HandyTrak: Recognizing the Holding Hand on a Commodity Smartphone from Body Silhouette Images

Hyunchul Lim, David Lin, Jessica Tweneboah, Cheng Zhang
SciFi Lab, Cornell University
Information Science, Cornell University
Ithaca, New York, United States
{hl2365,dxl2,jnt42,chengzhang}@cornell.edu

ABSTRACT

Understanding which hand a user holds a smartphone with can help improve the mobile interaction experience. For instance, the layout of the user interface (UI) can be adapted to the holding hand. In this paper, we present HandyTrak, an AI-powered software system that recognizes the holding hand on a commodity smartphone using body silhouette images captured by the front-facing camera. The silhouette images are processed and sent to a customized user-dependent deep learning model (CNN) to infer how the user holds the smartphone (left, right, or both hands). We evaluated our system on each participant's smartphone at five possible front camera positions in a user study with ten participants under two hand positions (in the middle and skewed) and three common usage cases (standing, sitting, and resting against a desk). The results showed that HandyTrak was able to continuously recognize the holding hand with an average accuracy of 89.03% (SD: 8.98%) at a 2 Hz sampling rate. We also discuss the challenges and opportunities to deploy HandyTrak on different commodity smartphones and potential applications in real-world scenarios.

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile devices**.

KEYWORDS

Mobile computing, Computer vision, Commodity smartphone

ACM Reference Format:
Hyunchul Lim, David Lin, Jessica Tweneboah, Cheng Zhang. 2021. HandyTrak: Recognizing the Holding Hand on a Commodity Smartphone from Body Silhouette Images. In *The 34th Annual ACM Symposium on User Interface Software and Technology (UIST '21), October 10–14, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3472749.3474817

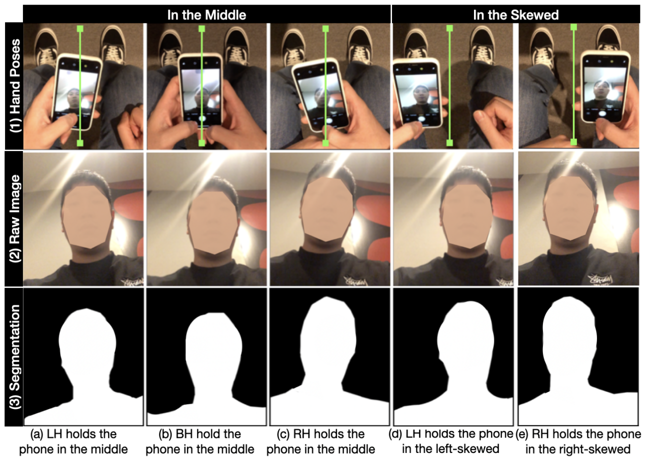

In the Middle
In the Skewed
(1) Hand Poses
(2) Raw Image
(3) Segmentation
(a) LH holds the phone in the middle
(b) BH hold the phone in the middle
(c) RH holds the phone in the middle
(d) LH holds the phone in the left-skewed
(e) RH holds the phone in the right-skewed

Figure 1: The research idea of HandyTrak. The position of the body (e.g., head and shoulder lines) in segmented image frames is differs depending on whether the phone is held in a user's (a) left hand, (b) right hand, (c) both hands, (d) tilted to the left, or (e) tilted to the right.

1 INTRODUCTION

Understanding which hand people hold their smartphones with is increasingly gaining importance in mobile interface design since a user's hand mode has a direct impact on their mobile user experience [26, 41]. This is because the "comfortable" phone screen areas for users' fingers differ depending on their hand mode [2, 12, 24] as shown in Figure 2.a. For instance, when holding smartphones, the thumbs cannot be easily stretched to reach certain parts of a mobile screen (e.g., middle to top). In such cases, users are forced to make an effort to touch the "further" areas (e.g., changing hand grip or tilting the device), which leads to a sub-optimal user experience. Furthermore, the gain in popularity of larger screen sizes in commercial mobile devices creates an even bigger user experience issue as it makes it even more difficult to touch certain parts of the screen with the thumb.

To address this issue, commercial companies have developed techniques such as Apple's Reachability feature [16], one-handed mode [18], and one-handed typing [15] as shown in Figure 2.b. These techniques shrink the UI and then shift it to the "comfortable" range of the left or right thumb, making it easier for users to access the "further" parts of the screen. However, explicit user input is required as activation to determine the holding hand, and the shrunk UI would need to be manually readjusted as users switch their hands when operating.

Researchers have identified more ideal solutions that automatically detect how a smartphone is being held and then tailor the

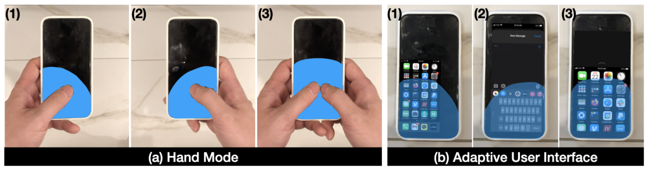


Figure 2: Hand Mode and Adaptive User Interface. Comfortable screen areas for the left hand (a.1), right hand (a.2), and both hands (a.3). Fig. 2b shows an adaptive user interface that is tailored to the user's hand mode (left-handed (b.1), right-handed (b.2), both hands (b.3))

UI accordingly. Previous research utilized interaction data such as touch events, swipes, and/or Inertial Measurement Unit (IMU) sensor values, to automatically detect hand mode [8, 9, 39]. However, these techniques require explicit input from the user (e.g., touching the phone, motion) before they are able to recognize the holding hand, which makes them not suitable in cases where adaptive UI is needed before any touch interaction. For example, with these techniques, one-handed mode (See Figure 2.b.1) cannot be activated immediately after turning on or unlocking the phone. Other research [1, 29] introduced techniques to determine hand mode before interacting with a phone by using the user's unlocking behavior (e.g., pin, swipe, pattern unlock, and IMU sensor stream). Still, these techniques can detect hand mode only during the phone unlock stage and cannot handle continuous hand switching scenarios during actual phone use. Also, they might be less useful in that unlock methods are gradually shifting to biometric unlock mechanisms such as facial recognition.

For continuous hand mode tracking, several researchers have proposed hand mode detection methods that work before the user interacts with the screen. These methods can provide a continuously adaptive UI depending on the holding hand, which helps to enrich the mobile interaction experience [25, 35]. However, all of these methods either require external sensors (e.g., additional capacitive sensors along the sides of the phone case [21], a self-capacitance touchscreen [11], or an additional device (e.g., a smartwatch [11]), which means that they are not immediately available for commodity smartphones. Thus, we believe that there is a need for a new passive sensing approach that can continuously track the holding hand mode on commodity smartphones without explicit input from the user.

In this paper, we introduce HandyTrak, an AI-powered software system that can continuously track the holding hand on a commodity smartphone using body silhouette images captured by the built-in front camera. The key idea of our system is that the position of the body (e.g., head and shoulder lines) in image frames is relatively different depending on whether the phone is held in a user's left, right, or both hands (See Figure 1). We believe these differences are highly informative in training a deep learning model to classify which hand is holding the phone. To implement this idea, we developed a customized user-dependent deep learning classification model (CNN) to infer the holding hand from the images captured by the front camera, and we evaluated the system in a user study with ten participants. The results showed that our system was able to continuously recognize three hand modes (left, right, both hands) at 2Hz, with an average of 89.03% (SD: 8.98%) accuracy under three scenarios (standing, sitting, and resting against a desk). Specifically, the system achieved an average of 95.84% (SD: 5.0%) accuracy when participants held and operated their phones in a left- or right-skewed position.

Though HandyTrak can continuously track the holding hand with a promising accuracy, this feature may not always be necessary for real-world applications due to the strain on the phone battery. As a result, the main goal of HandyTrak is to allow designers and developers of mobile user interfaces to acquire the holding hand information at any time without explicitly requiring input from the user. For example, HandyTrak can be integrated into the face ID [17] unlock feature of phones, which uses the front camera to take pictures of the user. In this case, HandyTrak can provide holding hand information using the images captured from the unlocking event, without collecting any other additional information. Additionally, HandyTrak can activate one-handed typing mode based on the user's hand mode after opening the keyboard. Moreover, it can be leveraged on the camera app to provide an adaptive virtual button that continuously adapts to the fingers of the holding hand when taking selfies, which would be especially helpful when taking selfies at odd angles. To verify how HandyTrak performs under these opportunistic sensing moments, we used images captured at the moments when the 10 participants were using three applications: 1) unlocking the device, 2) browsing an app, and 3) taking a selfie. The results showed that HandyTrak was able to recognize the holding hand with average accuracies of 87.56% (SD: 10.31%), 88.37% (SD: 6.28%), and 92.68% (SD: 6.47%) for the above three application scenarios respectively. Furthermore, we implemented the system in real-time on a commodity mobile phone and discuss possible opportunities and challenges when using our system in real-world scenarios such as walking scenario evaluation, user-independent model, and privacy issues.

The contributions of our paper are the following:

- We present a new method that leverages body silhouette images captured by the front camera of a commodity smartphone to continuously track hand mode.
- We evaluate our system under various categories: the five different positions of the front camera, two hand postures (in the middle and the skewed postures), and three scenarios of using the device (standing, sitting, and resting toward a desk).
- We implemented our system on a mobile phone and discuss possible issues and solutions of deploying our system in a real-world application.

2 RELATED WORK

This section reviews the literature related to the detection of hand mode and hand grips by leveraging various sensors in a smartphone and/or additional hardware. Table 1 summarizes hand mode tracking techniques.

With the commodity smartphone, researchers have explored solutions that automatically detect how a smartphone is being held. Researchers proposed methods that involved the use of sensors on the commodity smartphone. Goel et al [9] developed Gripsense,

Table 1: Summary of hand mode tracking techniques. HandyTrak is the first technology that can continuously track the holding hand without requiring any prior user interaction on a commodity smartphone. *LH: Left-handed, RH: Right-handed, BH: Both hands, L-Th: sLeft thumb, R-Th: Right thumb, L-In: Left index finger, R-In: Right index finger

Related Work	Sensing Method	Additional Hardware	Hand Mode	Classification Accuracy	Sensing Moment	Explicit Input	User Dependence
Gripsense [9]	IMU & Touch	-	*LH, RH, BH	84.3%	After touch	Needed	Independent
ContextType [8]	IMU & Touch	-	LH, RH, BH	89.7%	After touch	Needed	Independent
Park et al [39]	IMU & Touch	-	L-Th, R-Th, L-In, R-In	92.4%	After touch	Needed	Independent
Lochtefeld et al [29]	Touch	-	LH, RH, BH	98.5%	During unlock	Needed	Independent
Je Avery et al [1]	IMU	-	L-Th, R-Th	83.6%	During unlock	Needed	Dependent
Kim et al [21]	Capacitive	Needed	8 hand grips	99.1%	Continuous	-	Independent
Hinckley et al [11]	Self-capacitance	Needed	Hand grips	N/A	Continuous	-	Independent
WhichHand [28]	IMUs	Needed	LH, RH	97.0%	Continuous	-	Independent
HandyTrak	**Camera**	-	**LH, RH, BH**	**89.0%**	**Continuous**	-	**Dependent**

which differentiates between different hand postures (one- or two-handed interaction, use of thumb or index finger). Gripsense achieved this using touch events and IMU sensor values while users performed five touch interaction steps and attained an accuracy of 84.3%. Goel et al also introduced ContextType [8], an adaptive text entry system that improves mobile touch screen text entry, by utilizing IMU sensor values for an average of 4.9 touch events to infer a user's hand posture (one- or two-handed thumb or index finger interaction), doing so with an accuracy of 89.7%. Park et al [39] also made use of the smartphone's touchscreen and its built-in gyroscope and accelerometer to distinguish between five and four hand poses (left thumb, right thumb, left index finger, right index finger, and/or both thumbs) with 87.7% and 92.4% accuracies, respectively. Also, Lochtefeld et al [29] introduced a technique to determine handedness (left or right-handed) using the user's unlocking behavior (pin, swipe, and pattern unlock) and achieved a high 98.51% accuracy. They achieved this using a k-nearest neighbor comparison of the internal sensor readings of the smartphone during the unlocking process. Finally, Avery et al [1] proposed a method to detect hand mode (left- or right-handed) by analyzing the built-in orientation sensor stream as the phone is lifted and unlocked before the user touches the screen with 83.6% accuracy. The limitation with these works is that they require explicit input from the user [8, 9, 39] or can only provide hand mode tracking at limited scenarios [1, 29] such as unlocking the phone. In other words, these works cannot continuously track the holding hand at any moment without explicit input from the user.

To address these issues, some researchers have proposed methods to continuously track how people are holding their smartphones by using additional hardware. Kim et al [21], for example, placed an array of capacitive sensors underneath the phone case to detect eight different hand grip patterns (including single-handed, two-handed, horizontal, vertical), with an accuracy of 99.1%. Hinckley et al [11] developed a self-capacitance touchscreen that employed sensors on the touchscreen display to recognize multiple hovering fingers and determined accurately whether the smartphone was being held with both hands or one hand. Whichhand [28] uses the relationship of orientation sensors from a smartphone and a smartwatch to automatically detect hand mode (left- or right-handed), showing an accuracy of 97.0%. These approaches, however, require external sensors or an additional device to be implemented, which do not make them immediately available on commodity smartphones.

In contrast, HandyTrak is the first technology that can continuously track the holding hand without requiring any prior user interaction on a commodity smartphone.

3 DESIGN OF HANDYTRAK

Advances in computer vision and in mobile phones offer an opportunity to utilize the built-in front camera to create new mobile interactions beyond taking photos and videos [31, 33, 45, 48]. Our system, HandyTrak, is an AI-powered software system that leverages the built-in front camera in a commodity smartphone to detect which hand is holding the phone for providing an adaptive user interface.

The idea of our system is inspired by the camera relocalization problem [3, 7, 19, 23, 27] in robotics and computer vision where a camera position is estimated from the images captured by the camera. Likewise, our work infers which hand a user holds the smartphone with by employing the user's body images taken by the built-in front camera. As shown in Figure 1.3, captured body silhouettes are different depending on whether the phone is held in a user's left, right, or both hands. For example, the length of the shoulder line and the position of the head differs depending on the position of the phone while a user holding and operating it. Furthermore, the difference between the camera angles when the phone is held in the left arm versus when held in the right becomes more apparent when the phone is tilted to either to the left or right (see Figure 1.e and f). We believe that these differences are more than informative enough to train a deep learning model to classify hand mode.

To verify the feasibility of our new approach, we explore five hand positions under three common mobile phone use scenarios [13, 26] as shown in Figure 3. Also, we investigate the effect of the built-in camera locations on our system since the location of the front-facing camera in the phone varies by manufacturer. Based

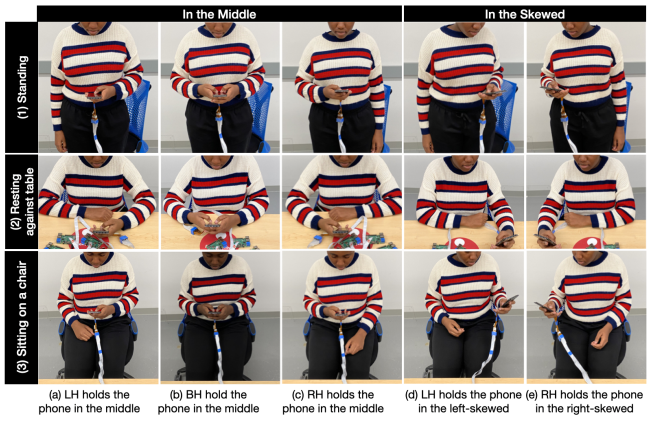

In the Middle
In the Skewed
(1) Standing
(2) Resting against table
(3) Sitting on a chair
(a) LH holds the phone in the middle
(b) BH hold the phone in the middle
(c) RH holds the phone in the middle
(d) LH holds the phone in the left-skewed
(e) RH holds the phone in the right-skewed

Figure 3: Experimental setup. Data collection was conducted under three common mobile use scenarios for the five different hand postures: (1) standing, (2) resting against a desk, and (3) sitting on a chair.

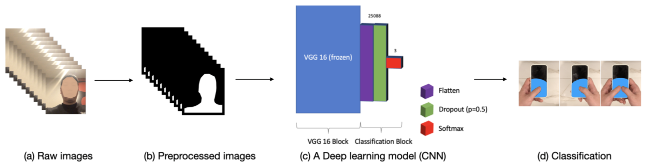


Figure 4: System Overview. Raw images of the user's upper body are preprocessed to produce segmented images. The segmented images are used as input to a VGG 16 model to perform classification to determine which hand(s) the user was using.

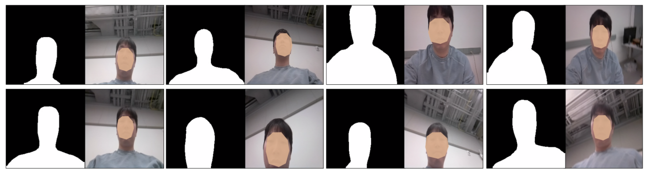


Figure 5: The output of segmented images using FCN-ResNet101 [30]. Raw images captured by the built-in front camera are processed and segmented from the background. Each pixel is labeled as either part of the human body (white), or as part of the background (black).

on these findings, we developed the following research questions (RQs) below:

- RQ1: Is it possible to continuously classify three types of hand mode (i.e., left, right, or both hands) by employing the body silhouette images captured by the front camera under three common scenarios of using the phone (i.e., standing, sitting, and resting toward a desk)?
- RQ2: Would the performance be better if the phone is held in a tilted way, towards the right or left rather than in the middle?
- RQ3: Will our recognition system work regardless of the five different camera positions?

To examine the research questions, we developed a deep learning classification model by leveraging computer vision techniques. Then, we conducted a user study to evaluate recognition accuracy and discuss the feasibility of deploying HandyTrak in real-world applications.

4 SYSTEM DESIGN

In this section, we present the implementation of HandyTrak, which consists of two parts: image preprocessing and a deep learning pipeline (see. Figure 4). First, we introduce human body segmentation techniques to get body silhouette images. Then, we describe the implementation of our deep learning model.

4.1 Image Preprocessing

The key idea of our system is to use the body silhouette images as input for our deep learning model. We first normalize the size of the input images to 224 × 224. Then, we segment the human body from the background. In this step, for each pixel, we decide whether it belongs to the background or the human body. Many previous machine learning systems have demonstrated reliable performance on this task [30, 34]. We decided to use a well-known pre-trained model named FCN-ResNet101 [30][1] as our image semantic segmentation method, which demonstrated promising performance in the paper and our experiments as shown in Figure 5. We used all data (even if the segmentation failed) for training and testing purposes.

4.2 Deep Learning Pipeline

These segmented body silhouette images are sent to a customized deep learning model to classify three modes of holding (left, right, and both hands). This model, called HandyNet, was developed on the pre-trained model VGG 16[2], which is known to perform well in image classification [42]. Our model consists of a VGG 16 backbone with a classification block behind it as shown in Figure 4.c. The classification block has a flatten layer, a dropout layer ($p = 0.5$) to reduce over-fitting [43], and a softmax layer for final outputs. The softmax layer outputs three probability values, determining left hand, right hand, or both hands. Before training, we freeze all VGG 16 layers.

Our model was trained for 2 epochs on the training set. We use a batch size of 32, with a categorical cross-entropy loss function and the Adam optimizer. The parameter ($p = 0.5$) of dropout and the batch size (32) were empirically decided. In total, there are 14,789,955 parameters in the model. There are 30.7 billion floating-point operations to perform inference on one image. This takes roughly 0.025 seconds on our server using the GPU (AMD Threadripper 3960X CPU and RTX2080Ti GPUs with 256GB memory).

[1]https://pytorch.org/hub/pytorch_vision_fcn_resnet101/
[2]https://keras.io/api/applications/vgg/

5 EVALUATION

To evaluate our system, we conducted a user study with ten participants. To simulate the different positions of the front cameras on different smartphones, we attached five miniature RGB cameras on the top of the participant's smartphone. Each participant was asked to interact with their smartphone under two hand positions (in the middle and in the skewed) and three common scenarios of using a smartphone (standing, sitting, and resting against a desk). We recorded the images on all five cameras which are used to classify which hand they hold the phone with.

5.1 Apparatus

Figure 6 shows our experiment setup consisting of five RGB cameras, a smartphone, and five Raspberry Pis. A camera module has a view angle of 64 × 48 degrees and fixed focus, allowing for the collection of images with 640 × 480 resolution at 30 FPS. Flexible printed circuit (FPC) cables are employed to connect the cameras and the Raspberry Pis. The five cameras were placed in a row on a black plastic sheet at intervals of 12 millimeters and the black sheet is attached to the participant's phone. We had each participant use their own phone in the study for two reasons. 1) A participant would be more comfortable using their own phone compared to using a new phone. 2) Using participants' phones would allow us to evaluate our system on a variety of phones of different dimensions. All participants reported that the attached flexible cable did not affect the experience with holding and operating the phone.

5.2 Participants

Ten participants (six females, two left-handed) ranging in age from 18 to 31 (M: 24.0, SD: 5.2) were recruited in a university. All male and four female participants had short hair while the rest of the female participants had long hair that touched their shoulders. Only eight participants had experience with using facial recognition for unlocking a smartphone. All participants had experience with operating their phones with left, right, and both hands.

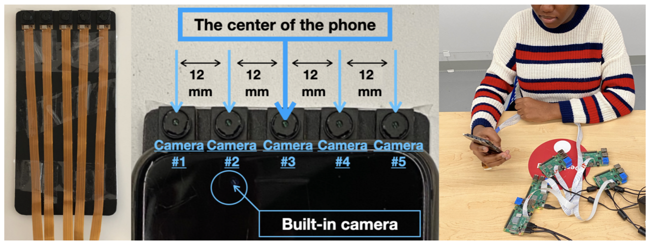


Figure 6: Device setup. Setup consists of five RGB cameras, a smartphone, and five Raspberry Pis. Flexible printed circuit (FPC) cables connect the camera module to the camera serial interface (CSI) of the Raspberry Pi. The five cameras are placed in a row on a black plastic sheet at intervals of 12 millimeters. The black sheet is attached to the participant's phone.

5.3 Procedure

We used a 3 × 2 within-subjects factorial design to collect images using front-facing cameras as shown in Figure 3. The data collection was conducted under three common mobile use scenarios [13]: standing, resting against a desk, and sitting on a chair. For each scenario, participants were asked to hold and operate the phone in two different ways: in the middle and in a skewed position. In total, participants were asked to hold their smartphones for five different hand poses: (a) left-hand (LH) in the middle, (b) both hands (BH) in the middle, (c) right-hand (RH) in the middle, (d) left-hand (LH) in left-skewed, and (e) right-hand (RH) in right-skewed. The order of the scenarios and the hand poses was randomized during the data collection.

For each hand pose, we asked participants to perform three common mobile tasks as they normally would: look at the phone for face ID unlock, open and browse through an application, and then take a selfie. Each round of the three tasks took about 6∼15 seconds to perform and was repeated at least 10 times in 3 minutes, which helped us to collect the images from various positions of the phone in hand poses. Also, since the participants were not aware of the study's purpose (detecting the holding hand) until the study was done, they were unlikely to perform the tasks in unnatural ways for the obtrusive cameras.

5.4 Data Collection

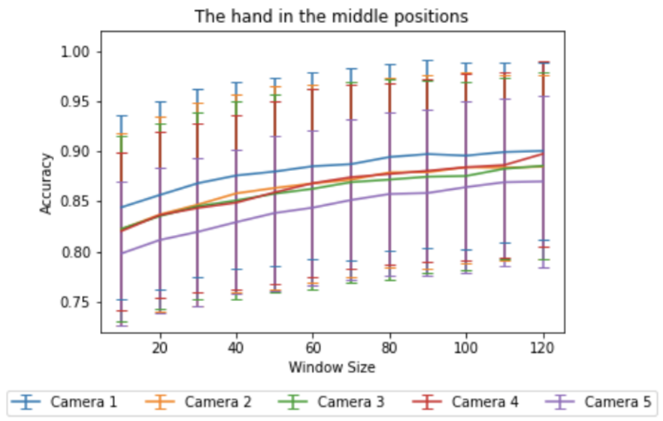
The RGB images from the five cameras were saved with the size of 244 × 244 in each Raspberry Pi while participants performed tasks with different hand poses under the three scenarios. A total of about 4M images were collected, including 810,000 per each camera position (30 fps × 60 seconds × 3 minutes × 5 hand poses × 3 scenarios × 10 participants). During the data collection, 804 images were lost from one participant's data due to technical issues (connection error on camera #5 from P6). All images except for the lost images were transferred to our main server for further analysis.

5.5 Validation Method

We built a user-dependent model for each participant and each camera (5 models per participant). For each participant, we first separated the data into 15 categories based on the hand positions and mobile phone usages scenarios as shown in Figure 3. Then, we split our data into a training set and a testing set. We use the first 80% of the data in each category for training the model and the last 20% of the data as the testing set. Please note that the collected data was not shuffled when splitting the training and testing set. For each camera, we used its own dataset for training and testing purposes.

5.6 Result

Overall, HandyTrak achieved a classification accuracy on hand mode, i.e., left, right, or both hands, with an average of 89.03% (SD: 8.98%) under two hand positions and three scenarios using five cameras at a 2Hz sampling rate. In the middle position, the average accuracies in the standing, sitting in a chair, and resting against a desk scenarios are 82.6%, 88.0%, and 82.9%, respectively. The range of accuracies across participants is 64.8%-94.6%, 64.4%-98.7%, and 53.8%-95.9% for each of the scenarios, respectively. For



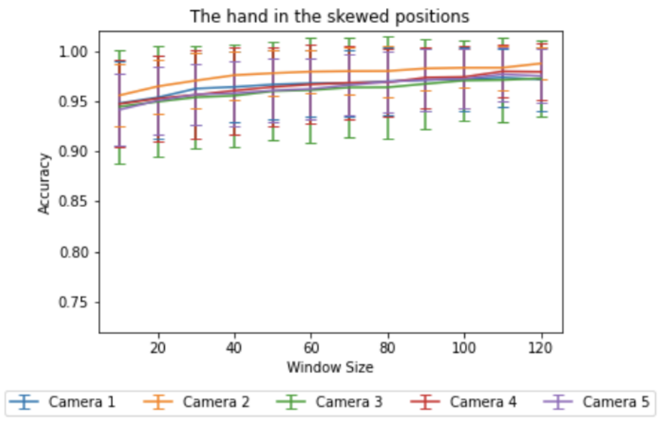


Figure 7: The effect of Window Size and Camera Positions. As the window size increases, the accuracy increases regardless of camera position. We choose a window size of 30 since the response time of one second is likely acceptable for mobile user interfaces [36].

the skewed position, the average accuracies are 96.2%, 94.6%, and 96.8% with ranges of 90.7%-100%, 85.5%-100%, and 85.9-100% across participants for each position, respectively.

In the following sections, we analyze our system performance in the following aspects: sliding window and majority vote, frame rate, camera positions, hand positions, and common mobile tasks.

5.6.1 Sliding window and Majority vote. The deep learning model we deployed makes a classification on each frame, which is not needed for most applications. Furthermore, temporal information can be used to improve recognition accuracy [5]. Therefore, we applied a sliding window with different sizes ranging from 10 (0.3 seconds) to 120 (4 seconds). There was a 50% overlap between windows. Within each window, the final recognition result of hand modes is based on the most common label over the individual classification results on frames. We examined two hand positions: in the middle and in the skewed. The results (see Figure 7) show that the accuracy improves as the window size gets bigger. The average accuracy over all positions and cameras when the window size was 10 was 87.2%. This increased to 92.3% when the window size was 120. However, the larger window means a less sensitive system is detecting the holding hand. Considering that a one-second response time is likely acceptable for mobile user interfaces [36], we decided to employ the window size of 30 for all of the following evaluations, which had an average accuracy of 89.1%.

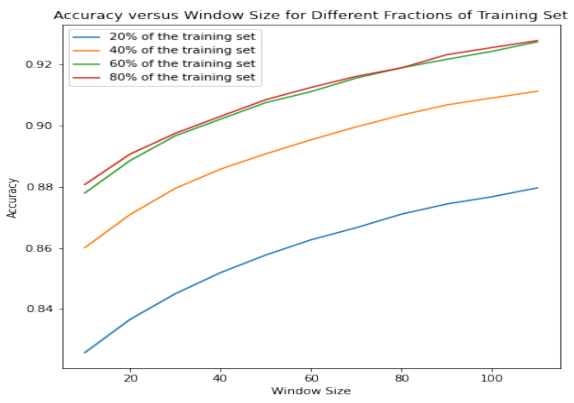


Figure 8: Training Data and Sliding Window Size Effect. HandyTrak can achieve comparable performance even with less training data if the window is large, which helps reduce the user's training effort to develop the model.

5.6.2 Training Data and Sliding Window Size Effect. We conducted a thorough analysis of how the amount of training data and the sliding window size would impact the performance. As shown in Figure 8, further experiments showed that when we only used 20% of the training data (3 minutes of training), HandyTrak still achieved accuracies of 84.5%, 88.2%, 90.92%, 91.83%, 93.15% with window sizes of 1 second, 4 seconds, 10 seconds, 30 seconds, and 60 seconds, respectively. When we use only 1 minute of the training data, the performance varies from 77.4% to 87.2% with window sizes of 1 second and 1 minute, respectively. If the application is less responsive to the response time, HandyTrack can achieve even better performance (over 90% with a window of 10 seconds) with larger sliding windows even with only 3 minutes of training data.

5.6.3 Frame Rate Effect. In our user study, we captured the pictures at a frame rate of 30. However, the frame rate has a considerable effect on power consumption [47]. Therefore, we further conducted experiments to understand how different frame rates would impact the performance of HandyTrak. In this experiment, the window size is the same, one second. We downsampled frame rate to 30Hz, 15Hz, 10Hz, 8Hz, 6Hz, 5Hz, 4Hz, and 3Hz. Figure 9 shows this effect. When we downsampled the framerate, we kept the 2Hz label update frequency and adjusted the number of images in each window accordingly.

At 30 frames per second, the average accuracy over all positions and cameras was 89.1%. When the frame rate was reduced to 3Hz,

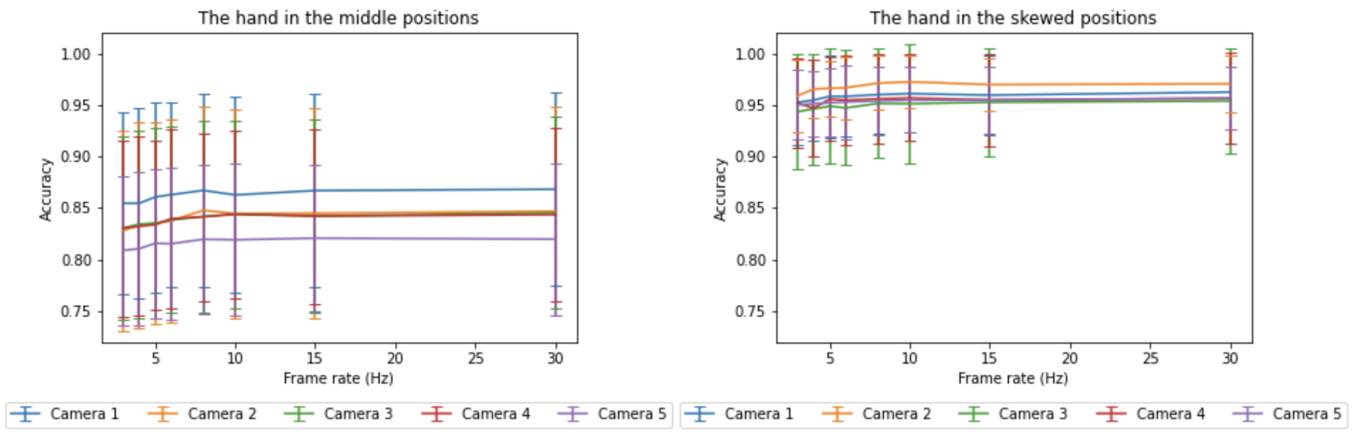


Figure 9: The effect of frame rate at a 2Hz sampling rate. Lower frame rates do not induce a significant drop in accuracy.

the average accuracy was 87.9%. This is a 90% decrease in frame rate, yet the average accuracy was only reduced by 1.4%, indicating that this setup can reduce battery consumption and remain accurate. In what follows, we report accuracy when sampling every frame to present the maximum performance of HandyTrak.

5.6.4 Five Camera Positions. Since the built-in front camera is positioned differently for different manufacturers, we investigated the effect of camera positions on our system under three scenarios for each hand position. The average accuracies of the five cameras (from left to right) were 90.1%, 89.6%, 88.9%, 88.9%, and 87.4%.

We found that there is no difference on performance between five camera positions and three scenarios for each hand position, respectively (middle: $F_{4,45}$ = 0.2689, p = 0.8964 | skewed: $F_{4,45}$ = 0.2151, p = 0.9286), indicating that our system could be applied to various types of smartphone models having different camera positions.

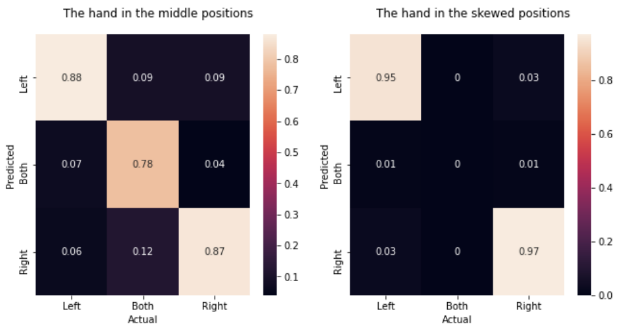


Figure 10: Confusion Matrix. For the skewed positions, we did not collect data where participants held the device with both hands. However, our model, which is trained on both middle and skewed position data, still predicts "Both" on skewed images. In the confusion matrix on the right, we insert zeros in the middle column because no images in the skewed positions have an actual label of "Both".

5.6.5 Hand Positions: Middle vs Skewed. We hypothesize that our system has better performance when the phone is in a skewed position (95.84% (SD: 5.05%)) rather than in the middle (84.49% (SD: 11.44%)) since the angle of the camera in a skewed pose makes bigger differences on body images. The result shows that HandyTrak achieved higher accuracy when participants use their smartphone in the skewed position rather than in the middle regardless of the three scenarios and five camera positions ($F_{1,90}$ = 61.98, p < 7.446e-12). Figure 10 and Figure 11 show the results under these two hand positions. They indicate that it was much harder to differentiate between hands when the device was held closer to the center of the body rather than to the side.

5.6.6 Opportunistic sensing for mobile applications. In opportunistic sensing of the holding hand, we explored how HandyTrak would perform when users are interacting with applications (i.e., unlocking, app browsing, and selfie). As we discussed in the introduction, HandyTrak allows for continuous tracking of the holding hand on commodity smartphones. In reality, most applications would not require continuous tracking of the holding hand as the user would not frequently change the holding hand, and taking and processing pictures can also take a lot of resources (e.g., energy). The goal of HandyTrak is to allow the developer and designer of mobile applications to acquire holding hand information at any time without explicitly requiring input from the user.

To verify this goal, we conducted further experiments to classify the holding hand only using the images captured under three application scenarios: unlocking moments, opening and browsing an application, and taking a selfie. We manually labeled the testing set (last 20%) into three application scenarios: 1) unlocking, 2) app browsing, and 3) selfie. On average, for each participant, we labeled 1735 unlocking frames, 7043 app browsing frames, and 4841 selfie frames. We used the model trained in the first experiment (first 80% of data for training) for the classification task. The window size is 1 second. On average, HandyTrack was able to recognize the holding hand with accuracies of 87.56% (SD: 10.31%), 88.37% (SD: 6.28%), and 92.68% (SD: 6.47%) for 1) unlocking, 2) app browsing, and 3) selfie, respectively. This result is encouraging, which demonstrates

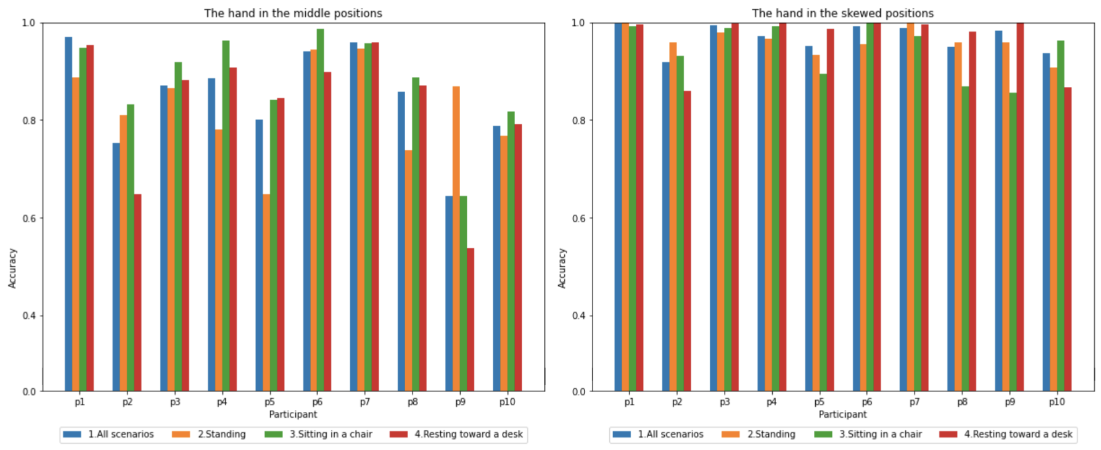


Figure 11: Result per participant under two positions: middle vs. skewed. The accuracy is better in the skewed positions for every participant.

the feasibility for mobile apps to acquire holding hand information when needed.

6 DISCUSSION AND LIMITATION

Our system, HandyTrak, is an AI-powered system that can continuously recognize hand mode by using a built-in camera in commodity smartphones. Here, we discuss possible applications and the opportunities and challenges of deploying HandyTrak in real-world scenarios.

6.1 Potential Applications

Continuous hand-mode detection can be used to make user input easier for enhancing mobile user interaction. At first, our system could potentially improve current UIs for one-handed use [25, 35] by automatically adjusting a device's layout and removing the need for manual input. For example, users can use one-handed mode right after unlocking the phone using face recognition. Also, our system provides one-handed input mode instantly, even if the user switches their hand mode to enter text. Second, with our system, interface designers could devise adaptive buttons [35] in various applications based on the continuous hand mode tracking. For instance, one could develop a virtual selfie mode button, a button that moves according to which hand is being used. Currently, selfies can be difficult to take because the button is typically located near the bottom of the screen, which tends to be difficult to touch when the arm is stretched at wide angles to take good pictures [44]. A virtual selfie mode button could help users take selfies more easily.

6.2 Walking Scenario

The study we presented above only evaluated the performance of the system when the user is not moving (e.g., sitting, standing). However, it is also important to understand how would HandyTrak perform if the user is in motion, as people do use smartphones while walking [12]. To investigate this issue, we conducted a follow-up user study with five of the study participants (p1, p3, p4, p5, p6) who participated in the previous study. The overall tasks and procedure are similar to the first study, except that the participant was asked to keep walking in a lab setting. Participants carried the Raspberry Pi, which was powered by a power bank while walking. We used the first 80% of the data for training and the last 20% of the data for testing. The results showed that HandyTrack was still able to recognize the three hand-holding modes with an average accuracy of 88.36%, which only dropped 0.67% compared to when the user is static. The result shows that HandyTrak still works when users operate their smartphones while walking.

6.3 Changing Environments and Scenarios & In-The-Wild Evaluation

Changing environments and scenarios can potentially impact the performance of our system. The walking scenario suggests that changing the environment would not affect the performance of our system. However, more study in the wild (e.g., various backgrounds, body shapes, hairstyles) needs to be done. When the participants were walking around the room, the background and light conditions constantly changed. The system achieved comparable performance in walking scenarios with an accuracy of 88% compared to 89% in static scenarios. This demonstrated that HandyTrak could work well when the background changes.

We plan to explore different technical solutions to address this problem. 1) We can generate huge synthetic training sets with a great diversity of hairstyles, body shapes and backgrounds, by replacing the backgrounds in the images virtually, as demonstrated in previous work [14]. This would help us to simulate the backgrounds and hairstyles in the real world without collecting more data. 2) Allowing the model to passively collect data and regularly update the model in the wild can help address this challenge.

6.4 Battery and Privacy Issues

Although our system can continuously track a user's hand mode while they operate the phone, it could quickly drain batteries on the phone and cause privacy issues from pictures being taken of users. However, we believe that these problems can be addressed by applying our system at selective times as well as integrating other sensors. For example, our system could be used to activate only for certain time periods depending on the purpose of its use in applications, e.g. at the moment of unlocking or at the touch moment for opening a keyboard layout. Deploying our system at certain time periods would save battery as well as protect users' privacy. Also, our system can be integrated with other detection techniques using touch interaction trace and IMU sensor values [8, 9, 39] to be more intelligent for sensing the hand mode. For instance, most people would not frequently change the holding hand. Therefore, instead of continuously tracking the holding hand, we can use the built-in IMU sensor to detect the possible moment of switching the holding hands and capture pictures only when a possible switching event happens. Furthermore, an on-screen indicator that the front camera is tracking can be one way to reduce privacy concerns for users.

6.5 Deploy HandyTrak in Real-time

Our current system shows good performance in detecting hand mode, but it takes a lot of computing power for both the data preprocessing and the neural network classifier. Since it could be difficult to run on a mobile device for real-time classification, we have experimented with alternative, less computationally expensive, methods for the mobile device. First, we reduced the size of the input images to 16×16 for speed. We leveraged an image segmenter named ShuffleNet V2 [46] [3] and a simple convolutional neural network classifier containing two Convolution2D layers (64 and 32 filters, respectively), a flatten layer, and a softmax layer for determining hand mode. This setup worked well in real-time on our Android device (see accompanying video).

We tested this on the data set from the user study with the same validation method. It showed an average accuracy of 84.85% (SD: 13.4%) under two hand positions and five cameras. Specifically, HandyTrak achieved an average accuracy of 73.85% (SD: 13.3%) and 95.84% (SD: 13.4%) for middle and skewed hand poses, respectively. These accuracies are slightly worse than those of HandyNet, but they can be potentially improved by using other ways to deploy our system on a mobile device. For the data preprocessing, there are many image segmentation techniques for mobile devices [40][4]. In addition, there are ways to deploy large networks, like our VGG 16 network, on mobile devices while running quickly [10, 22, 32, 37].

6.6 Wearing a Mask

Living in the pandemic caused by COVID-19, many people wear masks throughout the day. To understand how the mask would impact our system, we experimented with mask effects on two researchers (Figure 12.a) to classify hand mode in the sitting position. We developed our model using this data in the same fashion, and it achieved an accuracy of 98.8% in a preliminary experiment. The high accuracy indicates that our system would potentially work well even when the user wears a face mask. This is not a surprise to us, as the face mask would not change the silhouettes of the body images.

[3]https://github.com/sercant/android-segmentation-app
[4]https://github.com/tensorflow/examples/tree/master/lite/examples/image_segmentation https://www.tensorflow.org/lite/examples/segmentation/overview

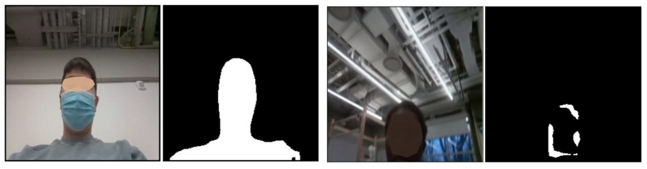


Figure 12: Effect of wearing masks and faulty segmentation outcomes. As shown in (a), even when users wear masks, image frames are still segmented properly. As shown in (b), when only a small part of the user's upper body such as the face appears in the frame, the segmented images frames are sometimes faulty.

6.7 Improving Segmentation

Robust segmentation is essential for our system. However, the current segmentation technique in our system is not perfect yet. For example, it tends to miss some pixels of the person on certain images where only a small part of the face and none of the body is shown (See Figure 12.b) due to grip changes [6, 20]. Sometimes, inaccurate segmentation also occurs when participants change their hand grip between tasks. We used all data (even if the segmentation failed) for training and testing purposes. Removing the images that fail in the segmentation would further improve performance.

To improve the image segmentation, we could consider using the 3D-depth camera. Some commercial phones already have 3D-depth cameras (e.g. Face ID on the latest iPhone). There are image segmentation techniques using both color and depth information [4, 38]. The additional information in a 3D-depth image could also be useful to recognize hand mode. We believe using depth information would further improve the segmentation of the body from the background, which can lead to improved performance. We plan to further explore this direction in the future.

6.8 User-Dependency Model and Possible Training Methods

Requiring the user to provide training data is one potential limitation of HandyTrak. However, we believe that how the user experience will be impacted depends on how much effort the user needs to spend to train the system. To further reduce the training efforts from the user, HandyTrak could potentially passively collect training data and regularly update the model. The ground truth can be acquired by 1) asking the user to input holding hand occasionally, or 2) using prior work which uses other methods (e.g., IMU, touch input) to recognize the holding hand.

6.9 User Independent Model Evaluation

We investigate how well a user-independent model would perform. We develop and evaluate our model using a leave-one-participant-out method, where we use nine participants' data for training and one participant for testing. We repeated this process ten times (i.e., all the combinations from the ten users). The result shows that HandyTrak achieved 56.42% (SD: 4.21%) and 59.12% (SD: 3.91%) respectively in middle and skewed positions, indicating that our system could not perform well for a user-independent study to infer which hand participants hold the smartphone with. This was not surprising at all. Because different participants have different hand poses and body shapes, it is hard to expect a model with such a small training set to generalize well on a new user's data. However, if provided with a huge amount of data on different hand positions and body shapes, our model can potentially be improved. Given the limited resources we have as a research lab for large-scale data collection, we hope to collaborate with industry partners to investigate how much training data is needed for the user-independent model. Using a synthetic dataset can also help address this question, which was discussed in Section 6.3.

6.10 Limitation and future work

Overall, our system performs well but there is still room for improvement, just like any research prototype. First, our system only uses three poses (standing, sitting, and resting against a desk). There are more diverse poses that people use when using a smartphone [13, 26] (e.g., on the floor, prone, spine, and so on) that we do not experiment with. To generalize our system in real-world scenarios, we need to collect the data and train the model using images from more diverse poses since different poses lead to different images. Second, our system differentiates between left, right, and both hands. However, there are other hand modes such as cradling [12] that we did not cover in our study. We will further explore more hand modes in future work with other sensors such as IMU. Third, we only evaluated our system when the phone is held in portrait mode. We believe that it could be possible to create a system that would also work in landscape mode. Lastly, computer vision-based techniques have an inherent problem when working in the dark. Our system may not work at night or in a dark room, but we believe it would be addressed with infra-red depth cameras like the one used in the latest iPhone.

7 CONCLUSION

In this paper, we present HandyTrak, an AI-powered software solution for commodity smartphones, to continuously detect which hand is holding the phone. It uses the front-facing camera to capture images of the user, which are learned by a customized deep learning model to infer the holding hand (left, right, or both). A user study with 10 participants showed that our system can continuously recognize the three hand holding modes with an average accuracy of 89.03% at 2Hz across different settings. This encouraging result suggests that HandyTrak can be deployed on mobile applications to acquire the holding hand information without requiring any additional hardware or explicit input from the user.

REFERENCES

[1] Jeff Avery, Daniel Vogel, Edward Lank, Damien Masson, and Hanae Rateau. 2019. Holding patterns: detecting handedness with a moving smartphone at pickup. In *Proceedings of the 31st Conference on l'Interaction Homme-Machine*. 1–7.

[2] Joanna Bergstrom-Lehtovirta and Antti Oulasvirta. 2014. Modeling the functional area of the thumb on mobile touchscreen surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1991–2000.

[3] Tat-Jen Cham, Arridhana Ciptadi, Wei-Chian Tan, Minh-Tri Pham, and Liang-Tien Chia. 2010. Estimating camera pose from a single urban ground-view omnidirectional image and a 2D building outline map. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 366–373.

[4] Meir Johnathan Dahan, Nir Chen, Ariel Shamir, and Daniel Cohen-Or. 2012. Combining color and depth for enhanced image segmentation and retargeting. *The Visual Computer* 28, 12 (2012), 1181–1193.

[5] Akbar Dehghani, Omid Sarbishei, Tristan Glatard, and Emad Shihab. 2019. A quantitative comparison of overlapping and non-overlapping sliding windows for human activity recognition using inertial sensors. *Sensors* 19, 22 (2019), 5026.

[6] Rachel Eardley, Anne Roudaut, Steve Gill, and Stephen J Thompson. 2018. Investigating How Smartphone Movement is Affected by Body Posture. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–8.

[7] Pradipta Ghosh, Xiaochen Liu, Hang Qiu, Marcos AM Vieira, Gaurav S Sukhatme, and Ramesh Govindan. 2020. On Localizing a Camera from a Single Image. *arXiv preprint arXiv:2003.10664* (2020).

[8] Mayank Goel, Alex Jansen, Travis Mandel, Shwetak N Patel, and Jacob O Wobbrock. 2013. ContextType: using hand posture information to improve mobile touch screen text entry. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 2795–2798.

[9] Mayank Goel, Jacob Wobbrock, and Shwetak Patel. 2012. Gripsense: using built-in sensors to detect hand posture and pressure on commodity mobile phones. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*. 545–554.

[10] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. 2018. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 784–800.

[11] Ken Hinckley, Seongkook Heo, Michel Pahud, Christian Holz, Hrvoje Benko, Abigail Sellen, Richard Banks, Kenton O'Hara, Gavin Smyth, and William Buxton. 2016. Pre-touch sensing for mobile interaction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 2869–2881.

[12] Steven Hoober. 2013. How do users really hold mobile devices. *Tillgänglig http://www. uxmatters. com/mt/archives/2013/02/how-do-users-really-hold-mobile-devices. php (Hämtad 2015-02-11)* (2013).

[13] Michael Xuelin Huang, Jiajia Li, Grace Ngai, and Hong Va Leong. 2017. Screenglint: Practical, in-situ gaze estimation on smartphones. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2546–2557.

[14] Dong-Hyun Hwang, Kohei Aso, Ye Yuan, Kris Kitani, and Hideki Koike. 2020. MonoEye: Multimodal Human Motion Capture System Using A Single Ultra-Wide Fisheye Camera. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 98–111.

[15] Apple Inc. 2021. About the keyboards settings on your iPhone, iPad, and iPod touch. *https://support.apple.com/en-us/HT202178* (2021).

[16] Apple Inc. 2021. Adjust touch settings on iPhone. *https://support.apple.com/guide/iphone/touch-iph77bcdd132/14.0/ios/14.0* (2021).

[17] Apple Inc. 2021. Use Face ID on your iPhone or iPad Pro. *https://support.apple.com/en-us/HT208109 / php (Hämtad 2021-01-22)* (2021).

[18] Samsung Inc. 2021. Using One Handed Mode on my Samsung Phone. *https://www.samsung.com/au/support/mobile-devices/using-one-handed-mode/ php (Hämtad 2021-03-19)* (2021).

[19] Alex Kendall, Matthew Grimes, and Roberto Cipolla. 2015. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*. 2938–2946.

[20] Mohamed Khamis, Anita Baier, Niels Henze, Florian Alt, and Andreas Bulling. 2018. Understanding face and eye visibility in front-facing cameras of smartphones used in the wild. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[21] Kee-Eung Kim, Wook Chang, Sung-Jung Cho, Junghyun Shim, Hyunjeong Lee, Joonah Park, Youngbeom Lee, Sangryoung Kim, et al. 2006. Hand grip pattern recognition for mobile user interfaces. In *AAAI*. 1789–1794.

[22] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. 2015. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530* (2015).

[23] Zakaria Laskar, Iaroslav Melekhov, Surya Kalia, and Juho Kannala. 2017. Camera relocalization by computing pairwise relative poses using convolutional neural network. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 929–938.

[24] Huy Viet Le, Sven Mayer, Patrick Bader, and Niels Henze. 2018. Fingers' Range and Comfortable Area for One-Handed Smartphone Interaction Beyond the Touchscreen. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.

[25] Hosub Lee and Young Sang Choi. 2011. Fit your hand: personalized user interface considering physical attributes of mobile device users. In *Proceedings of the 24th annual ACM symposium adjunct on User interface software and technology*. 59–60.

[26] Florian Lehmann and Michael Kipp. 2018. How to hold your phone when tapping: A comparative study of performance, precision, and errors. In *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces*. 115–127.

[27] Xiaotian Li, Juha Ylioinas, Jakob Verbeek, and Juho Kannala. 2018. Scene coordinate regression with angle-based reprojection loss for camera relocalization. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*. 0–0.

[28] Hyunchul Lim, Gwangseok An, Yoonkyong Cho, Kyogu Lee, and Bongwon Suh. 2016. WhichHand: automatic recognition of a smartphone's position in the hand using a smartwatch. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*. 675–681.

[29] Markus Löchtefeld, Phillip Schardt, Antonio Krüger, and Sebastian Boring. 2015. Detecting users handedness for ergonomic adaptation of mobile user interfaces. In *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*. 245–249.

[30] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3431–3440.

[31] Mona Hosseinkhani Loorak, Wei Zhou, Ha Trinh, Jian Zhao, and Wei Li. 2019. Hand-over-face input sensing for interaction with smartphones through the built-in camera. In *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services*. 1–12.

[32] Jiachen Mao, Zhongda Yang, Wei Wen, Chunpeng Wu, Linghao Song, Kent W Nixon, Xiang Chen, Hai Li, and Yiran Chen. 2017. Mednn: A distributed mobile system with enhanced partition and deployment for large-scale dnns. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 751–756.

[33] Sven Mayer, Gierad Laput, and Chris Harrison. 2020. Enhancing Mobile Voice Assistants with WorldGaze. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–10.

[34] Greg Mori, Xiaofeng Ren, Alexei A Efros, and Jitendra Malik. 2004. Recovering human body configurations: Combining segmentation and recognition. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, Vol. 2. IEEE, II–II.

[35] Kriti Nelavelli and Thomas Ploetz. 2018. Adaptive App Design by Detecting Handedness. *arXiv preprint arXiv:1805.08367* (2018).

[36] Jakob Nielsen. 1994. *Usability engineering*. Morgan Kaufmann.

[37] Wei Niu, Xiaolong Ma, Sheng Lin, Shihao Wang, Xuehai Qian, Xue Lin, Yanzhi Wang, and Bin Ren. 2020. Patdnn: Achieving real-time dnn execution on mobile devices with pattern-based weight pruning. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*. 907–922.

[38] Cristina Palmero, Albert Clapés, Chris Bahnsen, Andreas Møgelmose, Thomas B Moeslund, and Sergio Escalera. 2016. Multi-modal rgb–depth–thermal human body segmentation. *International Journal of Computer Vision* 118, 2 (2016), 217–239.

[39] Chanho Park and Takefumi Ogawa. 2015. A Study on Grasp Recognition Independent of Users' Situations Using Built-in Sensors of Smartphones. In *Adjunct Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 69–70.

[40] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.

[41] Sayan Sarcar, Chaklam Silpasuwanchai, William Delamare, Ayumu Ono, Antti Oulasvirta, and Xiangshi Ren. 2019. Exploring performance of thumb input for pointing and dragging tasks on mobile device. In *Proceedings of Asian CHI Symposium 2019: Emerging HCI Research Collection*. 38–45.

[42] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[43] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[44] Leigh Stark. 2018. How to take photos without touching your phone's screen. *https://www.pickr.com.au/how-to/2018/how-to-take-photos-without-touching-your-phones-screen/ php (Hämtad 2018-01-30)* (2018).

[45] Ke Sun, Chun Yu, Weinan Shi, Lan Liu, and Yuanchun Shi. 2018. Lip-interact: Improving mobile device interaction with silent speech commands. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. 581–593.

[46] Sercan Türkmen and Janne Heikkilä. 2019. An Efficient Solution for Semantic Segmentation: ShuffleNet V2 with Atrous Separable Convolutions. In *Image Analysis*, Michael Felsberg, Per-Erik Forssén, Ida-Maria Sintorn, and Jonas Unger (Eds.), Vol. 11482. Springer International Publishing, Cham, 41–53. https://doi.org/10.1007/978-3-030-20205-7_4

[47] Radu-Daniel Vatavu. 2011. The effect of sampling rate on the performance of template-based gesture recognizers. In *Proceedings of the 13th international conference on multimodal interfaces*. 271–278.

[48] Chun Yu, Xiaoying Wei, Shubh Vachher, Yue Qin, Chen Liang, Yueting Weng, Yizheng Gu, and Yuanchun Shi. 2019. Handsee: enabling full hand interaction on smartphone with front camera-based stereo vision. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–13.