InfiniTouch: Finger-Aware Interaction on Fully Touch Sensitive Smartphones

Huy Viet Le[1], **Sven Mayer**[1], **Niels Henze**[1,2]
[1]University of Stuttgart, Germany, [2]University of Regensburg, Germany
[1]{huy.le, sven.mayer}@vis.uni-stuttgart.de, [2]niels.henze@ur.de

ABSTRACT
Smartphones are the most successful mobile devices and offer intuitive interaction through touchscreens. Current devices treat all fingers equally and only sense touch contacts on the front of the device. In this paper, we present *InfiniTouch*, the first system that enables touch input on the whole device surface and identifies the fingers touching the device without external sensors while keeping the form factor of a standard smartphone. We first developed a prototype with capacitive sensors on the front, the back and on three sides. We then conducted a study to train a convolutional neural network that identifies fingers with an accuracy of 95.78 % while estimating their position with a mean absolute error of 0.74 *cm*. We demonstrate the usefulness of multiple use cases made possible with *InfiniTouch*, including finger-aware gestures and finger flexion state as an action modifier.

ACM Classification Keywords
H.5.2 User Interfaces: Input devices and strategies

Author Keywords
Touchscreen; machine learning; finger-aware interaction.

INTRODUCTION
Over two billion people use smartphones for applications that were previously exclusive to computers, including email writing, browsing the internet, and editing pictures [64]. In contrast to computers, smartphones can be used while on the move and in a wide range of situations in which only one hand is available for interaction. From a technical point of view, touchscreens translate the contact areas of touches into two-dimensional positions for an intuitive interaction through direct touch. However, this limits the input vocabulary as smartphones are commonly used one-handedly [5, 40] with only the thumb touching the display. This slows down interaction compared to input devices for computers and generally leads to reachability issues.

Input devices such as mouse and keyboard are designed to be used with multiple fingers and offer more input dimensions than touchscreens by enabling multiplex input. For example, a mouse offers at least two buttons that enable index and middle fingers to activate different functions at the same cursor position while a hardware keyboard offers modifier keys to perform shortcuts. Thus, smartphone manufacturers incorporated input controls beyond the touchscreen that can be controlled by fingers which previously only held the device (*e.g.*, fingerprint scanners). Similarly, previous work thoroughly investigated Back-of-Device (BoD) interaction and presented smartphone prototypes with touch input capability on the back [17, 42], the edges [34], and on the whole device surface [44, 53]. These devices enable multiple fingers to perform explicit (*e.g.*, gestures [42, 63]) or implicit input (*e.g.*, grip recognition [44]).

Figure 1. Our full-touch smartphone prototype based on two LG Nexus 5 and a Genuino MKR1000 for touch sensing on the edges.

Common approaches for BoD interaction simply add a second touchscreen [15, 17, 76], so that contact areas are still translated into two-dimensional positions while inputs of all fingers are treated equally. Thus, it is not possible to use different fingers for different functions. Moreover, the full finger and hand surface can get in contact with the back side (*c.f.*, Figure 3a) so that translating individual contact areas to two-dimensional positions is not feasible anymore. Previous work limited the size of the touch panel [6, 17, 42], or used patterns of the grip contact areas to activate different actions [11, 12, 13]. In addition, they used external hardware (*e.g.*, wearable sensors or cameras) which reduces the mobility but enables identification of the touching finger. However, there is no previous work that interprets all contact areas on the device surface to enable different fingers to activate different functions. The concept of treating different fingers individually for touch interaction has been called *finger-aware interaction*. Each finger could be responsible for a specific function similar to how computer

UIST '18, October 14–17, 2018, Berlin, Germany

ISBN 978-1-4503-5948-1/18/10…$15.00
DOI: https://doi.org/10.1145/3242587.3242605

mouses are used. This enables a wide range of applications including shortcuts, mode switches, finger-specific functions, reachability improvements, and one-handed pinch gestures.

In this paper, we develop *InfiniTouch*, a system that enables finger-aware interaction on a smartphone's whole device surface. In contrast to previous work, our prototype has the form factor of a standard smartphone and does not require external hardware. We use the contact areas on the whole device surface to train a convolutional neural network (CNN) for finger identification. The model achieved an accuracy of 95.78 % for identifying fingers on the device surface while estimating their 3D position with a mean absolute error (MAE) of 0.74 *cm*. In contrast to grip recognition [44], our model enables fingers to perform implicit as well as explicit input. We implemented multiple use cases to showcase the performance of our model and the usefulness of finger-aware interaction on the whole device surface of smartphones during one-handed interaction.

BACKGROUND AND RELATED WORK

We develop finger-aware interaction on smartphones with touch sensing on the whole device surface. Thus, we review previous approaches that enable finger-aware interaction on touch surfaces. Afterwards, we review previous approaches that enable touch interaction beyond the touchscreen.

Finger-Aware Touch Interaction

Three common approaches were presented in previous work to enable finger-aware interaction on touch surfaces: (1) using wearable sensors attached to the user, (2) using cameras to capture the hand, and (3) interpreting the shape geometry of touches. In general, attaching wearable sensors to the user, including vibration sensors [49], infrared sensors [29, 30], gloves [48], and electromyography [7], yields the highest accuracies. However, these approaches reduce mobility since additional sensors are required. Instead of instrumenting the hand, RGB cameras (*e.g.*, webcams) and computer vision techniques were used to identify fingers touching a surface [71, 83]. Moreover, depth and infrared cameras such as the Microsoft Kinect and the Leap Motion enable touch sensing [56, 73] and are commonly used to prototype finger-aware interaction [14]. However, the cameras need to be attached or close to the smartphone which affects form factor and mobility. Tabletops (*e.g.*, Microsoft PixelSense) use infrared cameras integrated into the display to provide high-resolution touch images that enable a wide range of applications [3, 9, 19, 22, 50]. Unfortunately, such data is not available on commodity smartphones.

Therefore, another branch of research used the raw data of touchscreens to infer the source of touch. These are low-resolution fingerprints that can be retrieved from mutual capacitive touchscreens and are referred to as *capacitive images* [28, 36, 43, 51]. Capacitive images are mainly used by the touch controller to determine a precise touch position and are typically not exposed to the application layer. Previous work used this data for a wide range of applications, including biometric authentication [28, 36], detecting palm touches [43], and estimating the finger orientation [51, 55, 78]. Gil *et al.* [23] used capacitive images to differentiate between touches from thumb, index, and middle finger on smartwatches. However, when not performed in exaggerated poses, the accuracy is around 70 % which is not reliable enough for interaction.

While the use of capacitive images does not require external hardware, their accuracy is not sufficient for interaction. In contrast to cameras or sensors attached to the user's hand, touchscreens can only sense individual fingertips without any information about the remaining hand. This makes it challenging to identify which fingers touched the display. To the best of our knowledge, there is no previous work that uses a standalone smartphone without wearable sensors or cameras to enable finger-aware interaction with a suitable accuracy.

Touch Sensing beyond the Front Touchscreen

The simplest and most common approach for BoD interaction is to attach two smartphones back-to-back [15, 17, 62, 75, 76]. However, this approach increases the device thickness which negatively affects the hand grip and interaction [42, 65]. This is detrimental for studies that observe the hand behavior during BoD interaction, and could lead to muscle strain. To avoid altering the device's form factor, researchers built custom devices that resemble smartphones [6, 11, 66]. However, these approaches mostly lack the support of an established operating system so that integrating novel interactions into common applications becomes tough. As a middle ground, researchers use small additional sensors that barely change the device's form factor. These include 3D-printed back cover replacements to attach a resistive touch panel [42], and custom flexible PCBs with 24 [53, 54] and 64 [11] square electrodes. However, neither the panel size nor the resolution is sufficient to enable precise finger-aware interactions such as gestures and absolute input on par with state-of-the-art touchscreens.

Beyond capacitive sensing, researchers proposed the use of inaudible sound signals [57, 61, 72], high-frequency AC signals [82], electric field tomography [81], conductive ink sensors [26], the smartphone's camera [77, 79], and other built-in sensors such as IMUs and microphones [27, 60, 80]. While these approaches do not increase device thickness substantially, their raw data lack details to enable finger-aware interaction.

While using flexible PCBs as presented in previous work is a promising approach, the resolution is not sufficient. Further, previous work used proprietary technologies so that other researchers cannot reproduce the prototype to investigate interactions on such devices. There is no previous work that presents a reproducible (*i.e.*, uses commodity hardware) full-touch smartphone prototype.

Summary

Related work predominantly used external hardware (*e.g.*, wearable sensors and cameras) to enable finger-aware interaction on touch surfaces. Since these approaches limit mobility, researchers investigated the use of the shape geometry of touches which can be retrieved from capacitive touchscreens. However, the achieved accuracy is insufficient for interaction as touchscreens can only register a limited part (*i.e.*, individual fingers) of the hand. Without information about the remaining hand and its degrees of freedom, it is challenging to infer which finger touched the device. Thus, we propose to use capacitive images representing touches on the whole device

surface to train a finger identification model. We develop a full-touch smartphone that provides these capacitive images and train a state-of-the-art machine learning model to identify touches. We show that fingers can be identified with an accuracy of 95.78 % while the finger positions can be estimated with an MAE of 0.74 *cm* during one-handed interaction.

FULL-TOUCH SMARTPHONE PROTOTYPE

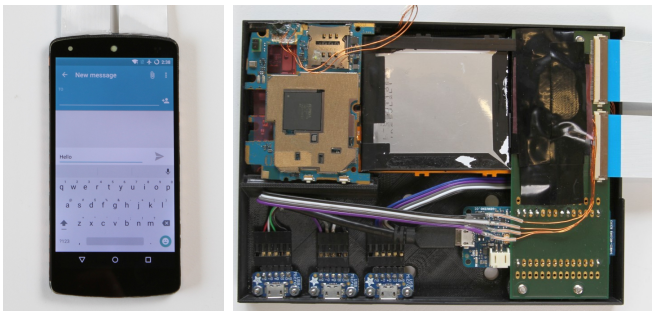
We developed a full-touch smartphone prototype that provides capacitive images for a finger identification model. We adapted an approach presented in previous work [44] and used two LG Nexus 5 as basis which provides capacitive images with a resolution of 27×15 *px* on the front and back side. As using the full hardware of both smartphones (*i.e.*, stacking the devices) lead to a noticeable increase in thickness, we separated the prototype into two modules to gain flexibility in form factor: a *Handheld Device*, and a *Hardware Container*. Each module is comprised of a self-designed printed circuit board (PCB) that act as extension adapters to connect the *Handheld Device* (PCB_{HD}) and *Hardware Container* (PCB_{HC}) with each other via flexible flat cables (FFC). Instead of manufacturing proprietary sensors, we based our prototype on two commodity smartphones and release the schemes of our self-designed PCB so that the community can re-implement our prototype. This enables further exploration of interaction techniques based on finger-aware interaction on the whole device surface. The PCB schemes, 3D models, component descriptions, and source code to reproduce our prototype are available on our project page[1] under the MIT license.

Handheld Device

The *Handheld Device* (see Figures 1 and 2a) consists of a 3D printed frame, two Nexus 5 touchscreens, 37 copper plates as capacitive touch sensors on the edges, and a PCB_{HD}. The 3D printed frame holds both touchscreens and encloses PCB_{HD}. The capacitive touch sensors are fixated on the left, right and bottom side of the frame. Each touch sensor is a $6 \times y \times 0.5$ *mm* (with $y = 6$ *mm* for left and right, and $y = 12$ *mm* for the bottom side) copper plate which is glued into engravings of the frame with a gap of 1.0 *mm* in between and sanded down for a smooth feeling. The copper plates are connected to the PCB_{HD} which in turn comprises three MPR121 capacitive touch controllers operated by a Genuino MKR 1000 microcontroller in the *Hardware Container*. Similarly, both touchscreens are connected via a board-to-board connector on the PCB_{HD} and are operated by the remaining components of the Nexus 5 located in the *Hardware Container*. The two FFCs are routed through the top side of the *Handheld Device* as there is a low likelihood that it disturbs the user when holding the phone in a usual grip [45, 46]. The dimensions of the *Handheld Device* are $137.6 \times 68.7 \times 8.9$ *mm* (115 *g*). In comparison, the dimensions of an off-the-shelf Nexus 5 are $137.8 \times 69.1 \times 8.6$ *mm* (130 *g*).

Hardware Container

The *Hardware Container* (see Figure 2b) is a 3D printed box that contains two Nexus 5 circuit boards and batteries, a Genuino MKR 1000 microcontroller, three micro USB breakout boards, and two tactile buttons. The circuit board of the Nexus 5 is connected to a compatible board-to-board connector on PCB_{HC} which in turn is connected to the touchscreens. To access the power buttons and USB ports of the two Nexus 5, we replaced them with tactile buttons and USB micro breakouts integrated in the *Hardware Container*. Moreover, we extended the Genuino's USB port to a USB micro breakout board. The Genuino MKR 1000 is connected to the PCB_{HC} to operate the side sensors connected to the PCB_{HD}, and can be powered by battery via the JST connector or through USB.



(a) Handheld Device (b) Hardware Container

Figure 2. Full-touch smartphone prototype: (a) the *Handheld Device*, and (b) *Hardware Container* containing the processing units. Both components are connected via our self-designed PCB and flexible flat cables.

Capacitive Images and Interconnection

We accessed the 27×15 *px* capacitive images of the front and back touchscreen by modifying the Android kernel. Each pixel corresponds to a 4.1×4.1 *mm* square on the 4.95″ touchscreen. The pixel values represent the differences in electrical capacitance (in *pF*) between the baseline measurement and the current measurement. We used I2C calls to access the register for test reporting as described in the RMI4 specification (511-000405-01 Rev.D) and the driver's source code[2]. We pulled the capacitive images from the debugging interface with 20 *fps* and stored them in the proc filesystem (procfs) to make them accessible in the application layer. As the edge sensors are square electrodes, we simply read their values with the MPR121 library to retrieve a capacitive image.

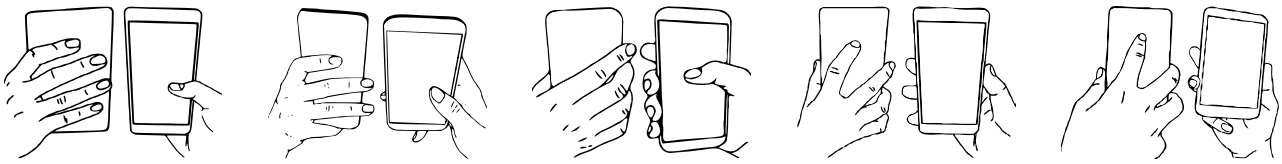
To generate a merged capacitive image of the sensor values on all sides, the Nexus 5 responsible for the front opens a WiFi hotspot to receive the values from the Nexus 5 on the back and the Genuino MKR 1000. The transfer latency measured by an average round trip time over 1000 samples is 7.2 *ms* (*SD* = 2.6 *ms*). As the capacitive images can be pulled from the debugging interface with 20 *fps* at most, the transfer latency can be neglected. Data from side sensors can be retrieved at 130 *fps*. We developed an Android library that retrieves the capacitive images, establishes a connection between front, back and side, and provides a callback function in which developers can retrieve the merged capacitive images.

GROUND TRUTH DATA COLLECTION

Using our prototype, we conducted a study to collect a dataset comprising the capacitive images and respective 3D motion

[1] https://github.com/interactionlab/InfiniTouch

[2] github.com/CyanogenMod/android_kernel_lge_hammerhead/blob/cm-12.1/drivers/input/touchscreen/touch_synaptics_ds5.c



(a) Grip 1 (b) Grip 2 (c) Grip 3 (d) Grip 4 (e) Grip 5

Figure 3. Five HAND GRIPs used in the study and adopted from previous work [47].

data of each joint of the hand. The former will be the input for our model and the fingertips of the latter the output. Participants performed finger movements starting from five hand grips as shown in Figure 3 to cover possible finger positions.

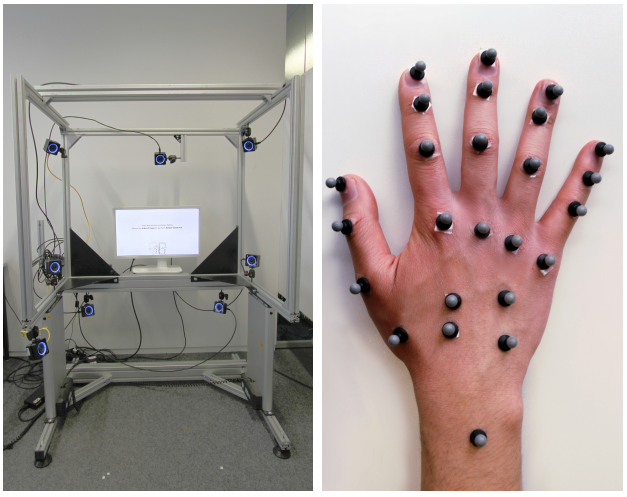
Apparatus

To record finger motions with sub-millimeter accuracy, we used an *OptiTrack* motion capture system with eight cameras (*OptiTrack* capturing at 240 *fps*). The cameras were firmly mounted to an aluminum profile grid as shown in Figure 4a. To enable these infrared cameras to record the finger movements, we attached 25 reflective markers (6.4 *mm* spherical markers with skin adhesive M3 base) on all joints of the hand similar to previous work [20, 45] and as shown in Figure 4b. Additionally, we attached four markers on the top part of the full-touch smartphone which enables us to track the device in six degrees of freedom. We installed a display in front of the participant to show instructions (see Figure 4a).

Design

The study has three independent variables, HAND GRIP, FINGER and TASK. For HAND GRIP we used known hand grips that were shown in previous work [47] and in Figure 3. For FINGER, we used all five fingers of the right hand. As tasks, we used *free movements*, in which participants freely moved a specified finger; *swipe gestures*, in which participants performed swipe gestures into left, right, bottom and up directions; and *free placements with thumb* in which participants placed the specified finger followed by a thumb movement to simulate using fingers on the rear as modifiers.

The three independent variables result in a $5 \times 5 \times 3$ within-subject design. We counterbalanced the GRIP using a balanced Latin square and used a random order for FINGER and TASK. The duration of each task was 30 seconds which results in a total duration of $30\,sec \times 5 \times 5 \times 3 = 37.5$ minutes. During these tasks, participants were surrounded by eight motion capture cameras and were seated on a chair without armrests as shown in Figure 4a. Including the briefing, optional breaks, and attaching markers, the study took around 60 minutes.



(a) (b)

Figure 4. Study setup: (a) motion capture system with 8 cameras mounted on a aluminum profiles and (b) reflective markers (6.4 *mm* spheres) attached to the hand of a participant.

Procedure

After we obtained informed consent, we collected demographic data using a questionnaire and measured the participants' hand size and finger lengths. We then proceeded to attach 25 skin adhesive markers on their right hand to enable motion tracking. Instruction slides were shown on the display which explains the procedure of the study as well as the finger movements and hand grips that participants should perform. We further showed them a demo of the required movements and asked them to perform it on trial to ensure that everything was fully understood.

After handing the full-touch smartphone to the participants, they first imitated a grip shown on the instruction display and were then instructed to perform the shown task. While the displayed finger specifies the main finger to move, we allowed the participants to also move other fingers if this was necessary to move the main finger. This is necessary to record hand grip states that are as realistic as possible (*e.g.*, the ring finger can only be moved individually to a lesser extent [31]). The described process was repeated for all HAND GRIPS, FINGERS, and TASKS. The experimenter monitored the markers throughout the study to ensure that the finger was moved at an adequate speed and that all markers are visible in the motion capturing.

Participants

We recruited 20 participants (7 female) between the ages of 20 and 29 ($M = 24.1$, $SD = 2.5$). All participants were right-handed. The average hand size was measured from the wrist crease to the middle fingertip and ranged from 15.6 *cm* to 25.0 *cm* ($M = 19.3\,cm$, $SD = 2.0\,cm$). Our collected data comprise samples from the 5th and 95th percentile of the anthropometric data reported in prior work [58]. Thus, the sample can be considered as representative. Participants were reimbursed with 10 EUR for their participation.

FINGER IDENTIFICATION MODEL

We train a model to estimate the fingertip locations using the capacitive images as input. The model output contains an estimated 3D location for each finger which can also be used to identify the source of the contact areas (referred to as *blobs*).

Data Set & Preprocessing

We synchronized the motion data with the capacitive images of the front, back, and edges of the full-touch smartphone. We used the capacitive images as input and the 3D motion data of the fingertips as ground truth for our machine learning model. We performed the following four data preprocessing steps:

1. *Labeling and cleaning motion data*: We labeled all markers in the captured 3D motion data using semi-automatic labeling provided by OptiTrack's *Motive:Body* software. We did not use any reconstruction and smoothing approaches to avoid generating artificial marker positions.

2. *Transforming global to local coordinate system*: We transformed each hand marker from the global coordinate system into the phone's coordinate system and projected them onto the device surfaces. We validated the transformation by sampling five random frames per participant which we manually checked for correctness.

3. *Removing incomplete and erroneous data*: To ensure a complete and valid data set for model training, we keep only frames in which the rigid body and finger tips are fully available. Further, we applied a heuristic to detect erroneous rigid body tracking by assuming that the phone was not held in uncommon poses (*e.g.*, up-side-down, flipped). This heuristic removed 0.21 % of all frames.

4. *Synchronizing motion data and capacitive images*: We merged the capacitive images with the transformed motion data using timestamps as the merge criteria. As touchscreen latency is unavoidable and higher than the latency of a motion capture systems [10], the finger's ground truth position is ahead of the touch, especially during fast movements. To counteract the latency, we used a sliding window of 240 frames for each capacitive image to find a motion capture frame in which the currently moving finger is within the generated blob (preferably in the center). We validated the merging process by checking whether the motion data corresponds to the blobs in the capacitive images. This was done by determining the blob's contour and checking whether the 2D position of the fingertip lies within the contour.

In total, our dataset consists of 9,435,903 valid samples stored in a 67.3 *GB* HDF5 file[3] to enable training on a large dataset.

Estimating the Fingertip Positions using CNNs

To develop the model, we used a participant-wise split of 70 %:20 %:10 % for the training, test, and validation set. *I.e.*, the model is trained on data from 14 participants, tested on 4 participants, and validated on the remaining 2 participants. We implemented CNNs using *Keras* 2.1.3 based on the *TensorFlow* backend. We performed a grid search as proposed by Hsu *et al.* [37] to determine the most suitable network architecture and hyperparameters. If we do not report a hyperparameter in the following, we applied the standard value (*e.g.*, optimizer settings) as reported in Keras' documentation.

[3] support.hdfgroup.org/HDF5/whatishdf5.html

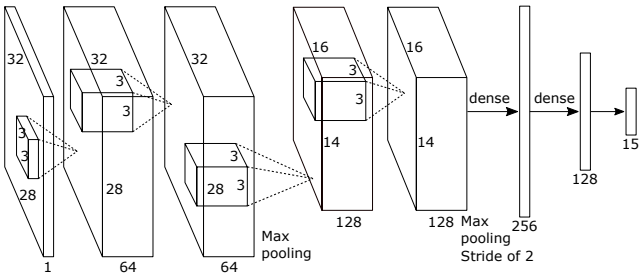


Figure 5. An illustration of the architecture of our CNN for finger position estimation. The network input is 896-dimensional, and the number of neurons in the network's remaining layers is given by 57,334–57,334–28,672–28,672–256–128–15.

Our final CNN architecture is shown in Figure 5. The input consists of capacitive images with 28×32 pixels normalized to a range between 0 and 1. The output consists of 15 values ((x,y,z) for five fingers) that represent the estimated 3D finger positions relative to the upper left corner of the display in *mm*. Thereby, a finger farther away from the device (*e.g.* lifted finger) has a higher distance in the z-axis as captured in the data collection study. We trained the CNN using an RMSprop optimizer [67] (similar to the AdaGrad [18] optimizer but with a less radical learning rate decay) with a batch size of 500. We experimented with different learning rates and found that an initial learning rate of .0001 leads to the best performance. We used batch normalization [38] and a 0.5 dropout after each pooling and dense layer to prevent overfitting. While we experimented with L2 Regularization, it did not improve the overall performance in our experiments. We initialized the network weights using the Xavier initialization scheme [25].

After experimenting with traditional loss functions for regression such as root-mean-squared error (RMSE), we developed a custom loss function to train our CNN. As fingers above or below the device cannot be physically tracked by the touchscreen (*e.g.*, thumb resting above the display), errors induced by movements perpendicular to the touchscreen would affect the RMSE loss function as substantial as an error in horizontal (x) or vertical (y) direction. However, when omitting the z axis, the CNN would lose a feature to differentiate whether fingers are touching the device. Since a less accurate estimation of the z-axis can be easily compensated by checking the blob availability at the time using the model, we lowered the influence of the z-axis error by using an RMSE for the x and y axis, and a root mean squared logarithmic error (RMSLE) [39] for the z-axis as follows:

$$loss = \sqrt{\frac{\sum_i^n (p_{xy_i} - \hat{p}_{xy_i})^2}{n}} + \sqrt{\frac{\sum_i^n log_e((p_{z_i} - \hat{p}_{z_i}) + 1)^2}{n}} \quad (1)$$

with $n = 5$ representing the five finger tips, p for the ground truth point, and $\hat{p}$ for the estimated point.

Identifying Touches from Individual Fingers

To identify the finger touching the device (*i.e.*, the responsible finger for a specific contact area), we used a nearest neighbor approach to map the estimated positions to the blobs in the capacitive images. This approach has two benefits over using the estimated positions directly from the CNN. Firstly, the jitter caused by noise and the nature of machine learning can be prevented since the contact areas on the capacitive images are more stable. As recent touch controllers have shown, a blob can be converted to a precise touch position without any jitter. Secondly, the processor workload can be reduced since model inference is only necessary when fingers are initially touching (*i.e.*, down event) and releasing (*i.e.*, up event) the device. In other cases (*i.e.*, finger moving), fingers can be tracked using the blob position on the capacitive images.

On a technical basis, we performed a contour detection on a $5\times$ up-scaled capacitive image to determine the blobs. We then used the contour points stored in a k-d tree [21] to find the closest blob for an estimated finger position in $O(\log n)$ time. We used OpenCV for the contour detection and the Lanczos4 algorithm [69] to scale up the capacitive image. We used the k-d tree implementation from scipy for the validation and a reimplementation thereof in our Android demo applications.

Validation

While we used the training and test set to experiment with hyperparameters, we used the validation set to evaluate our best model. Our CNN achieved an MAE of 0.74 *cm*. Table 1 shows the errors for each finger and axis. The MAE for the axes are 0.85 *cm*, 0.85 *cm*, and 0.53 *cm* for the x, y, and z axis respectively while the RMSEs are 1.41 *cm*, 1.39 *cm*, and 0.87 *cm*. The average Euclidean distance for all fingers when considering the error in 2D space (on screen) is 1.33 *cm* whereas the average error in 3D space is 1.52 *cm*. Since the RMSE involves a larger penalty for larger errors (*e.g.*, outliers), an RMSE > MAE indicates that errors can occur especially for uncommon finger placements. As expected, the z axis has the lowest error since the usable movement range perpendicular to the displays is the smallest of all axes.

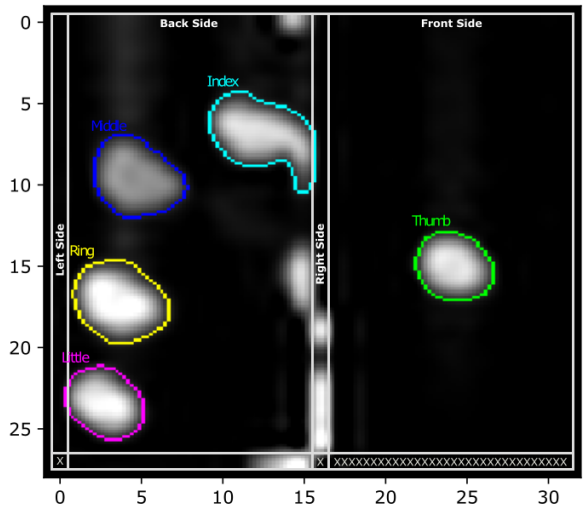


Figure 6. This image shows exemplary capacitive data retrieved from our prototype when held with Grip 1 as shown in Figure 3a. The colored contours represents the results of our finger identification model after mapping the estimations to the blob. The X's represents placeholder values that are required to build a 32×28 input matrix for the model.

	Thumb	Index	Middle	Ring	Little	Average
MAE (x)	1.04	1.03	0.98	0.63	0.56	0.85
MAE (y)	0.73	0.52	0.96	0.99	1.06	0.85
MAE (z)	0.50	0.28	0.46	0.50	0.89	0.53
RMSE (x)	1.80	1.75	1.63	1.06	0.79	1.41
RMSE (y)	0.98	0.87	1.43	1.74	1.93	1.39
RMSE (z)	0.73	0.86	0.79	0.72	1.26	0.87
Eucl. dist. (x, y)	1.40	1.25	1.45	1.25	1.30	1.33
Eucl. dist. (x, y, z)	1.55	1.32	1.57	1.42	1.76	1.52

Table 1. The mean absolute error (MAE), root-mean-squared error (RMSE) and Euclidean distances for each axis in *cm*.

These errors can be compensated for with the finger identification approach as described above. The accuracy of our model with this approach can be evaluated as a multi-label classification problem; multi-label since multiple fingers could be matched to one blob due to the low resolution of the capacitive image. We used both the ground truth fingertip positions and the estimated fingertip positions and matched them with their closest blobs. Based on the matchings, we used the Hamming score [68] which describes the accuracy of a multi-label classification on a scale between 0 (worst) to 1 (best). Our model achieved an average Hamming score of 0.9578.

MOBILE IMPLEMENTATION & SAMPLE APPLICATIONS

We combine the full-touch smartphone, CNN, and nearest neighbor approach to implement *InfiniTouch*. We present our implementation and a set of sample applications.

Mobile Implementation

We used *TensorFlow Mobile*[4] for Android on the processing unit responsible for the front display to run the CNN that estimates the fingertip positions. Capacitive images from the back side and the edges are sent to the front device that merges the data into an input matrix. The input consists of a 32×28 8-bit image representing the front, back, and edges as shown in Figure 6. A model inference for one capacitive image takes 24.7 *ms* on average (min = 17 *ms*, max = 29 *ms*, SD = 2.8 *ms*) over 1000 runs on our prototype. As this is faster than the sampling rate for the touchscreens' capacitive images, the inference can be performed on each sample in the background. With processor manufacturers recently optimizing their processors for machine learning (*e.g.*, Snapdragon 845), model inference can be sped up significantly.[5] The model can be further optimized for mobile devices with techniques such as quantization [33] and pruning [2] for a small loss of accuracy.

For the finger identification, the contour detection, including a scale up, takes M = 2.85 *ms* (SD = 0.77 *ms*, min = 1 *ms*, max = 4 *ms*) while finding the closest blob takes M = 0.48 *ms* (SD = 0.12 *ms*, min = 0.19 *ms*, max = 0.96 *ms*) over 1000 runs on our prototype. Tracking the blobs take M = 0.08 *ms* (SD = 0.04 *ms*, min = 0.001 *ms*, max = 0.82 *ms*). During these benchmarks,

[4] www.tensorflow.org/mobile/

[5] www.qualcomm.com/snapdragon/artificial-intelligence

the device was held one-handedly with all five fingers touching the device (*c.f.* Figure 3a).

Using Finger Identification in the Application Layer

We extended our Android library described above to provide the finger position estimations from the model and the respective position of the blob (*i.e.*, position of the upper-left contour point, and size) for each finger in a callback function. This enables developers to access the finger positions similar to Android's `OnTouchListener` interface. Besides the position (in an on-device coordinate system with the upper left corner of the front display being $(0,0,0)$), we also provide the event (*i.e.*, down, up, and move). With this, the blob's position and estimation can directly be fed into Android's `MotionEvent` which enables to use Android's own `GestureDetector`, or third-party gesture recognizers such as $P [70], $1 [74], $N [1], and the gesture recognition toolkit [24].

Sample Use Cases

Based on the mobile implementation of our model, we implemented two use cases for finger-aware interaction on the full-touch smartphone. We describe our implementation in the following and showcase them in the accompanying video.

Finger-Specific Touch Gestures

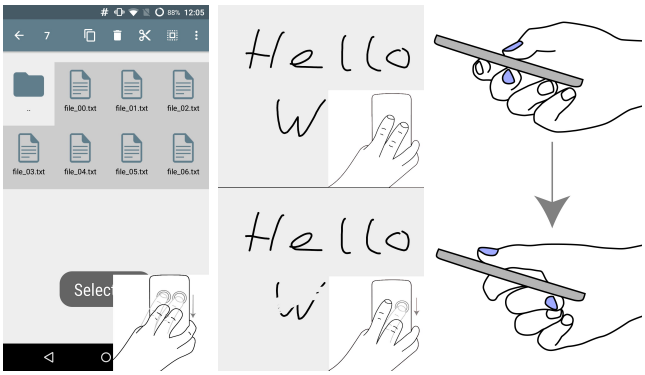
Implementations of BoD interaction in previous work [42, 59, 63] treated inputs of all fingers equally. Thus, performing a gesture with the index finger would result in the same function as a gesture performed with the middle finger. With *InfiniTouch*, the same gesture can activate different functions depending on which finger performed the input. This extends the input space similar to a computer mouse where the index finger is used for main actions, while the middle finger is used for the right mouse button to activate secondary actions.

In our sample use case, we mapped a swipe down performed by the index finger to copying selected items into the clipboard (inspired by the *come to me* gesture) while a swipe down by the middle finger pastes from the clipboard. A swipe down performed by both index and middle finger simultaneously selects all items as shown in Figure 7a. While we demonstrated this concept within a file manager, it can also be used in text editing applications, galleries, and further applications that support the clipboard.

BoD Finger Flexion State as Action Modifier

While hardware keyboards provide modifier keys to modify the action of another key, touchscreens implement this concept only via dwell times or applied force which requires additional execution time. We propose to use the position of the fingertips (*i.e.*, their flexion state) on the back to modify the actions performed on the front screen. For example, bending a specific finger can be done comfortably [45] and could be used similarly to a pressed `Ctrl` key on a hardware keyboard.

We implemented a simple paint application that maps drawing and erasing to the flexion state of the middle finger. When the middle finger is flexed, the pen is activated which enables the user to draw. When bending the middle finger (*c.f.* Figure 7b), the eraser will be activated to remove parts of the drawing. While we demonstrated this concept within a paint application,



(a) (b) (c)

Figure 7. Screenshots of our sample applications implemented on the *InfiniTouch*. Figure (a) showcases how a down-swipe with both index and middle finger selects all files in a file manager, Figure (b) demonstrates how the position of the middle finger can be used to switch between a pen and an eraser, and Figure (c) demonstrates an exemplary one-handed pinch gesture.

it can be applied to a wide range of applications that benefit from action modifiers and with all four fingers. Amongst others, this includes opening context menus similar to the right mouse button, text selection and highlighting, mode switching (*e.g.*, slower and faster scrolling), 3D navigation, and providing shortcuts.

Further Use Cases

We present further use cases for *InfiniTouch*.

One-Handed Pinch and Rotation Gestures

Users need to hold smartphones two-handed or place it on a surface to perform a pinch or a rotation gesture. We propose to use a pre-defined finger on the back of the device as the second finger to perform a pinch/rotation gesture with the thumb on the front screen. This enables users to zoom or rotate objects in a one-handed grip as shown in Figure 7c.

Enabling Transient Actions

Avery *et al.* [4] proposed transient gestures to enable users to temporarily change the view of an application which can be rapidly undone. As a zoom in always requires a zoom out to return to the initial state, they used an additional finger on a tablet to save the initial state. When this additional finger is released, it restores the initial state so that users can alter the view in between. Using our concept of finger positions as a modifier, we could replace the additional finger with a finger on the rear that is able to bend and flex.

Improving Reachability

Bergstrom-Lehtovirta and Oulasvirta [8] showed that the thumb's range could be modeled with the position of the index finger's tip as input. With *InfiniTouch*, we can determine the position of the index finger and can thus adapt the user interface to optimize reachability during one-handed interaction. Moreover, we can assign the functionality to move the screen content to a specific finger. This enables the finger to move the screen content to a more reachable position to improve one-handed interaction as proposed in previous work [42].

DISCUSSION AND LIMITATIONS

We developed *InfiniTouch*, a system that enables finger-aware interaction on full-touch smartphones. We developed a full-touch smartphone prototype and trained a CNN to identify fingers touching the device surface. We implemented and showcased a number of applications.

Model Accuracy

We trained a CNN that estimates the fingertip positions with an MAE of $0.74\,cm$ over all three axes. As a comparison, the average diameter of a human index finger is $1.6\,cm$ - $2.0\,cm$ [16] while Holz *et al.* [35] found that traditional touch interaction has a systematic offset of $0.4\,cm$. Even without using the positions of the blobs, this already enables users to perform precise interactions, such as gestures or finger placements as modifiers. Moreover, using the estimated positions enables differentiation between fingers even if their contact areas are united due to a low-resolution image. A limitation of our model is that estimations of more distant fingers (*e.g.*, a finger moving without touching the device) become less accurate since they cannot be sensed physically.

Based on the estimations, we used a nearest neighbor approach to identify the responsible finger for each blob with an accuracy of 95.78 %. As we perform this process only when the number of blobs in the capacitive image changes, we reduce the processor workload and potential jitter due to noise and the nature of machine learning. Moreover, since we track the blobs while keeping their label (if the number of blobs did not change), labeled blobs stay correctly labeled even if the model yields an inaccurate estimation in a rare hand posture. This means that we only identify fingers when they initially touch or release the device (*e.g.*, new hand grip) with an accuracy of 95.78 % while classification errors cannot occur afterwards. We provide both blobs as well as estimated positions in our Android library, and successfully implemented our sample use cases with both approaches. Further, the estimated location could be used as a fallback in case the blob detection is not capable of telling two blobs apart due to the low resolution.

Improving Accuracy and its Sufficiency for Use Cases

Our model estimates the 3D finger positions with an MAE of $0.74\,cm$ and classifies blobs with an accuracy of 95.78 %. While this is sufficient for a reliable recognition of gestures and the use of absolute positions, future work could further improve the accuracy as follows. Although our 32×28 capacitive images already comprise over 14 times the amount of sensors of previous approaches based on flexible PCBs (*e.g.*, 64 [11] or 24 [53, 54] measurements), further increasing the resolution could help to improve classification accuracy. High-resolution capacitive images certainly benefit the blob matching due to clearer contact area boundaries and also benefit the MAE since more features of the finger become detectable. Possible technologies include *frustrated total internal reflection* (FTIR) that enables high-resolution multi-touch sensing [32] and infrared sensors integrated into the LCD layer similar to the SUR40 interactive display by Samsung. While these technologies are yet to be mass-produced for mobile devices, our prototype is based on hardware that is already publicly available which enables the community to reproduce *InfiniTouch*.

The accuracy of our model is well beyond the 80 % that previous work considered sufficient [41]. However, sufficiency also depends on the action's consequence (easily recoverable action vs. permanent action) and how inferences are translated to actions. For *InfiniTouch*, the consequence depends on the action that future developers implement while a wide range of translation approaches can further minimize accidental activations. For example, accidental activations of BoD gestures can be minimized using the confidence score of gesture recognizers, using thresholds for a minimum gesture length, or using heuristics to differentiate gestures from grip changes (*e.g.*, only one finger can move at a time). While our implementation works reliably for the implemented use cases, we also suggest that future BoD gestures should be designed with false positives and negatives in mind. Moreover, the flexion state example could also involve users in avoiding unintended actions by using visual elements to indicate the recognized flexion state (*i.e.* action).

Practicality of Use Cases

We designed the sample use cases solely to demonstrate the possibilities offered by *InfiniTouch*. Thus, we chose fingers and movements that are easy to explain and understand, but we also designed them to be ergonomically viable based on previous findings by Le *et al.* [45]. Designing our explicit BoD gestures, we considered these findings that showed that index and middle fingers can move comfortably within a large area (around top to center for similar devices) without grip changes and independent from the grip. This indicates that our presented BoD gestures can be performed comfortably without a grip change. Moreover, subtly bending the middle finger for the second use case also takes place within the comfortable area. As this paper focuses on the technical contribution, future work could investigate the comfort of such BoD gestures and how to communicate them to end users [52].

Reproducibility with Publicly Available Hardware

We presented an approach to prototype a full-touch smartphone with publicly available hardware. While we used an LG Nexus 5 as the basis, our approach can be applied with any smartphone. This enables the community to reproduce our prototype and use our model to explore finger-aware interaction with *InfiniTouch*. As a tradeoff, data from both touchscreens and the edge sensors need to be synchronized over network which adds a latency of $7.2\,ms$ while an additional hardware container is required. Despite an additional container, our prototype can still be used in mobile situations (*e.g.*, in walking studies) since the hardware container is designed to be fixated on the forearm. Moreover, smartphone manufacturers could produce proprietary components for future commodity smartphones so that a hardware container is not needed in a mass-market version. These components could comprise flexible PCBs with a sufficient amount of sensors. These are already used in consumer products (*e.g.*, the Microsoft Touch Mouse) and provide capacitive images of touches. Using such components would also avoid the synchronization of data over the network while manufacturers can directly use our model for finger-aware touch interaction on the whole device surface.

Specialization on Common One-Handed Grips

The model presented in this work focuses on one-handed grips. Previous work has shown that fingers can comfortably reach around 70 % of the back (for similar device sizes [46, 45]) during one-handed smartphone interaction without grip changes. This enables the fingers on the back to be used for a wide range of explicit (*e.g.* BoD/side gestures) and implicit interactions (*e.g.* flexion/grip sensing) to increase the expressiveness of one-handed touch input. Since our comprehensive dataset covers a wide range of typical one-handed grips as performed in the study, our model also works when some fingers of the holding hand are not touching and stays robust even when other hand parts (*e.g.* palm) are touching or releasing. Two-handed grips and further touches beyond usual one-handed grips (*e.g.*, using other body parts) are currently not expected by our model and would lead to unexpected estimations. However, with minor adaptations to the implementation, our model can even be used to identify finger positions of the holding hand while the other hand performs input on the front. While we focused on right-handed grips to show the feasibility of our approach, our procedure and publicly available source code enables researchers to easily extend our work to other devices and use cases (*e.g.*, left-handed or bimanual grips for tablets).

CONCLUSION

We presented *InfiniTouch*, a smartphone prototype that enables touch input on the whole device surface and identifies fingers touching the device with an accuracy of 95.78 %. In contrast to previous approaches for finger-aware input, our prototype is the first that does not require external hardware (*e.g.*, wearable sensors and cameras), has the form factor of a standard smartphone, and identifies all fingers during one-handed interaction with a usable accuracy. We further based our prototype on publicly available components and release schemes and source code so that researchers who want to explore finger-aware interaction on full-touch smartphones can re-implement our system. We presented and implemented a series of use cases to showcase our system.

As we are publicly releasing our dataset which includes the 3D motion data of all finger joints and the capacitive images, future work could train a model to reconstruct the hand posture that is used to hold the device. This enables a wide range of use cases such as transferring the hand into virtual reality, enabling pre-touch sensing with standard touchscreens, and predicting actions based on hand kinematics.

PROTOTYPE SCHEMES, DATASET, AND MODELS

We release the PCB scheme and 3D models of our prototype for the reader to reproduce our prototype. Moreover, we are publicly releasing the data set, the finger position estimation model, and our Android library for *InfiniTouch* to enable interested parties to run and explore finger-aware touch interaction on a full-touch smartphone. To enable the reader to extend our models (*e.g.*, for left-handed grips or to estimate all joints), we also provide our Jupyter notebooks (in Python) that contain the training and test procedures. The data can be found on https://github.com/interactionlab/InfiniTouch.

ACKNOWLEDGEMENTS

This work is supported through project C04 of SFB/Transregio 161, the MWK Baden-Württemberg within the Juniorprofessuren-Programm, and by the DFG within the SimTech Cluster of Excellence (EXC 310/2).

REFERENCES

1. Lisa Anthony and Jacob O. Wobbrock. 2012. $N-protractor: A Fast and Accurate Multistroke Recognizer. In *Proceedings of Graphics Interface 2012 (GI '12)*. Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 117–120. http://dl.acm.org/citation.cfm?id=2305276.2305296
2. Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. 2017. Structured Pruning of Deep Convolutional Neural Networks. *J. Emerg. Technol. Comput. Syst.* 13, 3, Article 32 (Feb. 2017), 18 pages. DOI: http://dx.doi.org/10.1145/3005348
3. Oscar Kin-Chung Au and Chiew-Lan Tai. 2010. Multitouch Finger Registration and Its Applications. In *Proceedings of the 22Nd Conference of the Computer-Human Interaction Special Interest Group of Australia on Computer-Human Interaction (OZCHI '10)*. ACM, New York, NY, USA, 41–48. DOI: http://dx.doi.org/10.1145/1952222.1952233
4. Jeff Avery, Sylvain Malacria, Mathieu Nancel, Géry Casiez, and Edward Lank. 2018. Introducing Transient Gestures to Improve Pan and Zoom on Touch Surfaces. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, 8. DOI: http://dx.doi.org/10.1145/3173574.3173599
5. Shiri Azenkot and Shumin Zhai. 2012. Touch Behavior with Different Postures on Soft Smartphone Keyboards. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services (MobileHCI '12)*. ACM, New York, NY, USA, 251–260. DOI: http://dx.doi.org/10.1145/2371574.2371612
6. Patrick Bader, Valentin Schwind, Niels Henze, Stefan Schneegass, Nora Broy, and Albrecht Schmidt. 2014. Design and Evaluation of a Layered Handheld 3D Display with Touch-sensitive Front and Back. In *Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational (NordiCHI '14)*. ACM, New York, NY, USA, 315–318. DOI:http://dx.doi.org/10.1145/2639189.2639257
7. Hrvoje Benko, T. Scott Saponas, Dan Morris, and Desney Tan. 2009. Enhancing Input on and Above the Interactive Surface with Muscle Sensing. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '09)*. ACM, New York, NY, USA, 93–100. DOI:http://dx.doi.org/10.1145/1731903.1731924
8. Joanna Bergstrom-Lehtovirta and Antti Oulasvirta. 2014. Modeling the Functional Area of the Thumb on Mobile Touchscreen Surfaces. In *Proceedings of the 32Nd*

Annual ACM Conference on Human Factors in Computing Systems (CHI '14). ACM, New York, NY, USA, 1991–2000. DOI: http://dx.doi.org/10.1145/2556288.2557354

9. Xiang Cao, A. D. Wilson, R. Balakrishnan, K. Hinckley, and S. E. Hudson. 2008. ShapeTouch: Leveraging contact shape on interactive surfaces. In *2008 3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems*. 129–136. DOI: http://dx.doi.org/10.1109/TABLETOP.2008.4660195
10. Elie Cattan, Amélie Rochet-Capellan, Pascal Perrier, and François Bérard. 2015. Reducing Latency with a Continuous Prediction: Effects on Users' Performance in Direct-Touch Target Acquisitions. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces (ITS '15)*. ACM, New York, NY, USA, 205–214. DOI:http://dx.doi.org/10.1145/2817721.2817736
11. Wook Chang, Kee Eung Kim, Hyunjeong Lee, Joon Kee Cho, Byung Seok Soh, Jung Hyun Shim, Gyunghye Yang, Sung-Jung Cho, and Joonah Park. 2006. Recognition of grip-patterns by using capacitive touch sensors. In *Industrial Electronics, 2006 IEEE International Symposium on*, Vol. 4. IEEE, 2936–2941.
12. Lung-Pan Cheng, Fang-I Hsiao, Yen-Ting Liu, and Mike Y. Chen. 2012. iRotate Grasp: Automatic Screen Rotation Based on Grasp of Mobile Devices. In *Adjunct Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST Adjunct Proceedings '12)*. ACM, New York, NY, USA, 15–16. DOI:http://dx.doi.org/10.1145/2380296.2380305
13. Lung-Pan Cheng, Hsiang-Sheng Liang, Che-Yang Wu, and Mike Y. Chen. 2013. iGrasp: Grasp-based Adaptive Keyboard for Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 3037–3046. DOI: http://dx.doi.org/10.1145/2470654.2481422
14. Ashley Colley and Jonna Häkkilä. 2014. Exploring Finger Specific Touch Screen Interaction for Mobile Phone User Interfaces. In *Proceedings of the 26th Australian Computer-Human Interaction Conference on Designing Futures: The Future of Design (OzCHI '14)*. ACM, New York, NY, USA, 539–548. DOI: http://dx.doi.org/10.1145/2686612.2686699
15. Christian Corsten, Bjoern Daehlmann, Simon Voelker, and Jan Borchers. 2017. BackXPress: Using Back-of-Device Finger Pressure to Augment Touchscreen Input on Smartphones. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 4654–4666. DOI: http://dx.doi.org/10.1145/3025453.3025565
16. Kiran Dandekar, Balasundar I Raju, and Mandayam A Srinivasan. 2003. 3-D finite-element models of human and monkey fingertips to investigate the mechanics of tactile sense. *Journal of biomechanical engineering* 125, 5 (2003), 682–691.
17. Alexander De Luca, Emanuel von Zezschwitz, Ngo Dieu Huong Nguyen, Max-Emanuel Maurer, Elisa Rubegni, Marcello Paolo Scipioni, and Marc Langheinrich. 2013. Back-of-device Authentication on Smartphones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2389–2398. DOI: http://dx.doi.org/10.1145/2470654.2481330
18. John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research* 12 (July 2011), 2121–2159. http://dl.acm.org/citation.cfm?id=1953048.2021068
19. Philipp Ewerling, Alexander Kulik, and Bernd Froehlich. 2012. Finger and Hand Detection for Multi-touch Interfaces Based on Maximally Stable Extremal Regions. In *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces (ITS '12)*. ACM, New York, NY, USA, 173–182. DOI: http://dx.doi.org/10.1145/2396636.2396663
20. Anna Maria Feit, Daryl Weir, and Antti Oulasvirta. 2016. How We Type: Movement Strategies and Performance in Everyday Typing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 4262–4273. DOI: http://dx.doi.org/10.1145/2858036.2858233
21. Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. 1977. An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Trans. Math. Softw.* 3, 3 (Sept. 1977), 209–226. DOI: http://dx.doi.org/10.1145/355744.355745
22. Emilien Ghomi, Stéphane Huot, Olivier Bau, Michel Beaudouin-Lafon, and Wendy E. Mackay. 2013. ArpèGe: Learning Multitouch Chord Gestures Vocabularies. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces (ITS '13)*. ACM, New York, NY, USA, 209–218. DOI: http://dx.doi.org/10.1145/2512349.2512795
23. Hyunjae Gil, DoYoung Lee, Seunggyu Im, and Ian Oakley. 2017. TriTap: Identifying Finger Touches on Smartwatches. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 3879–3890. DOI: http://dx.doi.org/10.1145/3025453.3025561
24. Nicholas Gillian and Joseph A. Paradiso. 2014. The Gesture Recognition Toolkit. *J. Mach. Learn. Res.* 15, 1 (Jan. 2014), 3483–3487. http://dl.acm.org/citation.cfm?id=2627435.2697076
25. Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics. (AISTATS'10)*, Vol. 9. JMLR.org, 249–256. http://www.jmlr.org/proceedings/papers/v9/glorot10a.html

26. Nan-Wei Gong, Jürgen Steimle, Simon Olberding, Steve Hodges, Nicholas Edward Gillian, Yoshihiro Kawahara, and Joseph A. Paradiso. 2014. PrintSense: A Versatile Sensing Technique to Support Multimodal Flexible Surface Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 1407–1410. DOI: http://dx.doi.org/10.1145/2556288.2557173

27. Emilio Granell and Luis A. Leiva. 2016. Less Is More: Efficient Back-of-Device Tap Input Detection Using Built-in Smartphone Sensors. In *Proceedings of the 2016 ACM on Interactive Surfaces and Spaces (ISS '16)*. ACM, New York, NY, USA, 5–11. DOI: http://dx.doi.org/10.1145/2992154.2992166

28. Anhong Guo, Robert Xiao, and Chris Harrison. 2015. CapAuth: Identifying and Differentiating User Handprints on Commodity Capacitive Touchscreens. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces (ITS '15)*. ACM, New York, NY, USA, 59–62. DOI: http://dx.doi.org/10.1145/2817721.2817722

29. Aakar Gupta, Muhammed Anwar, and Ravin Balakrishnan. 2016. Porous Interfaces for Small Screen Multitasking Using Finger Identification. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16)*. ACM, New York, NY, USA, 145–156. DOI: http://dx.doi.org/10.1145/2984511.2984557

30. Aakar Gupta and Ravin Balakrishnan. 2016. DualKey: Miniature Screen Text Entry via Finger Identification. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 59–70. DOI: http://dx.doi.org/10.1145/2858036.2858052

31. Charlotte Häger-Ross and Marc H Schieber. 2000. Quantifying the independence of human finger movements: comparisons of digits, hands, and movement frequencies. *The Journal of neuroscience* 20, 22 (2000), 8542–8550.

32. Jefferson Y. Han. 2005. Low-cost Multi-touch Sensing Through Frustrated Total Internal Reflection. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology (UIST '05)*. ACM, New York, NY, USA, 115–118. DOI: http://dx.doi.org/10.1145/1095034.1095054

33. Song Han, Huizi Mao, and William J. Dally. 2015. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. *CoRR* abs/1510.00149 (2015). http://arxiv.org/abs/1510.00149

34. David Holman, Andreas Hollatz, Amartya Banerjee, and Roel Vertegaal. 2013. Unifone: Designing for Auxiliary Finger Input in One-handed Mobile Interactions. In *Proceedings of the 7th International Conference on Tangible, Embedded and Embodied Interaction (TEI '13)*. ACM, New York, NY, USA, 177–184. DOI: http://dx.doi.org/10.1145/2460625.2460653

35. Christian Holz and Patrick Baudisch. 2011. Understanding Touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2501–2510. DOI: http://dx.doi.org/10.1145/1978942.1979308

36. Christian Holz, Senaka Buthpitiya, and Marius Knaust. 2015. Bodyprint: Biometric User Identification on Mobile Devices Using the Capacitive Touchscreen to Scan Body Parts. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 3011–3014. DOI: http://dx.doi.org/10.1145/2702123.2702518

37. Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, and others. 2003. A practical guide to support vector classification. (2003).

38. Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR* abs/1502.03167 (2015). http://arxiv.org/abs/1502.03167

39. Stefanie Jachner, G Van den Boogaart, Thomas Petzoldt, and others. 2007. Statistical methods for the qualitative assessment of dynamic models with time delay (R Package qualV). *Journal of Statistical Software* 22, 8 (2007), 1–30.

40. Amy K. Karlson and Benjamin B. Bederson. 2006. Studies in One-Handed Mobile Design: Habit, Desire and Agility. In *Proceedings of the 4th ERCIM Workshop on User Interfaces for All (UI4ALL)*.

41. Vassilis Kostakos and Mirco Musolesi. 2017. Avoiding Pitfalls when Using Machine Learning in HCI Studies. *interactions* 24, 4 (June 2017), 34–37. DOI: http://dx.doi.org/10.1145/3085556

42. Huy Viet Le, Patrick Bader, Thomas Kosch, and Niels Henze. 2016. Investigating Screen Shifting Techniques to Improve One-Handed Smartphone Usage. In *Proceedings of the 9th Nordic Conference on Human-Computer Interaction (NordiCHI '16)*. ACM, New York, NY, USA, Article 27, 10 pages. DOI: http://dx.doi.org/10.1145/2971485.2971562

43. Huy Viet Le, Thomas Kosch, Patrick Bader, Sven Mayer, and Niels Henze. 2018. PalmTouch: Using the Palm as an Additional Input Modality on Commodity Smartphones. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, 10. DOI: http://dx.doi.org/10.1145/3173574.3173934

44. Huy Viet Le, Sven Mayer, Patrick Bader, and Niels Henze. 2017. A Smartphone Prototype for Touch Interaction on the Whole Device Surface. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI EA '17)*. ACM, New York, NY, USA, Article 100, 8 pages. DOI: http://dx.doi.org/10.1145/3098279.3122143

45. 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 (CHI'18)*. ACM, New York, NY, USA. DOI: http://dx.doi.org/10.1145/3173574.3173605

46. Huy Viet Le, Sven Mayer, Katrin Wolf, and Niels Henze. 2016. Finger Placement and Hand Grasp During Smartphone Interaction. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '16)*. ACM, New York, NY, USA, 2576–2584. DOI: http://dx.doi.org/10.1145/2851581.2892462

47. Songil Lee, Gyouhyung Kyung, Jungyong Lee, Seung Ki Moon, and Kyoung Jong Park. 2016. Grasp and index finger reach zone during one-handed smartphone rear interaction: effects of task type, phone width and hand length. *Ergonomics* 59, 11 (2016), 1462–1472. DOI: http://dx.doi.org/10.1080/00140139.2016.1146346

48. Nicolai Marquardt, Johannes Kiemer, David Ledo, Sebastian Boring, and Saul Greenberg. 2011. Designing User-, Hand-, and Handpart-aware Tabletop Interactions with the TouchID Toolkit. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS '11)*. ACM, New York, NY, USA, 21–30. DOI:http://dx.doi.org/10.1145/2076354.2076358

49. Damien Masson, Alix Goguey, Sylvain Malacria, and Géry Casiez. 2017. WhichFingers: Identifying Fingers on Touch Surfaces and Keyboards Using Vibration Sensors. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology (UIST '17)*. ACM, New York, NY, USA, 41–48. DOI: http://dx.doi.org/10.1145/3126594.3126619

50. Fabrice Matulic, Daniel Vogel, and Raimund Dachselt. 2017. Hand Contact Shape Recognition for Posture-Based Tabletop Widgets and Interaction. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces (ISS '17)*. ACM, New York, NY, USA, 3–11. DOI: http://dx.doi.org/10.1145/3132272.3134126

51. Sven Mayer, Huy Viet Le, and Niels Henze. 2017. Estimating the Finger Orientation on Capacitive Touchscreens Using Convolutional Neural Networks. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces (ISS '17)*. ACM, New York, NY, USA, 220–229. DOI: http://dx.doi.org/10.1145/3132272.3134130

52. Sven Mayer, Lars Lischke, Adrian Lanksweirt, Huy Viet Le, and Niels Henze. 2018. How to Communicate New Input Techniques. In *Proceedings of the 10th Nordic Conference on Human-Computer Interaction (NordiCHI '18)*. ACM, New York, NY, USA, 13. DOI: http://dx.doi.org/10.1145/3240167.3240176

53. Mohammad Faizuddin Mohd Noor, Andrew Ramsay, Stephen Hughes, Simon Rogers, John Williamson, and Roderick Murray-Smith. 2014. 28 Frames Later: Predicting Screen Touches from Back-of-device Grip Changes. In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 2005–2008. DOI: http://dx.doi.org/10.1145/2556288.2557148

54. Mohammad Faizuddin Mohd Noor, Simon Rogers, and John Williamson. 2016. Detecting Swipe Errors on Touchscreens Using Grip Modulation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 1909–1920. DOI: http://dx.doi.org/10.1145/2858036.2858474

55. Roderick Murray-Smith. 2017. Stratified, computational interaction via machine learning. In *Eighteenth Yale Workshop on Adaptive and Learning Systems (New Haven, CT, USA*. 95–101.

56. Sundar Murugappan, Vinayak, Niklas Elmqvist, and Karthik Ramani. 2012. Extended Multitouch: Recovering Touch Posture and Differentiating Users Using a Depth Camera. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*. ACM, New York, NY, USA, 487–496. DOI: http://dx.doi.org/10.1145/2380116.2380177

57. Rajalakshmi Nandakumar, Vikram Iyer, Desney Tan, and Shyamnath Gollakota. 2016. FingerIO: Using Active Sonar for Fine-Grained Finger Tracking. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 1515–1525. DOI: http://dx.doi.org/10.1145/2858036.2858580

58. A Poston. 2000. Human engineering design data digest. *Washington, DC: Department of Defense Human Factors Engineering Technical Advisory Group* (2000).

59. Simon Robinson, Nitendra Rajput, Matt Jones, Anupam Jain, Shrey Sahay, and Amit Nanavati. 2011. TapBack: Towards Richer Mobile Interfaces in Impoverished Contexts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11)*. ACM, New York, NY, USA, 2733–2736. DOI: http://dx.doi.org/10.1145/1978942.1979345

60. Anne Roudaut, Mathias Baglioni, and Eric Lecolinet. 2009. TimeTilt: using sensor-based gestures to travel through multiple applications on a mobile device. In *IFIP Conference on Human-Computer Interaction*. Springer, 830–834.

61. Wenjie Ruan, Quan Z. Sheng, Lei Yang, Tao Gu, Peipei Xu, and Longfei Shangguan. 2016. AudioGest: Enabling Fine-grained Hand Gesture Detection by Decoding Echo Signal. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '16)*. ACM, New York, NY, USA, 474–485. DOI:http://dx.doi.org/10.1145/2971648.2971736

62. Erh-li Early Shen, Sung-sheng Daniel Tsai, Hao-hua Chu, Yung-jen Jane Hsu, and Chi-wen Euro Chen. 2009. Double-side Multi-touch Input for Mobile Devices. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems (CHI EA '09)*. ACM, New York, NY, USA, 4339–4344. DOI: http://dx.doi.org/10.1145/1520340.1520663

63. Shaikh Shawon Arefin Shimon, Sarah Morrison-Smith, Noah John, Ghazal Fahimi, and Jaime Ruiz. 2015. Exploring User-Defined Back-Of-Device Gestures for Mobile Devices. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '15)*. ACM, New York, NY, USA, 227–232. DOI: http://dx.doi.org/10.1145/2785830.2785890

64. Smith Aaron. 2015. A "Week in the Life" Analysis of Smartphone Users. http://www.pewinternet.org/2015/04/01/chapter-three-a-week-in-the-life-analysis-of-smartphone-users/. (2015). [last accessed 2018-02-02].

65. Kiseok Sung, Jay Cho, and Andris Freivalds. 2016. Effects of grip span in one-handed thumb interaction with a smartphone. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 60, 1 (2016), 1048–1052. DOI: http://dx.doi.org/10.1177/1541931213601243

66. Brandon T. Taylor and V Michael Bove. 2008. The Bar of Soap: A Grasp Recognition System Implemented in a Multi-functional Handheld Device. In *CHI '08 Extended Abstracts on Human Factors in Computing Systems (CHI EA '08)*. ACM, New York, NY, USA, 3459–3464. DOI: http://dx.doi.org/10.1145/1358628.1358874

67. Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning* 4, 2 (2012), 26–31.

68. Grigorios Tsoumakas and Ioannis Katakis. 2006. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining* 3, 3 (2006).

69. Ken Turkowski. 1990. Filters for common resampling tasks. In *Graphics gems*. Academic Press Professional, Inc., 147–165.

70. Radu-Daniel Vatavu, Lisa Anthony, and Jacob O. Wobbrock. 2012. Gestures As Point Clouds: A $P Recognizer for User Interface Prototypes. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction (ICMI '12)*. ACM, New York, NY, USA, 273–280. DOI: http://dx.doi.org/10.1145/2388676.2388732

71. Jingtao Wang and John Canny. 2004. FingerSense: Augmenting Expressiveness to Physical Pushing Button by Fingertip Identification. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems (CHI EA '04)*. ACM, New York, NY, USA, 1267–1270. DOI: http://dx.doi.org/10.1145/985921.986040

72. Wei Wang, Alex X. Liu, and Ke Sun. 2016. Device-free Gesture Tracking Using Acoustic Signals. In *Proceedings of the 22Nd Annual International Conference on Mobile Computing and Networking (MobiCom '16)*. ACM, New York, NY, USA, 82–94. DOI: http://dx.doi.org/10.1145/2973750.2973764

73. Andrew D. Wilson. 2010. Using a Depth Camera As a Touch Sensor. In *ACM International Conference on Interactive Tabletops and Surfaces (ITS '10)*. ACM, New York, NY, USA, 69–72. DOI: http://dx.doi.org/10.1145/1936652.1936665

74. Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li. 2007. Gestures Without Libraries, Toolkits or Training: A $1 Recognizer for User Interface Prototypes. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology (UIST '07)*. ACM, New York, NY, USA, 159–168. DOI: http://dx.doi.org/10.1145/1294211.1294238

75. Katrin Wolf, Christian Müller-Tomfelde, Kelvin Cheng, and Ina Wechsung. 2012a. Does Proprioception Guide Back-of-device Pointing As Well As Vision?. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems (CHI EA '12)*. ACM, New York, NY, USA, 1739–1744. DOI: http://dx.doi.org/10.1145/2212776.2223702

76. Katrin Wolf, Christian Müller-Tomfelde, Kelvin Cheng, and Ina Wechsung. 2012b. PinchPad: Performance of Touch-based Gestures While Grasping Devices. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction (TEI '12)*. ACM, New York, NY, USA, 103–110. DOI: http://dx.doi.org/10.1145/2148131.2148155

77. Pui Chung Wong, Hongbo Fu, and Kening Zhu. 2016. Back-Mirror: Back-of-device One-handed Interaction on Smartphones. In *SIGGRAPH ASIA 2016 Mobile Graphics and Interactive Applications (SA '16)*. ACM, New York, NY, USA, 10:1–10:5. DOI: http://dx.doi.org/10.1145/2999508.2999522

78. Robert Xiao, Julia Schwarz, and Chris Harrison. 2015. Estimating 3D Finger Angle on Commodity Touchscreens. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces (ITS '15)*. ACM, New York, NY, USA, 47–50. DOI: http://dx.doi.org/10.1145/2817721.2817737

79. Xiang Xiao, Teng Han, and Jingtao Wang. 2013. LensGesture: Augmenting Mobile Interactions with Back-of-device Finger Gestures. In *Proceedings of the 15th ACM on International Conference on Multimodal Interaction (ICMI '13)*. ACM, New York, NY, USA, 287–294. DOI: http://dx.doi.org/10.1145/2522848.2522850

80. Cheng Zhang, Anhong Guo, Dingtian Zhang, Caleb Southern, Rosa Arriaga, and Gregory Abowd. 2015. BeyondTouch: Extending the Input Language with Built-in Sensors on Commodity Smartphones. In

Proceedings of the 20th International Conference on Intelligent User Interfaces (IUI '15). ACM, New York, NY, USA, 67–77. DOI: http://dx.doi.org/10.1145/2678025.2701374

81. Yang Zhang, Gierad Laput, and Chris Harrison. 2017. Electrick: Low-Cost Touch Sensing Using Electric Field Tomography. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 1–14. DOI: http://dx.doi.org/10.1145/3025453.3025842
82. Yang Zhang, Junhan Zhou, Gierad Laput, and Chris Harrison. 2016. SkinTrack: Using the Body As an Electrical Waveguide for Continuous Finger Tracking on the Skin. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 1491–1503. DOI: http://dx.doi.org/10.1145/2858036.2858082
83. Jingjie Zheng and Daniel Vogel. 2016. Finger-Aware Shortcuts. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 4274–4285. DOI: http://dx.doi.org/10.1145/2858036.2858355