using the thumb's contact size for single-handed mobile interaction. In: Proceedings of the 14th International Conference on Human-Computer Interaction with Mobile Devices and Services. MobileHCI '12, Association for Computing Machinery, New York, NY, USA, pp. 39–48. http://dx.doi.org/10.1145/2371574.2371582.

Chan, L., Chen, Y.-L., Hsieh, C.-H., Liang, R.-H., Chen, B.-Y., 2015. CyclopsRing: Enabling whole-hand and context-aware interactions through a fisheye ring. In: Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology. UIST '15, Association for Computing Machinery, New York, NY, USA, pp. 549–556. http://dx.doi.org/10.1145/2807442.2807450.

Chen, X.A., Schwarz, J., Harrison, C., Mankoff, J., Hudson, S.E., 2014. Air+touch: Interweaving touch & in-air gestures. In: Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology. UIST '14, Association for Computing Machinery, New York, NY, USA, pp. 519–525. http://dx.doi.org/10.1145/2642918.2647392.

Christoudias, C.M., Saenko, K., Morency, L.-P., Darrell, T., 2006. Co-adaptation of audio-visual speech and gesture classifiers. ICMI '06, Association for Computing Machinery, New York, NY, USA, pp. 84–91. http://dx.doi.org/10.1145/1180995.1181013.

Delamare, W., Coutrix, C., Nigay, L., 2015. Designing guiding systems for gesture-based interaction. In: Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems. EICS '15, Association for Computing Machinery, New York, NY, USA, pp. 44–53.

Delamare, W., Janssoone, T., Coutrix, C., Nigay, L., 2016. Designing 3D gesture guidance: Visual feedback and feedforward design options. In: Proceedings of the International Working Conference on Advanced Visual Interfaces. AVI '16, Association for Computing Machinery, New York, NY, USA, pp. 152–159. http://dx.doi.org/10.1145/2909132.2909260.

Dim, N., Ren, X., 2014. Designing motion gesture interfaces in mobile phones for blind people. J. Comput. Sci. Tech. 29, 812–824. http://dx.doi.org/10.1007/s11390-014-1470-5.

Dutta, A., Gupta, A., Zissermann, A., 2016. VGG image annotator (VIA). http://www.robots.ox.ac.uk/~vgg/software/via/. (Accessed 30 August 2021).

Fennedy, K., Hartmann, J., Roy, Q., Perrault, S.T., Vogel, D., 2021. OctoPocus in VR: Using a dynamic guide for 3D mid-air gestures in virtual reality. IEEE Trans. Vis. Comput. Graphics 27 (12), 4425–4438.

Ferron, M., Mana, N., Mich, O., 2019. Designing mid-air gesture interaction with mobile devices for older adults. In: Sayago, S. (Ed.), Perspectives on Human-Computer Interaction Research with Older People. Springer International Publishing, Cham, pp. 81–100. http://dx.doi.org/10.1007/978-3-030-06076-3_6.

Gil, H., Lee, D., Im, S., Oakley, I., 2017. TriTap: Identifying finger touches on smartwatches. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, pp. 3879–3890. http://dx.doi.org/10.1145/3025453.3025561.

Goguey, A., Malacria, S., Gutwin, C., 2018. Improving discoverability and expert performance in force-sensitive text selection for touch devices with mode gauges. CHI '18, Association for Computing Machinery, New York, NY, USA, pp. 1–12. http://dx.doi.org/10.1145/3173574.3174051.

Gong, J., Xu, Z., Guo, Q., Seyed, T., Chen, X.A., Bi, X., Yang, X.-D., 2018. WrisText: One-handed text entry on smartwatch using wrist gestures. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, pp. 1–14. http://dx.doi.org/10.1145/3173574.3173755.

Gong, J., Yang, X.-D., Irani, P., 2016. WristWhirl: One-handed continuous smartwatch input using wrist gestures. In: Proceedings of the 29th Annual Symposium on User Interface Software and Technology. UIST '16, Association for Computing Machinery, New York, NY, USA, pp. 861–872. http://dx.doi.org/10.1145/2984511.2984563.

Google ATAP (Advanced Technology and Projects), 2021b. Soli. https://atap.google.com/soli/. (Accessed 10 January 2022).

Google Colab, 2019. Google colaboratory. https://research.google.com/colaboratory/faq.html. (Accessed 15 January 2022).

Google LLC, 2022. YouTube (Android App). https://play.google.com/store/apps/details?id=com.google.android.youtube. (Accessed 15 May 2022).

Grossman, T., Hinckley, K., Baudisch, P., Agrawala, M., Balakrishnan, R., 2006. Hover widgets: Using the tracking state to extend the capabilities of pen-operated devices. CHI '06, Association for Computing Machinery, New York, NY, USA, pp. 861–870. http://dx.doi.org/10.1145/1124772.1124898.

Gupta, A., Balakrishnan, R., 2016. DualKey: Miniature screen text entry via finger identification. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, pp. 59–70. http://dx.doi.org/10.1145/2858036.2858052.

Han, J., Ahn, S., Park, K., Lee, G., 2017. Designing touch gestures using the space around the smartwatch as continuous input space. In: Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces. ISS '17, Association for Computing Machinery, New York, NY, USA, pp. 210–219. http://dx.doi.org/10.1145/3132272.3134134.

Hayashi, E., Lien, J., Gillian, N., Giusti, L., Weber, D., Yamanaka, J., Bedal, L., Poupyrev, I., 2021. RadarNet: Efficient gesture recognition technique utilizing a miniature radar sensor. In: Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. CHI '21, Association for Computing Machinery, New York, NY, USA, http://dx.doi.org/10.1145/3411764.3445367.

He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision. ICCV, pp. 2961–2969.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. CVPR, pp. 770–778.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H., 2017. MobileNets: Efficient convolutional neural networks for mobile vision applications. http://dx.doi.org/10.48550/ARXIV.1704.04861, arXiv, URL: https://arxiv.org/abs/1704.04861.

Jannat, M.-E., Vo, T.T., Hasan, K., 2022. Face-centered spatial user interfaces on smartwatches. In: Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems. In: CHI EA '22, Association for Computing Machinery, New York, NY, USA, http://dx.doi.org/10.1145/3491101.3519720.

Je, S., Lee, M., Kim, Y., Chan, L., Yang, X.-D., Bianchi, A., 2018. PokeRing: Notifications by poking around the finger. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. CHI '18, Association for Computing Machinery, New York, NY, USA, pp. 1–10. http://dx.doi.org/10.1145/3173574.3174116.

Khan, M.A.A.H., Roy, N., Misra, A., 2018. Scaling human activity recognition via deep learning-based domain adaptation. In: 2018 IEEE International Conference on Pervasive Computing and Communications. PerCom, pp. 1–9. http://dx.doi.org/10.1109/PERCOM.2018.8444585.

Kim, J., He, J., Lyons, K., Starner, T., 2007. The gesture watch: A wireless contact-free gesture based wrist interface. In: 2007 11th IEEE International Symposium on Wearable Computers. pp. 15–22. http://dx.doi.org/10.1109/ISWC.2007.4373770.

Kim, D., Hilliges, O., Izadi, S., Butler, A.D., Chen, J., Oikonomidis, I., Olivier, P., 2012. Digits: Freehand 3D interactions anywhere using a wrist-worn gloveless sensor. In: Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology. UIST '12, Association for Computing Machinery, New York, NY, USA, pp. 167–176. http://dx.doi.org/10.1145/2380116.2380139.

Klamka, K., Horak, T., Dachselt, R., 2020. Watch+Strap: Extending smartwatches with interactive StrapDisplays. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, pp. 1–15. http://dx.doi.org/10.1145/3313831.3376199.

Knibbe, J., Martinez Plasencia, D., Bainbridge, C., Chan, C.-K., Wu, J., Cable, T., Munir, H., Coyle, D., 2014. Extending interaction for smart watches: Enabling bimanual around device control. In: CHI '14 Extended Abstracts on Human Factors in Computing Systems. In: CHI EA '14, Association for Computing Machinery, New York, NY, USA, pp. 1891–1896. http://dx.doi.org/10.1145/2559206.2581315.

Kospet, 2021. Kospet Prime S Smartwatch. Retrieved December 30, 2021 from https://www.amazon.in/TimeTech-Cameras-Smartwatch-1050mAh-Healthcare/dp/B09FKBJS6V.

Kurtenbach, G., Moran, T.P., Buxton, W., 1994. Contextual animation of gestural commands. In: Computer Graphics Forum, Vol. 13, No. 5. 13, (5), Wiley Online Library, pp. 305–314.

LEMFO, 2021. LEMFO LEM14 Smartwatch. Retrieved December 30, 2021 from https://www.lemfo.com/.

Lim, H., Chung, J., Oh, C., Park, S., Lee, J., Suh, B., 2018. Touch+finger: Extending touch-based user interface capabilities with "Idle" finger gestures in the air. In: Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology. UIST '18, Association for Computing Machinery, New York, NY, USA, pp. 335–346. http://dx.doi.org/10.1145/3242587.3242651.

Lim, S.-C., Shin, J., Kim, S.-C., Park, J., 2015. Expansion of smartwatch touch interface from touchscreen to around device interface using infrared line image sensors. Sensors 15 (7), 16642–16653.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft COCO: Common Objects in Context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (Eds.), Computer Vision – ECCV 2014. Springer International Publishing, Cham, pp. 740–755. http://dx.doi.org/10.1007/978-3-319-10602-1_48.

Lu, Y., Huang, B., Yu, C., Liu, G., Shi, Y., 2020. Designing and evaluating hand-to-hand gestures with dual commodity wrist-worn devices. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 4 (1).

Lyons, K., 2020. Wearable magnetic field sensing for finger tracking. In: Proceedings of the 2020 International Symposium on Wearable Computers. ISWC '20, Association for Computing Machinery, New York, NY, USA, pp. 63–67. http://dx.doi.org/10.1145/3410531.3414304.

Markussen, A., Jakobsen, M.R., Hornbæk, K., 2014. Vulture: A mid-air word-gesture keyboard. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '14, Association for Computing Machinery, New York, NY, USA, pp. 1073–1082. http://dx.doi.org/10.1145/2556288.2556964.

McIntosh, J., Strohmeier, P., Knibbe, J., Boring, S., Hornbæk, K., 2019. Magnetips: Combining fingertip tracking and haptic feedback for around-device interaction. In: Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, pp. 1–12. http://dx.doi.org/10.1145/3290605.3300638.

Nascimento, T.H., Soares, F., 2020. WatchControl: A control for interactive movie using continuous gesture recognition in smartwatches. J. Inf. Process. 28, 643–649.

Nascimento, T.H., Soares, F.A.A.M.N., Nascimento, H.A.D., Vieira, M.A., Carvalho, T.P., Miranda, W.F.d., 2019. Netflix control method using smartwatches and continuous gesture recognition. In: 2019 IEEE Canadian Conference of Electrical and Computer Engineering. CCECE, pp. 1–4. http://dx.doi.org/10.1109/CCECE.2019.8861610.

Park, K., Kim, D., Heo, S., Lee, G., 2020. MagTouch: Robust finger identification for a smartwatch using a magnet ring and a built-in magnetometer. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, pp. 1–13. http://dx.doi.org/10.1145/3313831.3376234.

Park, K., Lee, G., 2019. FingMag: Finger identification method for smartwatch. In: Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems. In: CHI EA '19, Association for Computing Machinery, New York, NY, USA, pp. 1–6. http://dx.doi.org/10.1145/3290607.3312982.

Perrault, S., Malacria, S., Guiard, Y., Lecolinet, E., 2012. Watchit: Simple gestures for interacting with a watchstrap. In: CHI '12 Extended Abstracts on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, pp. 1467–1468. http://dx.doi.org/10.1145/2212776.2212489.

Pietroszek, K., Tahai, L., Wallace, J.R., Lank, E., 2017. Watchcasting: Freehand 3D interaction with off-the-shelf smartwatch. In: 2017 IEEE Symposium on 3D User Interfaces. 3DUI, pp. 172–175. http://dx.doi.org/10.1109/3DUI.2017.7893335.

Reyes, G., Wu, J., Juneja, N., Goldshtein, M., Edwards, W.K., Abowd, G.D., Starner, T., 2018. SynchroWatch: One-handed synchronous smartwatch gestures using correlation and magnetic sensing. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 1 (4).

Ruiz, J., Li, Y., Lank, E., 2011. User-defined motion gestures for mobile interaction. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '11, Association for Computing Machinery, New York, NY, USA, pp. 197–206. http://dx.doi.org/10.1145/1978942.1978971.

Schirra, S., Bentley, F.R., 2015. "It's kind of like an extra screen for my phone": Understanding everyday uses of consumer smart watches. In: Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems. In: CHI EA '15, Association for Computing Machinery, New York, NY, USA, pp. 2151–2156. http://dx.doi.org/10.1145/2702613.2732931.

Seyed, T., Yang, X.-D., Vogel, D., 2016. Doppio: A reconfigurable dual-face smartwatch for tangible interaction. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems. CHI '16, Association for Computing Machinery, New York, NY, USA, pp. 4675–4686. http://dx.doi.org/10.1145/2858036.2858256.

Sharp, T., Keskin, C., Robertson, D., Taylor, J., Shotton, J., Kim, D., Rhemann, C., Leichter, I., Vinnikov, A., Wei, Y., Freedman, D., Kohli, P., Krupka, E., Fitzgibbon, A., Izadi, S., 2015. Accurate, robust, and flexible real-time hand tracking. CHI '15, Association for Computing Machinery, New York, NY, USA, pp. 3633–3642. http://dx.doi.org/10.1145/2702123.2702179.

Shorten, C., Khoshgoftaar, T.M., 2019a. A survey on image data augmentation for deep learning. J. Big Data 6 (1), 1–48.

Siek, K.A., Rogers, Y., Connelly, K.H., 2005. Fat finger worries: How older and Younger users physically interact with PDAs. In: Costabile, M.F., Paternò, F. (Eds.), Human-Computer Interaction. INTERACT 2005, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 267–280.

Sodhi, R., Benko, H., Wilson, A., 2012. LightGuide: Projected visualizations for hand movement guidance. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI '12, Association for Computing Machinery, New York, NY, USA, pp. 179–188. http://dx.doi.org/10.1145/2207676.2207702.

Sridhar, S., Markussen, A., Oulasvirta, A., Theobalt, C., Boring, S., 2017. WatchSense: On- and above-skin input sensing through a wearable depth sensor. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, pp. 3891–3902. http://dx.doi.org/10.1145/3025453.3026005.

Sridhar, S., Mueller, F., Oulasvirta, A., Theobalt, C., 2015. Fast and robust hand tracking using detection-guided optimization. In: Proceedings of Computer Vision and Pattern Recognition. CVPR, URL: http://handtracker.mpi-inf.mpg.de/projects/FastHandTracker/.

Sun, K., Wang, Y., Yu, C., Yan, Y., Wen, H., Shi, Y., 2017. Float: One-handed and touch-free target selection on smartwatches. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, pp. 692–704. http://dx.doi.org/10.1145/3025453.3026027.

Trotta, S., Weber, D., Jungmaier, R.W., Baheti, A., Lien, J., Noppeney, D., Tabesh, M., Rumpler, C., Aichner, M., Albel, S., Bal, J.S., Poupyrev, I., 2021. 2.3 SOLI: A tiny device for a new human machine interface. In: 2021 IEEE International Solid-State Circuits Conference, Vol. 64. ISSCC, pp. 42–44. http://dx.doi.org/10.1109/ISSCC42613.2021.9365835.

Van Vlaenderen, W., Brulmans, J., Vermeulen, J., Schöning, J., 2015. WatchMe: A novel input method combining a smartwatch and bimanual interaction. In: Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems. In: CHI EA '15, Association for Computing Machinery, New York, NY, USA, pp. 2091–2095. http://dx.doi.org/10.1145/2702613.2732789.

Wang, H., Zhan, X., Liu, L., Ullah, A., Li, H., Gao, H., Wang, Y., Li, G., 2021. Unsupervised cross-user adaptation in taste sensationrecognition based on surface electromyography withconformal prediction and domain regularizedcomponent analysis. arXiv preprint arXiv:2110.11339.

Wen, H., Ramos Rojas, J., Dey, A.K., 2016. Serendipity: Finger gesture recognition using an off-the-shelf smartwatch. In: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems. Association for Computing Machinery, New York, NY, USA, pp. 3847–3851. http://dx.doi.org/10.1145/2858036.2858466.

Wobbrock, J.O., Aung, H.H., Rothrock, B., Myers, B.A., 2005. Maximizing the guessability of symbolic input. In: CHI '05 Extended Abstracts on Human Factors in Computing Systems. In: CHI EA '05, Association for Computing Machinery, New York, NY, USA, pp. 1869–1872. http://dx.doi.org/10.1145/1056808.1057043.

Xia, H., Grossman, T., Fitzmaurice, G., 2015. NanoStylus: Enhancing input on ultra-small displays with a finger-mounted stylus. In: Proceedings of the 28th Annual ACM Symposium on User Interface Software AndTechnology. UIST '15, Association for Computing Machinery, New York, NY, USA, pp. 447–456. http://dx.doi.org/10.1145/2807442.2807500.

Xu, C., Pathak, P.H., Mohapatra, P., 2015. Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch. In: Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications. HotMobile '15, Association for Computing Machinery, New York, NY, USA, pp. 9–14. http://dx.doi.org/10.1145/2699343.2699350.

Yang, Y., Chae, S., Shim, J., Han, T.-D., 2015. EMG sensor-based two-hand smart watch interaction. In: Adjunct Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology. In: UIST '15 Adjunct, Association for Computing Machinery, New York, NY, USA, pp. 73–74. http://dx.doi.org/10.1145/2815585.2815724.

Yeo, H.-S., Lee, J., Bianchi, A., Quigley, A., 2016. WatchMI: Pressure touch, twist and pan gesture input on unmodified smartwatches. In: Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services. MobileHCI '16, Association for Computing Machinery, New York, NY, USA, pp. 394–399. http://dx.doi.org/10.1145/2935334.2935375.

ZEBLAZE, 2021. ZEBLAZE THOR 6 Smartwatch. Retrieved December 30, 2021 from https://zeblaze.info/thor6.html.

Zhang, C., Bedri, A., Reyes, G., Bercik, B., Inan, O.T., Starner, T.E., Abowd, G.D., 2016a. TapSkin: Recognizing on-skin input for smartwatches. In: Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces. ISS '16, Association for Computing Machinery, New York, NY, USA, pp. 13–22. http://dx.doi.org/10.1145/2992154.2992187.

Zhang, C., Yang, J., Southern, C., Starner, T.E., Abowd, G.D., 2016b. WatchOut: Extending interactions on a smartwatch with inertial sensing. In: Proceedings of the 2016 ACM International Symposium on Wearable Computers. ISWC '16, Association for Computing Machinery, New York, NY, USA, pp. 136–143. http://dx.doi.org/10.1145/2971763.2971775.

Zhou, J., Zhang, Y., Laput, G., Harrison, C., 2016. AuraSense: Enabling expressive around-smartwatch interactions with electric field sensing. In: Proceedings of the 29th Annual Symposium on User Interface Software and Technology. UIST '16, Association for Computing Machinery, New York, NY, USA, pp. 81–86. http://dx.doi.org/10.1145/2984511.2984568.

Zhu, K., Fjeld, M., Ünlüer, A., 2018. WristOrigami: Exploring foldable design for multi-display smartwatch. In: Proceedings of the 2018 Designing Interactive Systems Conference. DIS '18, Association for Computing Machinery, New York, NY, USA, pp. 1207–1218. http://dx.doi.org/10.1145/3196709.3196713.

Zhu, P., Zhou, H., Cao, S., Yang, P., Xue, S., 2018b. Control with gestures: A hand gesture recognition system using off-the-shelf smartwatch. In: 2018 4th International Conference on Big Data Computing and Communications. BIGCOM, pp. 72–77. http://dx.doi.org/10.1109/BIGCOM.2018.00018.