Investigating the Feasibility of Finger Identification on Capacitive Touchscreens using Deep Learning

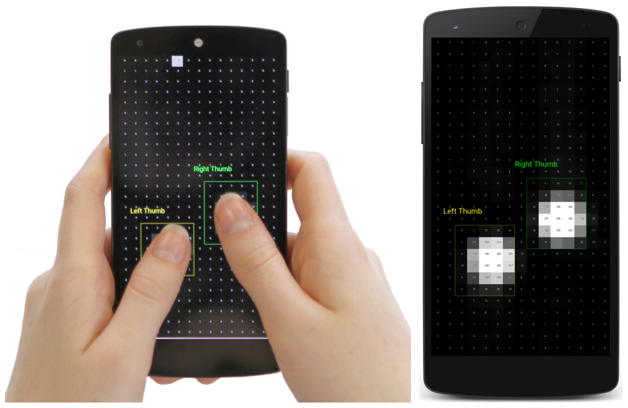
Huy Viet Le
University of Stuttgart
Stuttgart, Germany
mail@huyle.de

Sven Mayer
University of Stuttgart
Stuttgart, Germany
info@sven-mayer.com

Niels Henze
University of Regensburg
Regensburg, Germany
niels.henze@ur.de

ABSTRACT

Touchscreens enable intuitive mobile interaction. However, touch input is limited to 2D touch locations which makes it challenging to provide shortcuts and secondary actions similar to hardware keyboards and mice. Previous work presented a wide range of approaches to provide secondary actions by identifying which finger touched the display. While these approaches are based on external sensors which are inconvenient, we use capacitive images from mobile touchscreens to investigate the feasibility of finger identification. We collected a dataset of low-resolution fingerprints and trained convolutional neural networks that classify touches from eight combinations of fingers. We focused on combinations that involve the thumb and index finger as these are mainly used for interaction. As a result, we achieved an accuracy of over 92 % for a position-invariant differentiation between left and right thumbs. We evaluated the model and two use cases that users find useful and intuitive. We publicly share our data set (*CapFingerId*) comprising 455,709 capacitive images of touches from each finger on a representative mutual capacitive touchscreen and our models to enable future work using and improving them.



(a) Using two thumbs for interaction **(b) Raw Data**

Figure 1: Identifying left and right thumbs on a commodity smartphone using the raw capacitive data of touches.

CCS CONCEPTS

• **Human-centered computing → Interaction techniques**; *User studies*; *Laboratory experiments*; *Touch screens*;

KEYWORDS

Touchscreen; deep learning; capacitive image; finger-aware interaction; finger identification; smartphone

ACM Reference Format:
Huy Viet Le, Sven Mayer, and Niels Henze. 2019. Investigating the Feasibility of Finger Identification on Capacitive Touchscreens using Deep Learning. In *24th International Conference on Intelligent User Interfaces (IUI '19), March 17–20, 2019, Marina del Rey, CA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3301275.3302295

1 INTRODUCTION

Nearly all mobile devices incorporate a touchscreen as the main interface for interaction. Its combination of input and output in a single surface enables intuitive and dynamic user interfaces which can be operated with direct touch. Although the concept of direct touch feels natural to the user, the touch input vocabulary is limited compared to traditional input devices such as hardware keyboard and mouse. Capacitive touchscreens translate the raw data of fingers touching the display into 2D coordinates whereas the remaining raw data is omitted. With 2D coordinates alone, touch input lacks further dimensions which are fundamental to access secondary actions and shortcuts to frequently used functions. To extend the input vocabulary, a wide range of previous work in HCI focused on identifying individual fingers and other parts of the hand for touch interaction.

By differentiating between inputs of different fingers on the display, functions and action modifiers can be assigned to individual fingers. Performing the same input, but with different fingers, do now activate different functions similar to the use of multiple buttons on a computer mouse and modifier keys on keyboards. Previous work presented a wide range of promising use cases which ranges from improving text entry on small touch displays [25], through providing finger-aware shortcuts on touch keyboards [67], to enhancing multitasking on smartphones [24]. Accordingly, a wide range of hardware prototypes were presented which identify individual fingers based on sensors attached to the user [5, 24, 25, 47, 48] and device [54, 62, 64, 67]. While these approaches are accurate, they require additional sensors which reduces mobility and convenience. There is no standalone solution yet that identifies fingers on commodity smartphones.

One solution to avoid additional sensors for finger identification is to use the contact geometry of touches. Previous research focused predominantly on tabletops that provide high-resolution images of touches [2, 17, 18] to identify fingers based on multi-touch hand models. By modifying the firmware of smartphones, researchers used the raw data of commodity touchscreens (referred to as *capacitive images*) to infer further input dimensions. Capacitive images represent low-resolution fingerprints and can be used to estimate the finger orientation [50, 65], recognize body parts [32], palm touches [37, 42], and hand poses [53]. Gil *et al.* [19] used capacitive images of a smartwatch prototype to differentiate between touches of thumb, index, and middle finger. However, they used exaggerated poses on smartwatches so that each finger touched with a distinct angle. Expecting these poses does not only impact the usability but they are also not common and ergonomic for smartphone use (*e.g.*, touching with half the middle finger).

Previous work showed that capacitive images provided by mobile devices do not contain sufficient signal to identify each finger during regular interaction [19]. However, being able to differentiate between the primary input fingers (*e.g.*, right thumb) and others is already a useful addition to the input vocabulary. For example, a second finger could perform shortcuts, secondary actions, and even improve multitasking [24] or text entry [25]. Previous work required wearable sensors [24, 25, 48], sequential data such as gestures [45], pre-defined targets [6], or temporal features [66] to differentiate between a set of fingers (*e.g.*, left/right thumb). In contrast, we use *capacitive images* to identify fingers within single frames independent from context, position, and additional sensors. We collected a data set comprising of capacitive images for each finger and empirically studied finger combinations which can be differentiated with a usable accuracy. While a feature engineering approach with basic machine learning achieved inferior results, we present a user and position-independent deep learning model to differentiate between left and right thumbs with over 92 % accuracy. We evaluated it with novel use cases that users find intuitive and useful. Moreover, we publicly release our data set (*CapFingerId*) and models to enable future work to use and improve finger identification on commodity smartphones. Our contribution is threefold which includes (1) finger identification models, (2) data set, and (3) thumb-aware touch use cases.

2 RELATED WORK

Previous work presented a wide range of approaches to identify individual fingers independent from context and position of touches. They can be grouped into three categories: (1) attaching sensors to the user, (2) using cameras, and (3) interpreting the shape geometry of touches.

2.1 User-Worn Sensors

Finger identification approaches that attach sensors to the user generally yield the highest accuracies. Gupta *et al.* [24, 25] mounted infrared (IR) sensors on the index and middle finger to identify touches by these two fingers with an accuracy of 99.5 % upon individual calibration. Similarly, Masson *et al.* [48] achieved a recognition accuracy of 99.7 % on touchpads using vibration sensors attached to the user's fingers. Further approaches include using electromyography [5], gloves [47], and RFID tags [60] that are attached to the user to identify finger touches based on sensor measurements. Despite high accuracies, these approaches are not suitable for use outside of lab settings. Besides additional sensors attached to the user's fingers, they also require attached computing units which interpret the signal (*e.g.*, Arduino).

2.2 External Cameras

Another wide range of approaches focused on using cameras to identify touches from different fingers. Researchers predominantly used a combination of RGB cameras and computer vision [62, 67] to identify fingers for their prototypes; for example, Zheng *et al.* [67] used the built-in webcam of laptops to identify fingers and hands on the keyboard. Using depth cameras such as the Microsoft Kinect provides additional depth information for finger identification. Depth cameras were used by Murugappan [54] and Wilson [64] to implement touch sensors. The Leap Motion is a sensor device that uses proprietary algorithms to provide a hand model with an average accuracy of 0.7 *mm* [63]. Colley and Häkkilä [12] used a LeapMotion next to a smartphone to evaluate finger-aware interaction. While these approaches do not require users to wear additional sensors, they raise further challenges; amongst others, they reduce mobility and convenience as external cameras need to be attached to the device or mounted next to the touchscreen.

2.3 Imprint of Touch Inputs on Tabletops

One approach to avoid external sensors is to use the shape geometries of touches to differentiate between fingers and other parts of the hand. Tabletops are predominantly based on infrared cameras below the touch surface [8, 49] or *frustrated total internal reflection* [26]. These technologies provide images which represent imprints of touches performed on the interactive surface. Previous work in the tabletop domain used these images to explore novel touch-based interaction which goes beyond individual 2D touch locations [2, 8, 17, 18, 49]. For instance, Xiang *et al.* [8] used the touch images provided by the Microsoft PixelSense ($960 \times 540\,px$) to infer the posture of the hand while Ewerling *et al.* [17] used images of an IR camera ($640 \times 480\,px$) to differentiate between different hands and locations of individual fingers. However, these approaches cannot be used for finger identification on commodity mobile devices due to their immobile size.

2.4 Capacitive Touch Sensing

Mutual capacitive touchscreens are the predominant touch sensing technology for mobile devices. They comprise of spatially separated electrodes in two layers which are arranged as rows and columns [3]. To sense touches, the controller measures the change of coupling capacitance between two orthogonal electrodes, *i.e.* intersections of row and column pairs [13]. These measurements result in a low-resolution finger imprint which previous work also referred to as a *capacitive image* [23, 32, 37, 50]. Capacitive touchscreens of commodity smartphones comprise around 400 to 600 electrodes (*e.g.*, 15×27 electrodes with each being $4.1 \times 4.1\,mm$ on an LG Nexus 5). Although more electrodes enable a more detailed *capacitive image*, they also decrease the signal-to-noise ratio [9] while increasing the complexity of the manufacturing process. With electrode sizes of approximately $5 \times 5\ mm$, fingers and even stylus tips can already be precisely determined so that a higher sensing resolution becomes redundant regarding the translation of touches into 2D locations[1]. Figure 1 shows examples of touch input imprints.

Despite the low resolution, previous work presented a wide range of applications for the capacitive images [23, 32, 33, 37, 50, 61]. Not only do *capacitive images* provide promising means for biometric authentication [23, 32], but also enable to extend the touch input vocabulary by inferring additional features of the touching object. Using convolutional neural networks (CNNs), previous work showed that parts of the hand [32, 37] and the orientation of a finger [50, 65] can be estimated based on the touch imprint. Beyond the touchscreen, Le *et al.* [38, 39, 41] also used *capacitive images* of a fully touch sensitive smartphone to detect grips and estimate the 3D location of the holding fingers. Further use cases include grip adaptive interfaces [10, 11], touch prediction [51], and swipe error detection [52]. Closest to our work, Gil *et al.* [19] used *capacitive images* from a self-built smartwatch to differentiate between the thumb, index and middle finger. However, when not done in exaggerated poses (which is suitable for smartwatches but less for smartphone input), the classification accuracy is lower than 70 % which is not reliable enough for interaction.

2.5 Summary

Previous work showed that a reliable identification of each finger on commodity smartphones is not feasible due to the low resolution of *capacitive images*. As the majority of fingers are placed on the back when holding the device in common grips, we can assume that they are not used for input. Thus, reliably differentiating between combinations of input fingers (predominantly left/right thumbs and index fingers [15, 16, 40, 43, 55]) already extends the input vocabulary with useful features such as shortcuts and secondary actions. While finger identification is not a new challenge, there is no data set available which includes capacitive images of touches by each finger on a capacitive touchscreen. Such a data set is required to explore combinations of fingers which can be reliably differentiated. Due to steady advances in machine learning research, publicly releasing such a data set allows researchers and practitioners to improve the accuracy of finger identification and find new differentiable finger combinations in the future. Successful examples of steady accuracy improvements based on public data sets are the MNIST database[2] and ImageNet (ILSVRC).

3 RESEARCH APPROACH

We follow a data-driven approach similar to previous work [19, 37, 41] to explore the differentiation of finger pairs which can be used to enhance touch input. In particular, our research approach includes the following steps:

(1) *Gathering the Data Set*: We conduct a user study in which participants are instructed to use specific fingers to perform common touch input gestures. Since the input of participants are controlled with given instructions, all captured *capacitive images* are automatically labeled with the finger which performed the input and with the task during which input was performed.
(2) *Exploration and Model Development*: We explore the data set to provide an overview of the capacitive images for each finger and to find distinctive features which could be used for basic machine learning algorithms (*e.g.*, SVM, kNN, or random forests). We then train CNNs, a deep learning model specialized on image data, to investigate the feasibility of identifying fingers in different combinations. We use the data generated by 80 % of the participants to train the models and 20 % of the data to test the model. While training the models, we optimize the model parameters to achieve the highest accuracy on the test set.
(3) *Evaluation during Realistic Use*: As optimizing purely for the test set would introduce overfitting, we evaluate the generalization of our best model with a validation set (*i.e.*, how well they perform on unseen data). We conducted a second study in which we retrieve the validation set with similar tasks to determine the validation accuracy. Moreover, we evaluate the model accuracy with realistic use cases to validate beyond the data collection tasks, and to collect qualitative feedback on the perceived usability of the model.

[1]http://www.electronicproducts.com/Analog_Mixed_Signal_ICs/Communications_Interface/Touchscreens_large_and_small.aspx
[2]http://yann.lecun.com/exdb/mnist/

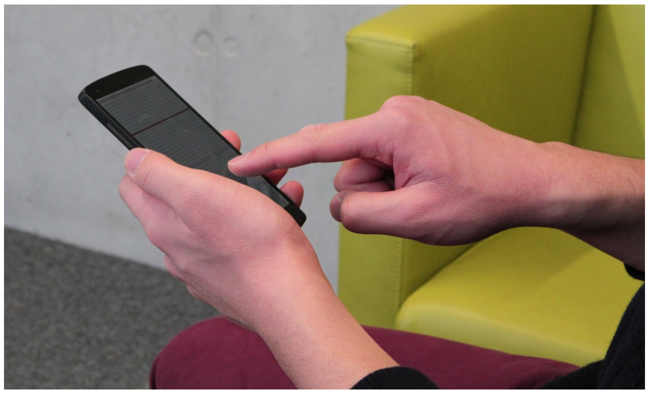


Figure 2: The study apparatus showing a participant solving a scrolling task with the index finger.

We focus on finger identification purely based on *capacitive images* and state-of-the-art deep learning techniques to show the feasibility of differentiating pairs of fingers. To enable future work to improve our results based on steady advances in machine learning research and specialized models, we publicly released our data set (see end of this paper).

4 DATA COLLECTION STUDY

We conducted a user study to collect labeled touch data while participants performed representative touch actions. This data enables us to train and evaluate models based on supervised learning for distinguishing between different fingers. We adopted the study design from previous work by Le *et al.* [37] who used tasks that cover

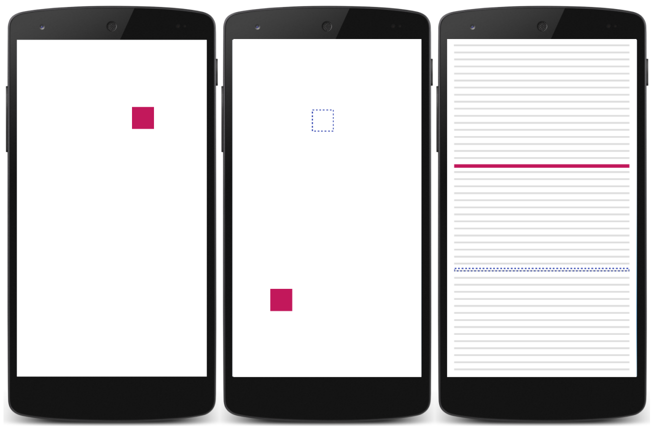

(a) Tapping task (b) Dragging task (c) Scrolling task

Figure 3: Tasks adapted from previous work [37] to cover representative inputs on mobile touchscreens.

typical touch gestures such as tapping, scrolling, and dragging to include representative actions.

4.1 Study Design & Tasks

To record representative touch input, we instructed participants to perform three different tasks with each of the ten fingers to generate data. The tasks are shown in Figure 3 and include a *tapping task* in which participants tapped and held the target for 1.5 seconds; a *scrolling task* in which a red line needs to match a blue line (horizontal and vertical); and a *dragging task* in which participants dragged a tile into a target. Targets and shapes appeared at randomized positions.

We used a 10×3 within-subjects design with the independent variables being the fingers and the tasks. With each finger, participants performed 30 repetitions of all three tasks resulting in $10 \times 30 \times 3 = 900$ tasks per participant. We further divided the 30 repetitions of the scrolling task into 15 vertical and 15 horizontal tasks to cover all scrolling directions. The order of fingers was balanced using a Latin square while the tasks appeared in a shuffled order.

4.2 Participants & Study Procedure

We recruited 20 participants (15 male and 5 female) with an average age of 22.4 ($SD = 2.8$). All except two were right-handed. The average hand size measured from the wrist crease to the middle fingertip ranged from 16.3 *cm* to 20.8 *cm* ($M = 18.9$ *cm*, $SD = 1.3$ *cm*). Our data includes samples from the 5th and 95th percentile of the anthropometric data [56]. Thus, the participants can be considered as representative.

After obtaining informed consent, we measured the participants' hand size and handed them an instruction sheet which explained all three tasks. Participants were seated on a chair without armrests and instructed to hold the device one-handed when touching with the thumb, and two-handed for all other fingers. We instructed participants to hold the device in the same angle for all fingers (*i.e.* the angle they used first) to avoid the models potentially overfitting to the angle between device and fingers (*e.g.*, participants shifting their grip or changing their body posture after a condition). On average, participants finished all tasks within 45 minutes including optional breaks. We reimbursed participants with 5 EUR for their participation.

4.3 Apparatus

We used an LG Nexus 5 with a modified kernel as described in previous work [37, 39] to access the 15×27 8-bit capacitive image of the Synaptics ClearPad 3350 touch sensor. Exemplary images of the raw capacitive data are shown in Figure 1, where each image pixel corresponds to a 4.1 *mm* × 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 developed an application for the tasks described above which logs a capacitive image every 50 *ms* (20 *fps*). Images were logged with the respective task name and finger to label each touch automatically. Figure 2 shows the study apparatus.

5 MODELS FOR FINGER IDENTIFICATION

We present our data set and describe three steps towards developing finger identification models: (1) cleaning the data set, (2) exploring the data set to understand distinctive features between touches of individual fingers, and (3) using deep learning to train models for finger identification.

5.1 Data Set & Preprocessing

We collected 921,538 capacitive images in the data collection study. We filtered empty images in which no touches were performed, as well as erroneous images in which more than one finger was touching the screen to avoid wrong labels. To train a position-invariant model and enable classification of multiple blobs within one capacitive image, we performed a blob detection, cropped the results and pasted each blob into an empty 15×27 matrix (referred to as *blob image*). The blob detection omitted all blobs that were not larger than one pixel of the image (4.1 *mm* × 4.1 *mm*) as these can be considered as noise of the capacitive touchscreen. In total, our data set consists of 455, 709 blob images (194, 571 while tapping; 111, 758 while dragging; 149, 380 while scrolling).

Table 1: Parameters of all fitted ellipses. Parameters a and b represent the length of minor and major semi-axes (in *mm*). θ represents the ellipse rotation in a counter-clockwise orientation in degrees.

			a		b		θ	
Hand	**Finger**	**Count**	**M**	**SD**	**M**	**SD**	**M**	**SD**
Left	Thumb	50,897	7.32	1.27	7.48	1.47	43.05	49.77
	Index	41,379	6.51	0.74	6.28	0.82	46.62	52.72
	Middle	39,079	6.64	0.84	6.38	0.91	46.09	52.03
	Ring	44,718	6.55	0.86	6.32	0.93	43.31	53.03
	Little	45,794	6.21	1.00	6.39	1.24	33.57	53.06
Right	Thumb	44,674	7.07	1.28	7.15	1.27	43.37	52.72
	Index	46,507	6.60	0.91	6.45	1.06	46.04	52.76
	Middle	47,082	6.73	0.95	6.55	1.10	51.86	49.33
	Ring	47,229	6.71	0.88	6.47	0.96	47.55	49.07
	Little	48,350	6.33	1.04	6.31	1.19	38.80	50.02

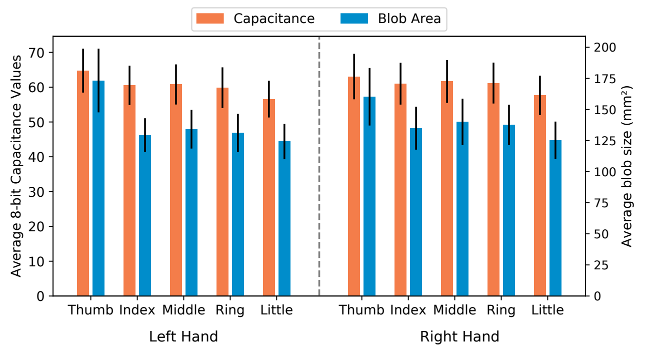


Figure 4: Average capacitances and blob sizes for each finger.

5.2 Data Set Exploration

We visually inspected the generated touch blobs of each finger during all tasks to find distinctive features. Figure 5 shows average touch blobs for each finger including the blob size and distribution of the measured capacitance. We generated these images by upscaling the capacitive images by a factor of 5 using the Lanczos4 algorithm [59] to increase clarity of the capacitance distribution. We then cropped the blobs and overlayed them for each finger. To describe the blobs, we fitted an ellipse around them using a 2D least squares estimator for ellipses[3]. The resulting ellipse parameters (minor-axis a, major-axis b, and orientation θ) in *mm* are averaged and shown in Table 1. We further explored the ellipse areas ($A = \pi * a * b$) and the average measured capacitance of a blob. We determined the average capacitance by averaging all electrode measurements of a blob larger than 0. Figure 4 shows the average capacitance (8-bit) and average blob size (in *mm*).

Similar to previous work [19, 37, 65], we used all five features (*i.e.*, mean capacitance, the ellipse area, a, b, and θ) to explore whether basic machine learning models based on feature engineering are sufficient for finger identification. For the sake of clarity, we focused on random forests over which we performed a grid search to find the best hyperparameters for each combination of fingers. Results are reported in Table 2 (Baseline (RF)) and are inferior to deep learning algorithms.

5.3 Convolutional Neural Networks

Deep learning algorithms such as CNNs learn features in part with the labeled input data and have been shown to be more successful than manual feature engineering [4]. Thus, we implemented CNNs using *Keras* (based on the *TensorFlow* backend) and performed a grid search as proposed by Hsu *et al.* [34] to determine the model parameters that achieve the highest test accuracy for all models as shown in Table 2. If we do not report a hyperparameter in the following, we applied the standard value (*e.g.*, optimizer settings) as reported in Keras' documentation. We started our grid search based on a CNN architecture which previous work found to perform the best on *capacitive images* [37, 41]. We performed our grid search as follows: We experimented with the number of convolution and dense layers in steps of 1. For the convolution part of the CNN, we varied the kernel size in steps of 1×1 and number of filters in steps of 16. For the dense layers, we experimented with the number of neurons in steps of 32. Moreover, we adapted the dropout factor in steps of 0.1. Figure 6 shows the final network architecture which achieved the highest test accuracy.

We trained the CNNs using an RMSprop optimizer [58] (similar to AdaGrad [14] but with a less radical approach to decrease the learning rate) with a batch size of 100. Further, we used the Xavier initialization scheme [20] to initialize the network weights. We used L2 regularization with a factor of 0.05, a 0.5 dropout after each pooling layer and the dense layer, and Batch Normalization to prevent overfitting during training. Our model expects a 15×27 *blob image* as input and returns the probability of each class (*i.e.* finger) as output.

5.4 Models and Accuracies

Table 2 shows the models that we trained and their accuracies on a test set. We trained and tested all models with a participant-wise split of 80% to 20% (16:4) to avoid samples of the same participant being in both training and test set.

The THUMB L/R and INDEX L/R models differentiate between touches of the respective finger from the left hand and the right hand. While the INDEX L/R model achieved an accuracy of 65.23 %, the THUMB L/R model discriminates left and right thumbs with an accuracy of 90.12 %. Differentiating between thumb and index finger independent from the hand (THUMB/INDEX) is feasible with an accuracy of 84.01 %. Similarly, identifying whether a touch was performed by the thumb or any other finger (THUMB/OTHERS) yields an accuracy of 86.44 %.

Identifying touches from the left or the right hand (HAND L/R) is feasible with an accuracy of 59.23 %. We further explored the differentiation between three fingers (*i.e.* thumb, index, and middle finger) similar to previous work by Gil *et al.* [19]. With our TRITAP model, we improved their accuracy by 2.92 % which results in 70.92 %. Increasing the number of fingers to identify decreases the accuracy. A hand-independent finger identification (5 FINGERS) leads to an accuracy of 46.13 % while additionally differentiating between hands (10 FINGERS) yields an accuracy of 35.55 %.

In addition, we trained models using a subset of the data set consisting of touches of the tapping task (*Tap Data*). Similar to Gil *et al.* [19], we achieved improvements in accuracy of up to 3.75 % compared to the full data set. Moreover, we trained models for each participant (*user-based models*) using their full datasets with a 80%:20% split sorted by timestamps. This increased the average accuracy by up to 32.4 % while reaching maximum accuracies of 80 % to 99 % per user. The improvements are substantial for 10 FINGERS, 5 FINGERS, TRITAP and INDEX L/R but not for models such as THUMB L/R with an already high accuracy. Out of all models, the THUMB L/R and THUMB/OTHERS achieved the highest accuracy.

5.5 Discussion

We started the model development by exploring the data set and training random forests based on features derived from the capacitive images. The results did not reveal any distinctive features

[3]http://scikit-image.org/docs/dev/api/skimage.measure.html#skimage.measure.EllipseModel

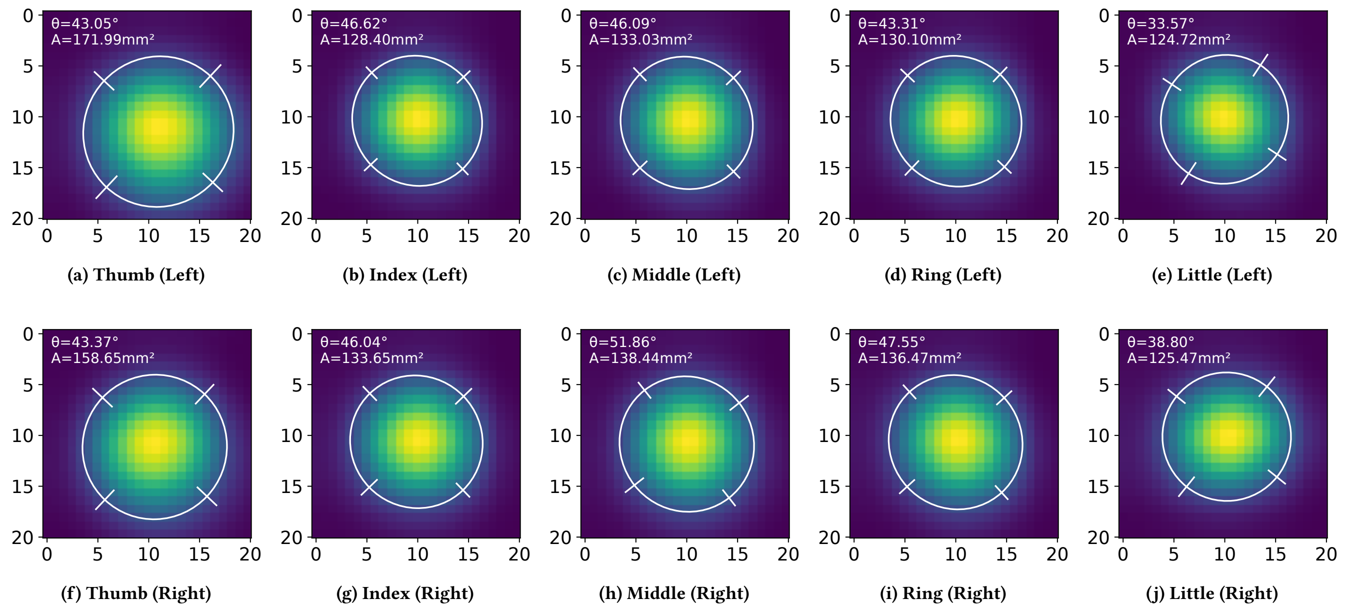


Figure 5: Average capacitive image for touches of each finger upscaled by a factor of 5 (for clarity purposes). Fitted ellipses represent the average area of touches in *mm* and the orientation θ thereof in degrees. The bars represent the standard deviation of the minor-axis a and major-axis b.

Table 2: Accuracies for differentiating between finger combinations. The first two columns show the accuracy on the test set based on a participant-wise 80%:20% (16:4) split. The third to sixth columns show user-based accuracies averaged over participants with a 80%:20% split (sorted by timestamp). ZeroR represents the baseline accuracy (using most frequent label) and Basic/RF represents the accuracy of random forests and feature engineering.

	Full Data	Tap Data	M_{user}	SD_{user}	Min_{user}	Max_{user}	Classes	Baseline (ZeroR)	Baseline (RF)
THUMB L/R	90.12 %	93.14 %	88.61 %	7.18 %	72.17 %	97.30 %	2	52.97 %	66.20 %
INDEX L/R	65.23 %	64.31 %	88.63 %	7.39 %	67.37 %	99.87 %	2	51.21 %	54.34 %
THUMB/INDEX	84.01 %	81.81 %	89.11 %	5.77 %	74.95 %	98.04 %	2	54.04 %	73.59 %
THUMB/OTHERS	86.44 %	88.89 %	84.52 %	12.62 %	48.37 %	95.55 %	2	78.92 %	79.91 %
HAND L/R	59.27 %	62.18 %	63.34 %	15.99 %	37.83 %	89.70 %	2	50.90 %	50.54 %
TRITAP	67.17 %	70.92 %	82.12 %	6.63 %	68.67 %	95.44 %	3	31.73 %	56.54 %
5 FINGERS	46.13 %	47.15 %	64.35 %	7.86 %	48.87 %	79.07 %	5	21.08 %	32.14 %
10 FINGERS	35.55 %	37.86 %	67.95 %	7.44 %	58.67 %	83.91 %	10	11.60 %	17.93 %

which basic machine learning algorithms could use for finger identification. Thus, we applied CNNs to develop models to differentiate between combinations of fingers. The achieved accuracies are shown in Table 2.

As expected, the model for identifying 10 FINGERS leads to an accuracy of 35.55 %, which is not practical for interaction. Confirming previous work by Gil *et al.* [19], this indicates that the information provided by the low-resolution capacitive images does not reveal enough differences between the fingers. To improve upon this, we then combined the same fingers of both hands into one class (5 FINGERS model) to achieve a higher accuracy (46.13 %). However, when considering the improvement factor over the baseline as suggested by Kostakos and Musolesi [35], we found that this factor decreases when combining fingers of both hands (2.1 for 10 FINGERS, 1.2 for 5 FINGERS). Similarly, combining all fingers of a hand into one class (HAND L/R) leads to an accuracy of 59.27 % but with an even lower improvement factor of 0.2. Moreover, discriminating thumbs from other fingers (THUMB/OTHERS) resulted in an improvement factor of 0.1. This further suggests that combining touches from multiple fingers into one class leads to more overlaps between classes and a decrease of accuracy improvements over the baseline. These results suggest that involving multiple fingers and classes in the classification leads to accuracies that are not sufficient for interaction.

To improve the accuracy, we explored models to differentiate between the two fingers mainly used for input: THUMB L/R, INDEX L/R, and THUMB/INDEX. While INDEX L/R and THUMB/INDEX achieved accuracies of 65.23 % and 84.01 % respectively, THUMB L/R achieved the highest accuracy of all models (90.12 %). The high accuracy of the THUMB L/R model could be due to different reasons. We observed that the thumb does not touch the display in a nearly perpendicular angle as other fingers do. This results in a larger contact surface which provides more information for classification. Amongst others, this includes the thumb's yaw angle (angle between thumb and vertical axis of the touchscreen) which is different for touches of the left and the right thumb (see Figure 1 and yellow hotspots in Figure 5). While this works for the CNN, the pure orientation of the blob is not sufficient for basic ML algorithms due to the high standard deviation.

In an interaction scenario, fingers should be identified directly after touching the display. Since the first touch is always a tap, we trained models using only the tap data. We achieved accuracy improvements of up to 3 % (*e.g.*, 93.14 % for THUMB L/R) as moving fingers add additional noise, especially at a lower frame rate. We further explored user-based models as collecting touches for on-device training works similar to the setup of fingerprint scanners. While THUMB L/R did not improve, the 10 FINGERS model improved by over 32 %. The accuracy increase for user-based models could be explained by individual postures (*e.g.* orientation) of each finger which resulted in differentiable touch shapes. Our models can be applied to other devices by retraining or scaling the raw data.

In summary, we found that reducing the number of fingers to identify increases the overall accuracy. While identifying all 10 fingers is not sufficiently accurate on capacitive touchscreens of commodity smartphones, differentiating between the left and right thumb is feasible with an accuracy of over 92 %. This extends the touch input vocabulary as the second thumb can be used for secondary actions, similar to the right mouse button. Moreover, previous work showed that using both thumbs is already a common posture for most users [15, 16, 43, 55]. In addition to an offline validation, we demonstrate the usefulness of our THUMB L/R model, suitable use cases, and the model's accuracy during real use cases on a commodity smartphone in the following.

6 MOBILE IMPLEMENTATION & USE CASES

We present our mobile implementation of the THUMB L/R model, and further use cases made possible with the model.

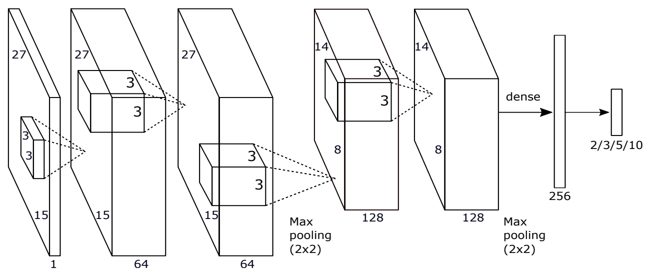


Figure 6: General architecture used after performing an initial grid search for all finger combinations listed in Table 2.

6.1 Mobile Implementation

After freezing and exporting the trained model into a protocol buffer file, we used *TensorFlow Mobile* for Android to run the CNN on an LG Nexus 5. A classification including blob detection and cropping takes 19.2 *ms* on average (*min* = 12 *ms*, *max* = 25 *ms*, *SD* = 4.2 *ms*) over 1000 runs. As this is faster than the 20 *fps* sampling rate for the capacitive images, the inference can be performed on each sample in the background. Since recent processors (*e.g.*, Snapdragon 845) are optimized for machine learning, the classification time can be reduced to a neglectable duration. The model can be further optimized for mobile devices with techniques such as quantization [27] and pruning [1] for a small loss of accuracy.

6.2 Sample Use Cases

We implemented two use cases for thumb identification which we evaluated in a study as described below.

6.2.1 Multitasking with Porous User Interfaces. Gupta *et al.* [24] presented *Porous User Interfaces*, which is a concept that overlays two applications. Thereby, one finger interacts with the application in the foreground while the other interacts with the background application. We implemented an abstract scenario based on the tasks used in the data collection study (see Figure 7b). Users can scroll (horizontal bar) with the right thumb while they can drag the square with the left thumb. This use case enables users to interact with two applications (here: dragging and scrolling) simultaneously without switching between applications. When touching the display, the upper left corner displays the recognized finger and thus which action users are performing. Concretely, the concept could be applied for purposes such as multitasking (*e.g.*, instant messaging in the foreground, calendar in the background) and exchanging data between applications (*e.g.*, dragging a file from a file manager application in the background into a messaging application in the foreground).

6.2.2 Painting Application. We implemented a painting application in which users can draw using the right thumb and use the left thumb for secondary tools (*e.g.*, erasing or selecting colors using a pie menu). In contrast to common painting applications, the full screen space can be used for painting instead of sharing the space between painting and menu area. Similar to the porous interfaces, the upper left corner displays which thumb was recognized and thus which action the user is performing (see Figure 7c).

6.3 Further Use Cases

We present further use cases for thumb-aware touch. We envision that the functions can be inverted so that left-handed and right-handed people can use them. In the following, we refer to the thumb of the dominant hand as the first thumb while the other thumb is referred to as the second thumb.

6.3.1 UI Components with Multiple Functions. Differentiating between two thumbs can be used similarly to two buttons on a hardware mouse. For example, touching with the first thumb in a file manager can be used to select and open files while touching with the second thumb can be used to open a context menu for the selected file. Previous work presented GUI widgets that could be used with our model [7].

6.3.2 3D Navigation. To navigate in 3D space, two thumbs can be used to navigate in different dimensions. For example, the first thumb rotates the view while the second thumb enables one to move within the 3D space. This is similar to using arrow keys and a mouse.

6.3.3 Handedness-Aware UI. While most user interfaces are designed for right-handed users, they offer a setting to flip the layout. Our model enables an adaptive layout based on the recognized thumb.

7 EVALUATION STUDY

We conducted a study to validate the model's accuracy and to evaluate our sample applications with users. We focused on the following two aspects: 1) model validation with new participants and thus a dataset that was not involved in training and test, and 2) collecting qualitative feedback on the sample use cases and the concept of *thumb-aware interaction.*

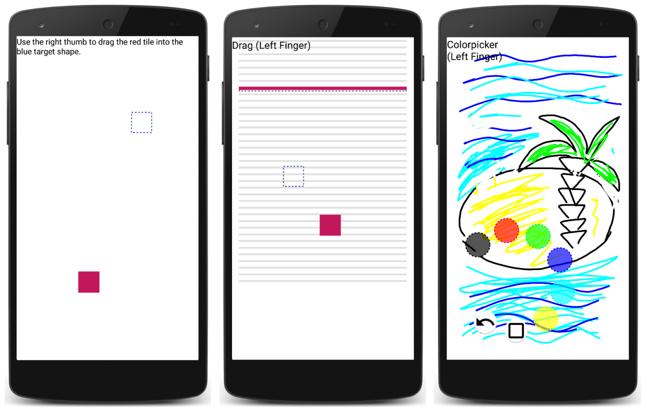
7.1 Study Procedure and Design

We designed three tasks to evaluate the two aspects described above. After we obtained informed consent from participants, we measured their hand sizes and collected demographic data. We handed them an instruction sheet that explained all parts of the study so they could refer to the instructions at any time.

7.1.1 Part 1 (Model Validation). In this part, we collect the validation set to evaluate the model performance with data from different participants than the ones used to train and test the model. We used the same tasks as in the data collection study (see Figure 3) and instructed participants to perform dragging, tapping, and scrolling tasks in a randomized order. All tasks were performed with the left and the right thumb in a counterbalanced order so that we could collect ground truth labels for the validation of the THUMB L/R model. Additionally, participants filled in a raw NASA-TLX questionnaire [22, 29] to compare the perceived workload with results from part 2.

7.1.2 Part 2 (Abstract Porous Interface). In addition to the first part, we evaluate the effective accuracy which includes the model's classification accuracy and human errors. The human error describes the user's error-proneness to use the correct fingers to solve the tasks. To do so, we used the porous interface application to instruct participants to solve dragging and scrolling tasks with different thumbs. To collect ground truth labels for accuracy evaluation, new targets appear as soon as the previous target was filled (*e.g.*, in Figure 7b a new target for dragging appeared after the scrolling target was filled). Thus, the current task (*e.g.*, dragging in Figure 7b) can be used as the ground truth label. We asked participants to fill in a NASA-TLX questionnaire to assess the perceived workload for using the correct thumb to solve the task.

7.1.3 Part 3 (Painting Application). To evaluate the THUMB L/R model in a concrete scenario, we used the painting application shown in Figure 7c in which users can draw using the right thumb


Use the right thumb to drag the red tile into the blue target shape.

(a) Dragging **(b) Porous UI** **(c) Painting**

Figure 7: Screenshots of (a) a dragging task in part 1; (b) a combined dragging and scrolling task as an abstract porous interface in part 2; (c) the drawing application in part 3 with a pie menu for color selection.

and use the left thumb for secondary tools (*e.g.*, erasing or selecting colors using a pie menu). Similar to the previous part, the upper left corner displays which thumb was recognized and thus which action the user is performing. We use this part to collect qualitative feedback from participants on the concept of *thumb-aware interaction* on a commodity smartphone. The qualitative feedback includes a questionnaire for ratings, and an interview focused on the advantages and disadvantages of the interaction method. Further, we asked for use cases that participants envisioned for *thumb-aware interaction* on smartphones.

7.2 Participants

We recruited 10 participants (6 male, 4 females) with an average age of 24.1 (SD = 3.0) who had not participated in the previous study. All participants were right-handed. The average hand size measured from the wrist crease to the middle fingertip ranged from 17.3 cm to 21.0 cm (M = 18.5 cm, SD = 1.1 cm). We reimbursed participants with 5 EUR for their participation.

8 RESULTS

We present the evaluation results which covers a model validation, the effective (model and human) accuracy in an abstract use case, and qualitative feedback on *thumb-aware interaction.*

8.1 Model Validation

Based on the collected capacitive images of new participants, the THUMB L/R model (trained with full data) achieved a mean accuracy of 89.78 % (SD = 3.30 %, min = 84.90 %, max = 96.50 %). The mean precision for detecting the left thumb was 88.72 % (SD = 4.43 %, min = 82.31 %, max = 95.68 %) and the recall was 89.85 % (SD = 3.90 %, min = 82.12 %, max = 95.87 %). Pearson's correlation test did not reveal a significant correlation between the hand size and accuracy ($r = -0.03$, $p = 0.94$).

A validation of the THUMB L/R model (trained with tap data) with the tap data subset resulted in a mean accuracy of 91.98 % (*SD* = 5.24 %, *min* = 81.98 %, *max* = 99.23 %). The mean precision for detecting the left thumb was 90.80 % (*SD* = 4.40 %, *min* = 85.29 %, *max* = 98.84 %) and the recall was 91.77 % (*SD* = 7.81 %, *min* = 77.15 %, *max* = 99.48 %). Again, Pearson's correlation test did not reveal a significant correlation between hand size and accuracy ($r = -0.04, p = 0.92$).

8.2 Effective Accuracy in Porous Interfaces

Based on the performed task as ground truth (*i.e.*, scroll or drag), the following results represent the effective accuracy when considering both model and human errors. Human errors occured when participants mistake, *e.g.*, the left for the right thumb for the scroll task. Therefore, these results describe the accuracy that one can expect when also considering the error-proneness of users to use the correct thumb for the tasks.

When classifying touches using the THUMB L/R model (trained with full data), the effective accuracy was 85.16 % (*SD* = 3.50 %, *min* = 78.16 %, *max* = 91.36 %) with a precision of 86.77 % (*SD* = 3.60 %, *min* = 81.19 %, *max* = 92.34 %) and recall of 84.17 % (*SD* = 4.74 %, *min* = 74.03 %, *max* = 89.96 %) for detecting the left thumb. Pearson's correlation test did not reveal a significant correlation between the participant's hand size and classification accuracy ($r = -0.46, p = 0.18$).

8.3 Subjective Feedback

We present the subjective feedback on the use cases. For the interviews, two researchers employed a simplified version of qualitative coding with affinity diagramming [28] by coding the answers, printing them on paper cards, and finally clustering the answers.

8.3.1 Perceived Workload Ratings. We used a raw NASA-TLX questionnaire [22] to assess participants' perceived workload after using the abstract porous interface. Moreover, we assessed the perceived workload after part 1 as a comparison. Mauchly's Test of Sphericity indicated that the assumption of sphericity had not been violated, $\chi^2(2) = .745, p = .689$. A one-way ANOVA with repeated-measures does not reveal any statistically significant differences ($F_{2,18} = 2.711, p = .093$) between the perceived cognitive load when using the left hand (*M* = 13.3, *SD* = 9.2), right hand (*M* = 7.3, *SD* = 7.3), or both hands for the abstract porous interface task (*M* = 11.2, *SD* = 6.1).

8.3.2 Interview. When asked about the first impression after using *thumb-aware interaction*, the majority (8) provided positive feedback. In particular, participants found it useful in general ("*very useful*" - P7), for painting applications ("*it is faster, especially since one can switch color with the left hand*" - P1), for multitasking purposes ("*very useful, especially to use two apps simultaneously*" - P5), and to avoid unintended touches ("*one can not activate something unintentionally*" - P4). They commended the idea ("*cool and innovative idea*" - P10) especially for the abstract porous interface task ("*the first task is easier to solve with two thumbs*" - P5) and the painting task ("*makes painting easier, even quite good when holding the device slightly different*" - P1). Moreover, they (6) found the interaction method intuitive ("*more intuitive [than without]*" - P7) and easy to learn ("*I am already used to using both thumbs. This makes learning this interaction method easier.*" - P6).

Confirming the perceived workload ratings, participants found that they had no difficulties to coordinate the thumbs for the two layers of the porous interface ("*I had no cognitive difficulties*" - P2, "*Needed to get used to in the beginning, but then it became easy*" - P4). Only one participant (P3) mentioned that it might be "*confusing to focus on two things simultaneously*". While two participants were impressed by the finger identification accuracy ("*Recognition was already very good - there were only two cases in which my finger was wrongly identified.*" - P5), other (6) participants clearly noticed it when fingers were wrongly identified ("*little bit frustrating since false recognitions leads to [unintended lines] that needs to be erased*" - P7). However, in the porous interface task, such identification errors could be "*easily fixed by touching the display once again*" (P5). Further, P5 noted that he "*quickly learned how to [place] the thumbs to control [the user interface]*".

When asked about use cases which they envision for *thumb-aware interaction*, all participants were unanimous about multitasking and shortcuts as the main use case. Moreover, they suggested using the interaction method for mobile games and image editing. For example, applications could offer multiple modes that make use of the porous interface concept (P9, P10) to avoid manual switches. Further, *thumb-aware interaction* could be used to interact with 3D objects so that each finger manipulated one dimension (P2, P5, P9). This would also benefit mobile games so that each finger could be assigned to one joystick or button so that fixed positions for control elements would not be required (P1, P4, P6). When asked about which fingers participants would use if all 10 fingers could be recognized with a sufficient accuracy, participants were unanimous that the thumb is the main finger for interacting with smartphones. Further, 4 participants considered the index finger for interaction while 2 would additionally consider the middle finger. To interact with tablets on a table, all participants would use all fingers while one participant further suggested using knuckles. In general, nine participants would use the concept of thumb-aware interaction on their devices ("*definitely, if apps support it*" - P4) while one would not.

9 DISCUSSION

We conducted a user study to validate the accuracy of the THUMB L/R model with participants who had not participated in the data collection study. Further, we combined the model validation with an evaluation of two use cases that we implemented using *thumb-aware touch interaction*. This includes an abstract scenario of porous interfaces initially proposed by Gupta *et al.* [24], and a painting application in which the right thumb can draw while the left thumb is responsible for the settings (*e.g.*, color and tool selection).

9.1 Model Validation Accuracy and Qualitative Feedback

The model validation resulted in accuracies similar to the results achieved in the offline validation with the test set. This suggests that the THUMB L/R model generalizes well across different users and does not overfit. We also considered human errors (*i.e.*, mixing up between fingers) together with the model accuracy which resulted in an effective accuracy of 85.16 %. The 5 % difference in contrast to the model validation could be due to a number of reasons. Human errors are inevitable especially since users are not yet fully familiar with this interaction concept. This conforms with the statements in the interview. Further, there are technical limitations that affect the accuracy of this live scenario. Due to the low retrieval rate of the capacitive images (20 *fps*), the classification could have happened on images in which the thumb was still in motion so that it just barely touched the display. While one solution could be using multiple frames and taking the most predicted class, this would have introduced latency.

Despite a decrease of 5 % accuracy in a live scenario, participants were positive about the use cases for *thumb-aware interaction* and argued that wrong classifications could be fixed effortlessly by placing the finger on the display again. One participant even mentioned that he learned how to place the thumb on the screen to avoid wrong classifications after the first iterations. The qualitative feedback revealed that participants were unanimously positive about the interaction method and that it can be a useful addition to the touch input vocabulary. Moreover, the ratings showed that interacting with porous interfaces using *thumb-aware interaction* does not increase the perceived workload. This suggests that interacting with two applications simultaneously can be intuitive for users and further avoids repeatedly switching between applications or splitting the screen which decreases the interaction space. Shortcuts (*e.g.*, pie menu for color selection) were perceived as intuitive and can save screen space that is used for menu bars otherwise. However, wrong identifications are reportedly more noticeable in this use case.

9.2 Improving the Classification Performance

While the thumb models (*i.e.*, THUMB L/R, THUMB/INDEX, and THUMB/OTHERS) achieved accuracies well beyond the 80 % that previous work considered sufficient in general [35], sufficiency also depends on the action's consequence (*e.g.*, easily recoverable action vs. permanent action) and how classifications are translated to actions. While the consequence depends on the application/developer, we discuss translation approaches and improvements that can further minimize accidental activations to a neglectable amount in the following.

Instead of translating a single classification result into an action, previous work showed that taking the majority class of a set of results noticeably improves the accuracy (*i.e.*, majority voting [36]). Since multiple results are considered, single incorrect results (*e.g.*, due to outliers) can be compensated. This is especially useful for recoverable actions and scenarios that provide enough time to gather multiple classifications (*e.g.*, finger identification while performing a gesture). Further, a threshold for the confidence score [46] of the most likely class could be used to avoid incorrect translations due to similarities. In case of a low confidence score, a touch could be either omitted with a warning to the user, or a fallback function could be activated that can easily be recovered. Especially with recoverable functions in case of a wrong identification, the system can collect touch data in the background to continuously improve the finger identification model using on-device learning.

Our approach is solely based on capacitive images to investigate the feasibility of identifying fingers within a single frame and independent from context and position. Finger identification, in general, could be improved with additional context information from the touchscreen or additional sensors. The touch position provides more information about the finger's yaw angle for thumb identification since distant touches (*e.g.*, close to top edge) lead to larger contact surfaces due to a stretched thumb. Similarly, touch offsets on smaller targets (*e.g.*, right thumb tends to hit the right of the target and vice versa for the left thumb) represent an additional feature to predict hand postures [6]. Further, gestures (*e.g.*, unlock trajectories) could be used to detect the handedness of users [45] and combined with the majority voting approach described above. Sequential models (*e.g.*, recurrent neural networks (RNN) and long short-term memory (LSTM)) can be trained with sequences of capacitive images (*i.e.*, trajectories of touches) to consider the information that gestures provide for detecting handedness.

Besides software-based approaches, touchscreens with a higher sensing resolution could be used. The Samsung SUR40 display offers touch images in a higher resolution based on IR sensing which contain more signal to improve the classification accuracy. However, such touchscreens need yet to be produced and incorporated into mass-market mobile devices. Not only are they more complex to manufacture but would also need more resources to be operated. Further improvements includes pre-touch sensing [31] to sense the finger above the display or built-in inertial measurement units [21, 30, 44, 57].

10 LIMITATIONS

While we showed a usable accuracy for using the THUMB L/R model for interaction, we focused solely on capacitive images as the source of input. This shows the feasibility of this approach as a standalone solution without the support of further information that could be dependent of context and position. It is left to future work to combine other sources of input as discussed above (*e.g.*, IMUs, touch trajectories, etc.) to further increase the classification accuracy.

By classifying only between the main input finger and a second finger, we increased the accuracy to a practical level compared to a ten-finger model. Since the model expects only certain fingers (*e.g.*, thumbs), all touches from other fingers are also treated as thumbs. One solution for application developers would be to introduce a mode exclusively for thumbs to avoid confusing the user. Another solution would be to recognize whether the touching finger is a thumb using a preceding model (*e.g.*, THUMB/OTHERS or IMUs).

11 CONCLUSION AND FUTURE WORK

In this paper, we investigated finger identification models based on deep learning and the capacitive images of commodity touchscreens. While previous work showed that capacitive images from mobile devices do not contain sufficient signal for a reliable identification of each finger, they also showed that thumbs and index fingers are predominantly used for input on mobile devices. Thus, we focused on identifying fingers within different combinations of fingers mainly used for input. We present an exploration of capacitive images for each finger which revealed that an identification with visual features and basic machine learning is inferior to deep learning algorithms. Based on CNNs, we showed that left and right thumbs can be differentiated with an accuracy of over 92 %. To demonstrate the usability of this model, we implemented two use cases and evaluated the concept of *thumb-aware interaction*. This includes porous interfaces and a pie menu for the non-dominant hand which participants found intuitive and useful. Moreover, we found that users did not perceive an increase in workload when using thumb-specific functions. Since our contribution is purely software-based, it can be readily deployed to every mobile device with a capacitive touchscreen.

We are publicly releasing our data set of capacitive images comprising touches of all ten fingers. While we solely used the capacitive images for finger identification, future work could improve the accuracy by including further information (*e.g.*, context, position, and sensors) into the identification process. Moreover, due to steady advances in the field of deep learning, our results could be further improved by simply training new models with the same data set in the future. Future work could also use our other models (*e.g.*, THUMB/INDEX) or train user-dependent models with our scripts for prototyping purposes.

12 DATASET AND MODELS

One outcome of the studies is a labeled dataset (*CapFingerId*) that consists of capacitive images representing touches from all ten fingers. We are publicly releasing the data set together with Python 3.6 scripts to preprocess the data as well as train and test the model described in this paper under the MIT license. We further provide the trained models, the software to run our models, and implementations of the use cases readily deployable on Android. These will enable the community to run our models on their devices. We hope that the provided models in combination with the dataset can serve as a baseline that enables other researchers to further improve the accuracy: https://github.com/interactionlab/CapFingerId.

ACKNOWLEDGEMENTS

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

REFERENCES

[1] 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. https://doi.org/10.1145/3005348

[2] 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. https://doi.org/10.1145/1952222.1952233

[3] Gary Barrett and Ryomei Omote. 2010. Projected-capacitive touch technology. *Information Display* 26, 3 (2010), 16–21.

[4] Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (2013), 1798–1828. https://doi.org/10.1109/TPAMI.2013.50

[5] 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. https://doi.org/10.1145/1731903.1731924

[6] Daniel Buschek and Florian Alt. 2015. TouchML: A Machine Learning Toolkit for Modelling Spatial Touch Targeting Behaviour. In *Proceedings of the 20th International Conference on Intelligent User Interfaces (IUI '15)*. ACM, New York, NY, USA, 110–114. https://doi.org/10.1145/2678025.2701381

[7] Daniel Buschek and Florian Alt. 2017. ProbUI: Generalising Touch Target Representations to Enable Declarative Gesture Definition for Probabilistic GUIs. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 4640–4653. https://doi.org/10.1145/3025453.3025502

[8] 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. https://doi.org/10.1109/TABLETOP.2008.4660195

[9] John Carey. 2009. Getting in touch with capacitance sensor algorithms. *Embedded* (2009).

[10] 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. https://doi.org/10.1145/2380296.2380305

[11] 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. https://doi.org/10.1145/2470654.2481422

[12] 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. https://doi.org/10.1145/2686612.2686699

[13] Li Du. 2016. An Overview of Mobile Capacitive Touch Technologies Trends. *arXiv preprint arXiv:1612.08227* (2016).

[14] 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

[15] Rachel Eardley, Anne Roudaut, Steve Gill, and Stephen J. Thompson. 2017. Understanding Grip Shifts: How Form Factors Impact Hand Movements on Mobile Phones. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 4680–4691. https://doi.org/10.1145/3025453.3025835

[16] 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 (CHI '18)*. ACM, New York, NY, USA, Article 202, 8 pages. https://doi.org/10.1145/3173574.3173776

[17] 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. https://doi.org/10.1145/2396636.2396663

[18] 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. https://doi.org/10.1145/2512349.2512795

[19] 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. https://doi.org/10.1145/3025453.3025561

[20] 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

[21] 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 (UIST '12)*. ACM, New York, NY, USA, 545–554. https://doi.org/10.1145/2380116.2380184

[22] Human Performance Research Group et al. 1988. NASA Task Load Index (TLX) v1. 0. *Paper and Pencil Package. NASA Ames Research Center, Moffett Field CA* (1988).

[23] 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. https://doi.org/10.1145/2817721.2817722

[24] 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. https://doi.org/10.1145/2984511.2984557

[25] 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. https://doi.org/10.1145/2858036.2858052

[26] 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. https://doi.org/10.1145/1095034.1095054

[27] 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). arXiv:1510.00149 http://arxiv.org/abs/1510.00149

[28] Gunnar Harboe and Elaine M. Huang. 2015. Real-World Affinity Diagramming Practices: Bridging the Paper-Digital Gap. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 95–104. https://doi.org/10.1145/2702123.2702561

[29] Sandra G Hart. 2006. NASA-task load index (NASA-TLX); 20 years later. In *Proceedings of the human factors and ergonomics society annual meeting*, Vol. 50. Sage Publications Sage CA: Los Angeles, CA, 904–908.

[30] Niels Henze, Sven Mayer, Huy Viet Le, and Valentin Schwind. 2017. Improving Software-reduced Touchscreen Latency. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '17)*. ACM, New York, NY, USA, Article 107, 8 pages. https://doi.org/10.1145/3098279.3122150

[31] 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 (CHI '16)*. ACM, New York, NY, USA, 2869–2881. https://doi.org/10.1145/2858036.2858095

[32] 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. https://doi.org/10.1145/2702123.2702518

[33] Christian Holz and Marius Knaust. 2015. Biometric Touch Sensing: Seamlessly Augmenting Each Touch with Continuous Authentication. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*. ACM, New York, NY, USA, 303–312. https://doi.org/10.1145/2807442.2807458

[34] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. 2003. A practical guide to support vector classification. (2003).

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

[36] L. Lam and S. Y. Suen. 1997. Application of majority voting to pattern recognition: an analysis of its behavior and performance. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 27, 5 (Sep 1997), 553–568. https://doi.org/10.1109/3468.618255

[37] 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. https://doi.org/10.1145/3173574.3173934

[38] Huy Viet Le, Sven Mayer, Patrick Bader, Frank Bastian, and Niels Henze. 2017. Interaction Methods and Use Cases for a Full-Touch Sensing Smartphone. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '17)*. ACM, New York, NY, USA, 2730–2737. https://doi.org/10.1145/3027063.3053196

[39] 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. https://doi.org/10.1145/3098279.3122143

[40] 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. https://doi.org/10.1145/3173574.3173605

[41] Huy Viet Le, Sven Mayer, and Niels Henze. 2018. InfiniTouch: Finger-Aware Interaction on Fully Touch Sensitive Smartphones. In *Proceedings of the 31th Annual ACM Symposium on User Interface Software and Technology (UIST '18)*. ACM, New York, NY, USA, 14. https://doi.org/10.1145/3242587.3242605

[42] Huy Viet Le, Sven Mayer, Thomas Kosch, and Niels Henze. 2018. Demonstrating PalmTouch: The Palm as An Additional Input Modality on Commodity Smartphones. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct (MobileHCI '18)*. ACM, New York, NY, USA, 4. https://doi.org/10.1145/3236112.3236163

[43] 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. https://doi.org/10.1145/2851581.2892462

[44] Huy Viet Le, Valentin Schwind, Philipp Göttlich, and Niels Henze. 2017. PredicTouch: A System to Reduce Touchscreen Latency using Neural Networks and Inertial Measurement Units. In *Proceedings of the 2017 International Conference on Interactive Surfaces and Spaces* (2017-10-17) *(ISS'17)*, Vol. 17. ACM, New York, NY, USA. https://doi.org/10.1145/3132272.3134138

[45] 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 (MUM '15)*. ACM, New York, NY, USA, 245–249. https://doi.org/10.1145/2836041.2836066

[46] Amit Mandelbaum and Daphna Weinshall. 2017. Distance-based Confidence Score for Neural Network Classifiers. *arXiv preprint arXiv:1709.09844* (2017).

[47] 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. https://doi.org/10.1145/2076354.2076358

[48] 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. https://doi.org/10.1145/3126594.3126619

[49] 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. https://doi.org/10.1145/3132272.3134126

[50] 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. https://doi.org/10.1145/3132272.3134130

[51] 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. https://doi.org/10.1145/2556288.2557148

[52] 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. https://doi.org/10.1145/2858036.2858474

[53] 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.

[54] 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. https://doi.org/10.1145/2380116.2380177

[55] Antti Oulasvirta, Anna Reichel, Wenbin Li, Yan Zhang, Myroslav Bachynskyi, Keith Vertanen, and Per Ola Kristensson. 2013. Improving Two-thumb Text Entry on Touchscreen Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2765–2774. https://doi.org/10.1145/2470654.2481383

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

[57] Karsten Seipp and Kate Devlin. 2015. One-Touch Pose Detection on Touchscreen Smartphones. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces (ITS '15)*. ACM, New York, NY, USA, 51–54. https://doi.org/10.1145/2817721.2817739

[58] 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.

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

[60] Katia Vega and Hugo Fuks. 2013. Beauty Tech Nails: Interactive Technology at Your Fingertips. In *Proceedings of the 8th International Conference on Tangible, Embedded and Embodied Interaction (TEI '14)*. ACM, New York, NY, USA, 61–64. https://doi.org/10.1145/2540930.2540961

[61] Nicolas Villar, Daniel Cletheroe, Greg Saul, Christian Holz, Tim Regan, Oscar Salandin, Misha Sra, Hui-Shyong Yeo, William Field, and Haiyan Zhang. 2018. Project Zanzibar: A Portable and Flexible Tangible Interaction Platform. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 515, 13 pages. https://doi.org/10.1145/3173574.3174089

[62] 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. https://doi.org/10.1145/985921.986040

[63] Frank Weichert, Daniel Bachmann, Bartholomäus Rudak, and Denis Fisseler. 2013. Analysis of the accuracy and robustness of the leap motion controller. *Sensors* 13, 5 (2013), 6380–6393.

[64] 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. https://doi.org/10.1145/1936652.1936665

[65] 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. https://doi.org/10.1145/2817721.2817737

[66] Ying Yin, Tom Yu Ouyang, Kurt Partridge, and Shumin Zhai. 2013. Making Touchscreen Keyboards Adaptive to Keys, Hand Postures, and Individuals: A Hierarchical Spatial Backoff Model Approach. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2775–2784. https://doi.org/10.1145/2470654.2481384

[67] 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. https://doi.org/10.1145/2858036.2858355