Estimating the Finger Orientation on Capacitive Touchscreens Using Convolutional Neural Networks

Sven Mayer, Huy Viet Le, Niels Henze
University of Stuttgart, Stuttgart, Germany
{firstname.lastname}@vis.uni-stuttgart.de

ABSTRACT
In the last years, touchscreens became the most common input device for a wide range of computers. While touchscreens are truly pervasive, commercial devices reduce the richness of touch input to two-dimensional positions on the screen. Recent work proposed interaction techniques to extend the richness of the input vocabulary using the finger orientation. Approaches for determining a finger's orientation using off-the-shelf capacitive touchscreens proposed in previous work already enable compelling use cases. However, the low estimation accuracy limits the usability and restricts the usage of finger orientation to non-precise input. With this paper, we provide a ground truth data set for capacitive touch screens recorded with a high-precision motion capture system. Using this data set, we show that a Convolutional Neural Network can outperform approaches proposed in previous work. Instead of relying on hand-crafted features, we trained the model based on the raw capacitive images. Thereby we reduce the pitch error by 9.8% and the yaw error by 45.7%.

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

Author Keywords
Finger orientation; touchscreen; mobile device; capacitive sensing.

INTRODUCTION
Over the last years, touchscreen input evolved to the main mechanism for mobile devices. Through direct touch, users can intuitively interact with the user interface (UI). UI elements can simply be selected by touching them. Recent capacitive touchscreens sense touch input by measuring a change in capacitance when a finger touches the display. These measurements are translated into a 2D point by the touchscreen controller. Since the measured capacitance is omitted afterward, touchscreen input is limited to 2D input.

Commercial devices, as well as previous research, presented a wide range of novel interaction techniques. Already in the first version of Android and iOS, they leveraged the time dimension to provide the long press. With the iOS 6s in 2015, Apple introduced *3D Touch* which adds a pressure dimension the interaction. Both methods are used to modify the touch input and alter the action. Roudaut et al. [35] presented a technique to use the roll of the finger to scroll through lists. Xiao et al. [41] and Zaliva [42] proposed using the finger orientation to increase the richness of the touch input. A larger input vocabulary enables a richer interaction and thereby enables new ways to manipulate potential applications.

ISS '17, October 17–20, 2017, Brighton, United Kingdom
©2017 Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-4691-7/17/10…$15.00
DOI: https://doi.org/10.1145/3132272.3134130

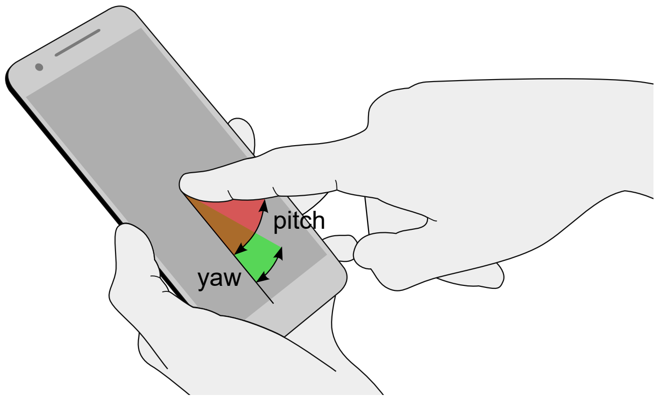


Figure 1. Finger orientation input with pitch (red) and yaw (green) input can enlarge the input space for mobile devices. Image source: [27].

The finger orientation can deliver up to three additional dimensions: pitch, roll, and yaw. Pitch is the angle between the finger, and the horizontal touch surface and yaw is the angle between the finger and the vertical axis. Pitch is presented in red and yaw in green in Figure 1. These additional dimensions are especially useful when interacting with smartwatches. For smartwatches, the finger orientation instead of the finger position could be the main input.

Previous work proposed algorithms to determine pitch and yaw to use it as an additional input modality. However, determining a finger's orientation using off-the-shelf devices and existing algorithms is still not precise. Due to the high potential of using finger orientation as an additional input, we aim to improve the algorithms proposed by previous work. In this paper, we present a range of machine learning models to estimate the fingers' pitch and yaw angle.

To summarize, this paper contributes the following:

1. A more precise estimation of the fingers' pitch and yaw orientation using the raw capacitive sensor data and a machine learning approach;

2. A labeled raw capacitive sensor data set that enables other researchers to build their own machine learning models and further improve our results.

In the following sections, we review previous work that aims to expand the touchscreen vocabulary. We then present the details of our study design and the data set we collected in our experiment. Next, we interpret and discuss the implications of the data obtained. Finally, we present how we built our new finger orientation estimation using machine learning.

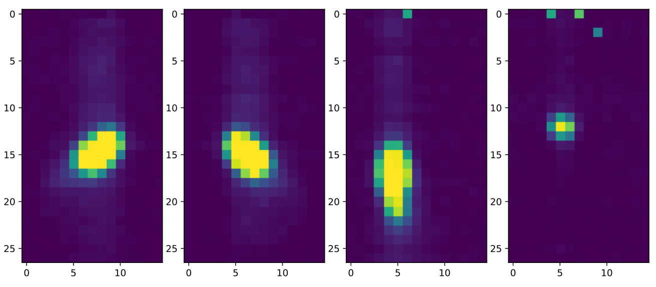
RELATED WORK

Previous work presented a wide range of novel modalities to enhance touch input on touchscreen devices. Amongst others, this includes using the finger shape [31] and size [3], part of the hand [12], pressure [33], and the shear force [11]. Further work also used additional sensors to offer additional input modalities. For example, Le et al. used a Back-of-Device (BoD) touch panel to enhance one-handed smartphone interaction [22, 25] and further presented a novel smartphone prototype that registers touch input on the whole device surface to enable use cases such as grip recognition or touch input on the whole device surface [23]. Colley and Häkkilä [6] used a Leap Motion on the bottom side of a smartphone to explore finger specific interaction on smartphones. Wilkinson et al. [38] used a wrist-worn inertial measurement unit to add expressiveness to touch-based interactions.

Evaluations of the input modalities mentioned above have shown that they are already suitable for frequent use. One input modality which was investigated in a wide range of prior work, and is still not usable in typical smartphones, is the use of the finger's pitch and yaw angle. While Mayer et al. [27] investigated the ergonomic constraints of the input modality, a number of researchers [19, 34, 41, 42] presented different approaches to estimate the finger orientation on commodity smartphones. Being able to determine the finger orientation enables use cases such as increasing touch targeting accuracy using the pitch angle [16], manipulating 3D objects, zooming, or setting values on a small touchscreen (e.g., smartwatch) by twisting the finger. Kratz et al. [19] attached a depth camera above the touchscreen to estimate the finger orientation and showed that users could consistently select and hold given target positions. While the estimation works adequately, this approach requires an additional depth camera attached to the device which mainly is a prototyping tool. Further, Mayer et al. [28] recently modded the systematic error using ground truth data to improve the initial approach. Rogers et al. [34] built a custom device based on conventional capacitive sensors to show the feasibility of estimating the finger orientation. Further approaches could use touchscreens that sense the fingers proximity to the display (e.g. Hinckley et al. [14] and Samsung's AirView) to reconstruct the 3D finger position. Other approaches could attached mirror to the front-camera display to capture the finger orientation when touching the display similar to Wong et al.'s work [40].

To enable finger orientation estimation on off-the-shelf smartphones without the need for additional sensors, researchers started to use the capacitive images provided by capacitive touchscreens. A capacitive image describes the differences in electrical capacitance between a baseline measurement when no finger is touching the screen, and a current measurement when a finger touches the screen. An example is shown in Figure 2. Amongst others, previous work used these images for biometric user identification [10, 17], hand grip recognition [5, 24], and envisioned a wide range of other use cases such as determining user's handedness, adaptive UIs based on finger position, or predicting user actions [23]. Zaliva et al. [42] used a sliding window approach combined with an artificial neural network to estimate the finger's pitch and yaw orientation. However, this was done on a table top and used a sliding window approach to calculate the pitch and yaw angle of the finger. Due to the sliding window approach, an unavoidable latency is introduced while absolute input based on finger orientation is not possible. Similar to our work, Xiao et al. [41] used Gaussian process (GP) to estimation the pitch angle based features gained from the capacitive images of an off-the-shelf smartphone. However, for yaw, they used a simple heuristic. Their evaluation revealed an accuracy that is still not suitable for daily use.

While in 2011 Henze et al. [13] showed that touch screen offsets can be models using a polynomial function, Weir et al. [37] later showed that GPs are suitable to model the offsets for two-handed interaction and even improve the touch accuracy. Recently Murray-Smith [29] proposed using CNNs to improve touchscreens' capabilities further. With this recent progress in machine learning, CNNs are now the state-of-the-art approach to train models based on images [20]. As CNNs require a large data set to be trained on [36], we conducted a study to collect a large number of capacitive images that are automatically labeled with pitch and yaw angles by a motion capture system as ground truth. Based on this data set, we train estimation models using different machine learning algorithms, starting from the recent work by Xiao et al. [41] over established machine learning algorithms such as



(a) $23°, -42°$ (b) $23°, 41°$ (c) $17°, 4°$ (d) $80°, 3°$

Figure 2. For different capacitive images form a Nexus 5 with different finger orientations. The labels represent the finger orientations pitch, yaw.

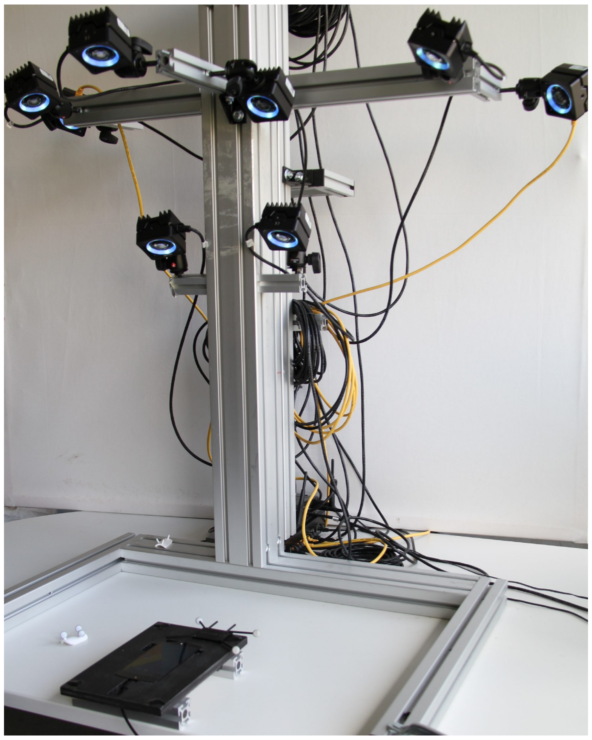


Figure 3. The study setup showing the Nexus 5 and the aluminum profiles where the cameras are firmly mounted to.

k-nearest neighbor (kNN) and Random Forest (RF) and eventually showing the best results with CNNs.

DATA COLLECTION STUDY

In this study, we collect capacitive images and respective finger orientation angles as ground truth using a motion capture system. We followed the approach by Holz and Baudisch [15] to collect ground truth data of the finger orientation. Specifically, a finger orientation consists of a pitch angle and a yaw angle. We define pitch as the angle between the finger and the horizontal touch surface. The pitch is $0°$ when the finger is parallel to the touch surface, i.e., the entire finger touches the surface. The yaw angle represents the angle between the finger and the vertical axis. Yaw is $0°$ when the finger is parallel to the long edge of the phone and increases when the finger is rotated counterclockwise.

Apparatus

The apparatus, shown in Figure 3, includes an LG Nexus 5 smartphone, eight *OptiTrack* Prime 13W motion capture cameras, and one PC to operate the *OptiTrack*. We modified the Android kernel of the LG Nexus 5 as described by Le et al. [24] to gain access to the capacitive images. Using our Android application, we recorded the 15×27 pixel image at $20\,fps$ as well as the respective 2D touch point provided by the Android SDK. While recording, the Android application instructs participants to touch on a red $2 \times 2\,cm$ crosshair as shown in Figure 4. To further record the respective 3D finger orientation in relation to the orientation of the smartphone, we attached a rigid body with 3 markers each onto the participant's index finger and the smartphone. This enables the

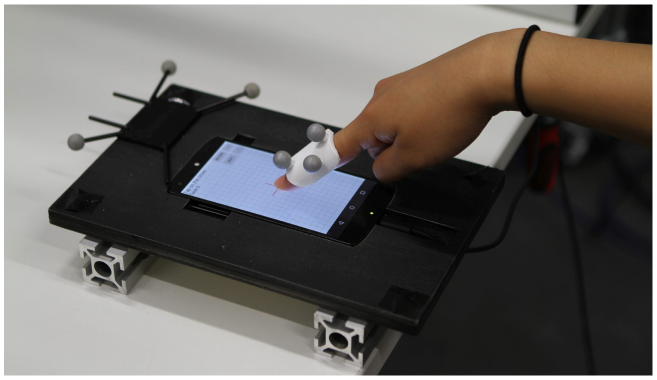


Figure 4. A close up of a participants hand while performing the study. On the participants index finger we attacked the ridget body to track the finger orientation.

motion capture system to reconstruct the pitch and yaw angle of the finger, and record them at $240\,fps$. Based on this information, the experimenter could monitor all orientations that were covered by the participants live on the PC.

Design & Task

The experiment consists out of two phases, the *tapping phase* in which participants tapped the screen repeatedly, and the *moving phase* in which participants altered the finger tip without removing the finger from the screen. Since the resolution of the capacitive image is low, we hypothesized that the blob representing the finger could look different depending on whether the touch is performed in the center of a pixel, or on the pixel borders. Thus, touches in each phase were performed on a *pixel center* and on *pixel borders*, which was represented by the red cross-hair. In total, this results in 2×2 tasks which are counterbalanced using a Latin square.

Participants

We recruited 35 participants (7 female) through our university's mailing list. Due to due to technical issues we excluded two participants. For the remaining 33 participant (7 female) the age ranged from 20 to 33 years ($M = 22.9$, $SD = 3.4$). All of them had either no visual impairment or corrected to normal vision. None of the participants had any locomotor disabilities. Further, all participants were right handed. Only participants with short fingernails were invited to participate, as this was stated to be an issue by Xiao et al. [41]. Therefore participants were able to cover the full pitch range from $0°$ and $90°$.

Procedure

After signing the consent form and filling out a demographic questionnaire, we attached the reflective rigid body markers to the participant's index finger. Participants were then instructed to touch the display with their index finger, while slowly altering the pitch and yaw angle of the finger. In the *moving phase*, they were instructed not to remove the finger while altering the orientation. In the *tapping phase*, they were instructed to altering the orientation then touch the screen lift off the finger and repeat the procedure. This was done until

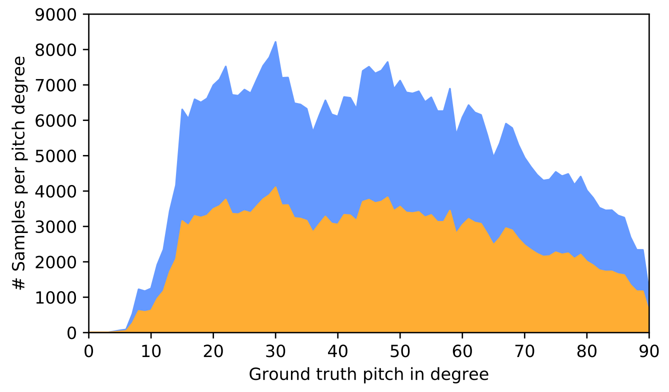


Figure 5. The blue counts are representing the distribution of pitch samples which we used for the modeling. The yellow area represents the distribution of pitch samples we recorded in our study. The are in between in obtained by flipping the yaw data.

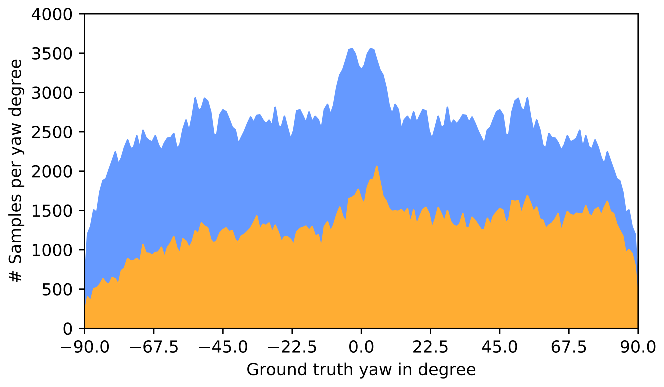


Figure 6. The blue counts are representing the distribution of yaw samples which we used for the modeling. The yellow area represents the distribution of yaw samples we recorded in our study. The area in between in obtained by flipping the yaw data.

the experimenter confirms that all angles between $0°$ and $90°$ for pitch and $-90°$ to $90°$ degrees for yaw was covered. A degree counts as covered when at least 20 capacitive images for that degree were recorded. Using the live monitor application, the experimenter instructed the participants to cover all angles and ensured that the recordings were complete.

MODELING

In a pre-processing step, we mapped the capacitive data record on the phone with the OptiTack recorded on the PC. As the capacitive images are recorded at $20\,fps$, and the OptiTack samples are recorded at $240\,fps$, we used the closest OptiTack sample for each capacitive image. This resulted in an average offset of $25\,\mu sec$ ($SD = 162\,\mu sec$).

As the first step, we removed all samples from the tapping condition. The sampling rate $20\,fps$ did often only capture the finger while moving but not the finger fully touching the touchscreen. As the second step, we followed Xiao et al. [41] and filtered noise below $3\,pF$ (picofarad). The sample distribution is shown in Figure 6 for yaw and the pitch distribution in Figure 5. As Mayer et al. [27] found that yaw input beyond $-33.75°$ falls into a non-comfort zone, we recorded fewer samples towards $yaw = -90°$. To compensate for this effect, we added vertically flipped versions of all initial capacitive images to our data set to balance the yaw samples. We then performed a blob detection on the capacitive images; the biggest blob was 15×22. We cropped all blobs and pasted the blobs into the upper left corner of an empty 15×22 image (referred to as *blob image*). The blob detection omitted all blobs that were not greater than one pixel of the image ($4.1 \ \times 4.1\,mm$) as these can be considered as the noise of the capacitive touchscreen. We used the pixel values of the blob images as input features for the model. In total, our data set consists of $457,268$ blob images.

For all models, we derived training and test sets using an 80%:20% split by participants respectively. As we had 33 participants in total, we split them into the train and test set. The first 26 participants were used for training, and remaining 7 participants were used for testing. For all models, we used an optimizer function to reduce the root mean squared error (RMSE).

Feature-Based Approach

The recent feature-based approach by Xiao et al. [41] uses 42 features extracted from the capacitive image to estimate the pitch and yaw angle. For the pitch angle, they used a GP regression. We reimplemented these 42 features. However, when feeding them to the GP, we hit the limits of GP due to a training time of $\mathcal{O}(n^3)$ and memory scaling of $\mathcal{O}(n^2)$ [21]. Since Xiao et al. recorded 1,224 samples to train their model, a GP regression worked for their in comparatively small data set to estimate pitch. However, with our data set, which is 373 times larger, training GPs on the full data set is not feasible anymore.

To train a GP we used a subset of our data set. To not vary the number of reference points, we used only samples which have the properties of the original implementation, pitch: $0°$ to $90°$ in $15°$ steps and yaw: $-60°$ to $60°$ in $15°$ steps. Since no $pitch = 0°$ samples have been recorded in our study the following comparison lacks the $0°$ validity. Further, we used $\pm 1°$ for pitch and yaw to create a data set to implement the approach presented by Xiao et al. [41]. This resulted in a $4,977$ samples large data set. We divided the data set in a train- and a test- data set. For each reference point, 75% are used for training and 25% for testing. This ensured that each original reference point was trained and tested, resulting in $3,711$ training samples and $1,266$ test samples. As Xiao et al. [41] did not report how many trainings samples they recorded, we can not ensure the same size. However, they reported that the test set contained $1,224$ samples, which is roughly the amount we use for testing. Due to the proprietary implementation[1], Xiao et al. [41] did not report which kernel or parameters they used. Thus we used the trial-and-error method [7] combined with a grid search to find parameters for the GP reimplementation of Xiao et al. [41]. To train our reimplemented GP, we used scikit-learn[2].

[1] `qeexo.com/moreproducts` - last accessed 08-26-2017
[2] `scikit-learn.org/stable/modules/gaussian_process.html`

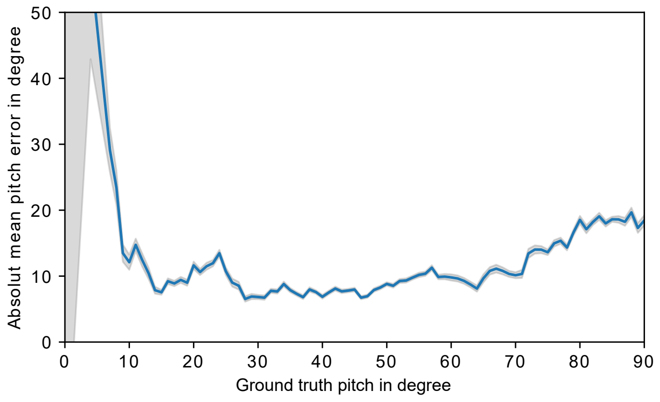


Figure 7. The remaining pitch error when using out CNN + L2 model. The gray area shows the 95% CI.

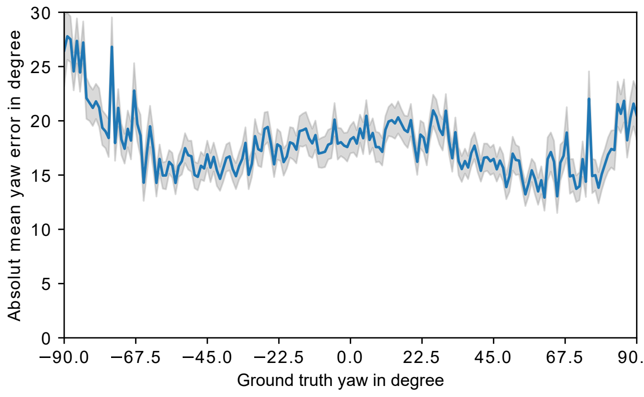


Figure 8. The remaining yaw error when using out CNN + L2 model. The gray area shows the 95% CI.

To make use of the rich dataset we collected, we additionally decided to use pseudo-implementation of the approach by Xiao et al. [41]. We use a simple k-nearest neighbor (kNN) approach as well as a Deep Neural Network (DNN) approach. For yaw Xiao et al. used the ellipsoid of the blob with a $90°$ correction when the pitch is larger than $50°$.

Pitch Estimation Using Features

For the *GP reimplementation* of Xiao et al. [41] using their 42 features we found that a RationalQuadratic[3] kernel performed best with $Alpha = .01$ and $LengthScale = 100$.

For the *pseudo implementation* we replaced the GP with a kNN, and a DNN. (1) For the kNN estimation approach using the features by Xiao et al. we achieved the best results using $k = 129$. We used a change RMSE smaller than $\epsilon = .001$ as a stopping criteria. (2) For the DNN we used a two ReLu [30] layer structure with 100 and 50 neurons respectively. To train our model we used an Adagrad Optimizer [8] with an exponential decay learning rate ($LearningRate = .01$ and $DecayRate = .2$). We initialized the weights using the Xavier initialization scheme [9] while the biases were initialized with .01.

Yaw Estimation Using Features

For the *GP reimplementation* we use the S_1 ellipsoid feature with the heuristic described by Xiao et al. [41] to estimate yaw.

The simple heuristic is based on the estimated pitch however we also want to take full advantage of the large data set. Thus, for the *pseudo implementation* we used the simple heuristic using S_1 ellipsoid feature as well a kNN and a RF model for the yaw estimation. We performed a grid search for kNN and RF. We again used an $\epsilon = .001$ as an early stopping in RMSE change. For kNN, the best results were achieved with a $k = 109$ while $Estimators = 79$ performed the best for RF .

Representation Learning Approaches

We propose a new way to determine the pitch and yaw of the touching finger. This method uses DNNs with the raw capacitive blob values as input also known as *representation learning* [1]. Thus we first applied a blob detection to identify the touching finger and then directly feed the 15×22 sized blob into the DNN to estimate the pitch and the yaw of the finger.

Beyond the feature-based baseline, we implemented two baselines that use the raw pixels of the blob to estimate pitch and yaw. Therefore, we trained kNN and RF models to estimate pitch and yaw independent from each other. We used the implementations of scikit-learn[4].

We implemented a neural network for classification using *TensorFlow*[5] 1 and tested different network configurations by varying the amount of neurons and layers, activation functions, and optimizers provided by *TensorFlow*. We trained 6 different neural network structures: two DNNs one for pitch and one for yaw and one DNN and three CNNs which estimate pitch and yaw at the same time. Further, we used early stopping to prevent overfitting for all neural networks [4]. *TensorFlow* has a large amount of parameter for their functions if the parameter is not reported in the following section we used the standard parameter of *TensorFlow* version 1.2.1. We applied the trial-and-error method [7] to find the best parameters for our models.

k-nearest neighbor (kNN): We started using kNN regression as a baseline estimation for pitch and yaw interdependently. We performed a grid search to identify the best k for the two models. For pitch, we found that $k = 180$ for pitch and $k = 278$ for yaw performed best.

Random Forest (RF): As a more advanced model than kNN, we used two RFs as a second baseline to estimate pitch and yaw interdependently. We performed a grid search to identify the best i number of trees in the forest for the two models. For pitch, we found that $i = 85$ for pitch and $i = 17$ for yaw performed best.

[3] scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.RationalQuadratic.html

[4] scikit-learn version v0.19.0: scikit-learn.org

[5] tensorflow version v1.2.1: tensorflow.org

	Pitch			Yaw			Overall
Method	**RMSE**	**MAE**	**SD**	**RMSE**	**MAE**	**SD**	**RMSE**
GP reimplementation of Xiao et al. [41]*	14.74	11.78	14.38	56.58	40.51	39.51	–
pseudo implementation of Xiao et al. [41]**	14.19	11.58	8.21	44.53	33.39	29.46	–
*k*NN	13.96	11.25	8.27	33.07	23.06	23.7	–
RF	12.99	10.24	7.99	28.55	20.89	19.46	–
DNN	13.05	10.25	8.07	27.10	19.53	18.79	–
Combined DNN	13.44	10.71	8.13	26.98	19.51	18.4	29.74
CNN	12.80	10.03	7.96	24.5	17.6	17.04	27.43
CNN + L2	12.8	10.09	7.88	24.19	17.62	16.58	27.16
CNN + L2 + BatchNorm	12.75	9.99	7.92	24.48	18.33	16.24	27.59

Table 1. The best results for all tested estimation models. Errors are reported in angular degree error. *) These results have been achieved with a small subset of the original data set ($1.4\,\%$). **) For the reported values we used a DNN instated of a GP regression for the pitch estimation as the data set was to big for a GP.

Deep Neural Network (DNN): We used two DNNs, one to estimate pitch and one for yaw. We achieved the best results using a 3-layer DNN. We used a ReLu-ReLu-Sigmoid structure, with 500, 300, 200 neurons respectively. To train our model, we used an AdaGrad Optimizer[8] with a exponential decay learning rate ($\mathit{LearningRate} = .01$ and $\mathit{DecayRate} = .2$). We initialized our weights using Xavier initialization scheme [9] while the biases were initialized with .01.

Combined DNN: For a DNN with 2 output neurons to estimate pitch and yaw at the same time, we achieved the best results using a 3-layer DNN. As layers, we used a ReLu-Relu-Sigmoid structure, with 1200, 800, and 400 neurons respectively. We initialized the weights using the Xavier initialization scheme [9] while the biases were initialized with .01. As optimizer we used a Adagrad Optimizer [8] combined with a exponential decay learning rate ($\mathit{LearningRate} = .01$ and $\mathit{DecayRate} = .2$).

Convolutional Neural Network (CNN): Next, we used a CNN with 2 output neurons to also estimate pitch and yaw at the same time. As CNNs are designed for image-like data, this was the next obvious step for us. We used 3 convolution layer each with 2×2 max-pooling and a ReLu activation function. All convolutional layers have a filter size of 7×7 and 32, 72 and 160 filter banks respectively for the three layer, followed by 2 fully connected layers (FCLs). The first FCL uses a softplus[6] activation function. We used 2000 output neurons for the first FCL and initialized the weights using Xavier initialization scheme [9] while the biases were initialized with .01. As optimizer we used a Momentum Optimizer [32] using a momentum of .9 combined with a exponential decay learning rate ($\mathit{LearningRate} = .02$ and $\mathit{DecayRate} = .1$).

CNN + L2: To improve our first CNN, we applied L2 Regularization [2]. An L2 Regularization of .015 for the two FCLs performed best with an extra change in the network structure. We changed the filter size from 7×7 of all convolutional layers in cmparcion to the previous model to 6×6 as is yield better accuracy.

[6]The softplus function is defined as $\mathit{softplus}(x) = \mathit{log}(1 + \mathit{exp}(x))$

CNN + L2 + BatchNorm: In the last model we added batch normalization [18] to CNN model with L2 Regularization. We used the same structure as the CNN with L2 Regularization model with enabled scale as well as an optimizer, exponential decay learning rate, and L2 Regularization.

RESULTS

All results of our 2 baseline approaches (*k*NN and RF), 5 Neural Network (NN) approaches as well as the results of the best estimation for *reimplementation* and *pseudo implementation* using the features proposed by Xiao et al. [41] are presented in Table 1.

Feature-Based Approaches

We used a subset of our data set to train and test the *reimplementation* due to the limitations of GPs and the full data sat for the *pseudo implementation*.

Reimplementation

For the *reimplementation* we achieved a RMSE of $14.74°$ ($\mathit{MAE} = 11.78°$, $\mathit{SD} = 14.38°$) for pitch usning a GP and a RMSE of $56.58°$ ($\mathit{MAE} = 40.51°$, $\mathit{SD} = 39.51°$) for yaw using the simple heuristic.

Pseudo implementation

For the *pseudo implementation* we achieved a RMSE of $16.14°$ ($\mathit{MAE} = 13.2°$, $\mathit{SD} = 9.28°$) using *k*NN. The DNN estimator using features, which replaced the original GP, achieved a RMSE of $14.19°$. Thus, the DNN using the features by Xiao et al. [41] outperforms the *k*NN with features by 13.7%.

The simple heuristic to estimate the yaw of the finger proposed by Xiao et al. [41] achieved a RMSE of $44.53°$. Further, we achieved a RMSE of $31.77°$ ($\mathit{MAE} = 22.96°$, $\mathit{SD} = 21.96°$) when we use a *k*NN as an estimator. And a RF model achieved a RMSE of $31.75°$ ($\mathit{MAE} = 22.95°$, $\mathit{SD} = 21.94°$).

Thus our baseline comparisons using *k*NN and RF both performed better than the simple heuristic by 28.7 %.

Representation Learning Approaches

Next, we again used *k*NN and RF as a baseline estimator. However, now we are using the raw blob values. For the *k*NN baseline, we achieved a remaining RMSE of 13.96° for pitch and 33.07° for yaw. For the Random Forest (RF) baseline, we achieved a remaining RMSE of 12.99° for pitch and 28.5° for yaw. The RF using the raw blob outperforms the simple heuristic using features by 8.5 % for pitch and by 36.% for yaw.

With our first with two separate DNNs, one for pitch and one for yaw, we achieved a RMSE of 13.05° for pitch and 27.10° for yaw. We further achieved similar results when using a DNN to predict pitch and yaw at the same time this resulted in an overall RMSE of 29.74°.

Then we used three different types of CNN. A simple CNN outperformed the DNN by 7.8 %. We further were able to reduce the RMSE to 27.16° when using a CNN with L2 Regularization. However, when applying batch normalization to the previous model, the overall result dropped to a RMSE of 27.59°.

Thus the estimator with CNN and L2 Regularization performed best with an overall RMSE error of 27.16°. The error distribution for pitch is shown in Figure 7 and for yaw in Figure 8.

DISCUSSION

In this work, we collected a data set automatically labeled by a motion capture system in the context of a study. In total, we used 457, 268 labeled samples to train our models. However, Figures 5 and 6 indicate an unequal distribution for pitch and yaw samples. Results by Mayer et al. [27] indicate that performing low pitch can be hard even in the yaw range observed in this paper. Further, they stated that performing yaw angles outside of the range $-33.75°$ to 101.25° is significantly harder for right-handed people than performing yaw angles within the range. Since we had 31 right-handed participants, this explains the the unequal sample distribution for the yaw samples.

Using our labeled data set, we evaluated the feature-based approach both the *GP reimplementation* as well as the *pseudo implementation*, and further presented multiple models including two baseline approaches (*k*-nearest neighbor (*k*NN) and Random Forest (RF)), and five different Neural Networks (NNs). In contrast to Xiao et al. [41], we used the raw capacitive image instead of feature engineering. Even the two baseline approaches using *representation learning* yield a lower estimation error than the two feature-based approach implementations.

In contrast to all other models, we trained the *GP reimplementation* with a subset of the data set which makes a real comparison hard. However, the RMSE for the GP pitch estimator is in the same range as the other models. On the other hand, the *SD* is 175% larger than the second worst pitch *SD*. Further, the simple heuristic for yaw performed worth for the GP reimplementation throughout all other yaw estimations. Additionally, the *SD* is the highest which is 134% larger than the second worst yaw *SD*.

A comparison of our results with the *pseudo implementation* of Xiao et al. [41] revealed that the *pseudo implementation* of the feature-based approach performed worse by 16.2% for pitch and 19.7% for yaw. Since our data set consists out of 457, 268 labeled samples, we have a large variance compared to their data set which consists of only 1,224 test samples. Further, they trained and evaluated their model in 15° steps while our model was trained and evaluated on a floating point level of precision. As shown in Figures 5 and 6, we cover the full pitch (from 0° to 90° degrees) and yaw range (from $-90°$ to 90°) in 1° steps.

Both our baselines (*k*NN and RF) using the raw capacitive image outperformed our implementations of the feature-based approach which are using the features proposed by Xiao et al. [41]. Using five different NNs, we showed that we could further improve the estimation accuracy. We started with two separate DNNs to predict pitch and yaw. We achieved similar estimation results using one combined DNN. Eventually, we used Convolutional Neural Networks (CNNs) to further improve the finger orientation estimation accuracy of the combined DNN by 7.8% in RMSE. Overall, we reduced the pitch RMSE by 9.8% and the yaw RMSE by 45.7% in comparison to the best feature-based approach.

While this is a step towards a precise estimation of the finger orientation, there is still a remaining error in both pitch and yaw which could result in jitter. This could limit the usability and restrict the usage of finger orientation to non-precise input. One reason includes the limitation of the touch sensor. With a pixel size of $4.1 \times 4.1\,mm$, the capacitive image still has a low-resolution which restricts the performance of the estimation. While we removed the majority of the touchscreen noise, the remaining noise still affects the estimation precision negatively. This could be improved by using a more precise high-resolution touch sensor. Further, Williamson [39] showed that increasing the sensing range above the display surface enables, for instance, to detect if two fingers belong to the same hand and Hinckley et al. [14] used an increasing the sensing range to sense a finger before the accentual touch and thereby enables new interaction techniques. Both enables detection of the whole finger without actually touching the display which can be used to model the finger shape and thus also orientation. This technology is already available in commercial smartphones, such as the Samsung Galaxy S4 which has the Air View feature. Better sensing range was well as a higher resolution could improve the capabilities to detect the finger orientation.

One limitation of our current model is that it is only trained with samples where the whole finger was captured by the touch sensor. Thus, we assume a drop in accuracy when touching close to the screen edges where only part of the finger is visible. This should be investigated in further developments. Our data set could be used to train models which take edge inputs into account by cropping the images and thereby simulating edge inputs.

THE FINGER ORIENTATION DATA SET AND MODEL

The data set collected in this paper is freely available under a GPL license[7] and available on GitHub[8]. The data set contains the full capacitive image as well as the labels for pitch and for yaw which we automatically labeled by a motion capture system in the context of our study. Further, we provide the scripts for prepossessing, train and testing on GitHub. The prepossessing scripts include the blob detections using scikit-image's implementation[9] of *find contours* by Lorensen and Cline [26]. Training and testing scripts of the method using a CNN with L2 Regularization are also published. Finally, the model which performs best is released together with data set and code. The model can directly be deployed using TensorFlow Mobile[10] directly to mobile devices (Android and iOS) and prototyping platforms such as the Raspberry Pi. The data is published in CSV files, and the code is written in Python 3.6 using *TensorFlow* version 1.1.0.

CONCLUSION

In this paper, we collected a data set of capacitive images which represents the change in capacitance caused by fingers touching the display in different finger orientations. These images are labeled with the pitch and yaw angles of the finger through a high-precision motion capture system. The feature-based approach by Xiao et al. [41] is based on a Gaussian process (GP) and a simple heuristic to estimate the finger orientation using a set of engineered features. In contrast, our model uses the representation approach using a Convolutional Neural Network (CNN) trained with raw capacitive images. We showed that this reduces the estimation error for pitch by 9.8% and yaw by 45.7% in RMSE when comparing to our *pseudo implementation* of the Xiao et al. [41] approach using their features. Besides the data set, one outcome of this work is an improved finger orientation estimation model that can be readily deployed to Android and iOS devices. We are publicly releasing the model for others to deploy them on their mobile devices, as well as the data set for further usage.

Using the latest machine learning algorithms and further additional methods, we showed a noticeable improvement over previously presented approaches. The next steps to improve finger orientation estimation on mobile devices would include using a touchscreen that provides a higher capacitive image sampling resolution. Moreover, using touchscreens with sensing capabilities above the display would enable a reconstruction of the full finger and provide more vital information to estimate the finger orientation.

ACKNOWLEDGMENTS

This work is financially supported by the German Research Foundation (DFG) within Cluster of Excellence in Simulation Technology (EXC 310/2) at the University of Stuttgart and the MWK Baden-Württemberg within the Juniorprofessuren-Programm.

[7]gnu.org/licenses/gpl-3.0.en.html
[8]github.com/interactionlab/Capacitive-Finger-Orientation-Estimation
[9]scikit-image.org/docs/dev/api/skimage.measure.html#skimage.measure.find_contours
[10]tensorflow.org/mobile/

REFERENCES

1. Bengio, Y., Courville, A., and Vincent, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*, 8 (Aug 2013), 1798–1828.

2. Bilgic, B., Chatnuntawech, I., Fan, A. P., Setsompop, K., Cauley, S. F., Wald, L. L., and Adalsteinsson, E. Fast image reconstruction with l2-regularization. *Journal of Magnetic Resonance Imaging 40*, 1 (2014), 181–191.

3. Boring, S., Ledo, D., Chen, X. A., Marquardt, N., Tang, A., and Greenberg, S. The fat thumb: Using the thumb's contact size for single-handed mobile interaction. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services*, MobileHCI '12, ACM (New York, NY, USA, 2012), 39–48.

4. Caruana, R., Lawrence, S., and Giles, C. L. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems* (2001), 402–408.

5. Chang, W., Kim, K. E., Lee, H., Cho, J. K., Soh, B. S., Shim, J. H., Yang, G., Cho, S.-J., and Park, J. Recognition of grip-patterns by using capacitive touch sensors. In *IEEE International Symposium on Industrial Electronics*, vol. 4, IEEE (2006), 2936–2941.

6. Colley, A., and Häkkilä, J. 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, 2014), 539–548.

7. Coulibaly, P., Anctil, F., and Bobe, B. Daily reservoir inflow forecasting using artificial neural networks with stopped training approach. *Journal of Hydrology 230*, 3 (2000), 244 – 257.

8. Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research 12* (July 2011), 2121–2159.

9. Glorot, X., and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *In Proceedings of the International Conference on Artificial Intelligence and Statistics. Society for Artificial Intelligence and Statistics*, vol. 9 of *AISTATS'10*, JMLR.org (2010), 249–256.

10. Guo, A., Xiao, R., and Harrison, C. 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, 2015), 59–62.

11. Harrison, C., and Hudson, S. Using shear as a supplemental two-dimensional input channel for rich touchscreen interaction. In *Proceedings of the SIGCHI*

Conference on Human Factors in Computing Systems, CHI '12, ACM (New York, NY, USA, 2012), 3149–3152.

12. Harrison, C., Schwarz, J., and Hudson, S. E. Tapsense: Enhancing finger interaction on touch surfaces. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, ACM (New York, NY, USA, 2011), 627–636.
13. Henze, N., Rukzio, E., and Boll, S. 100,000,000 taps: Analysis and improvement of touch performance in the large. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, ACM (New York, NY, USA, 2011), 133–142.
14. Hinckley, K., Heo, S., Pahud, M., Holz, C., Benko, H., Sellen, A., Banks, R., O'Hara, K., Smyth, G., and Buxton, W. 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, 2016), 2869–2881.
15. Holz, C., and Baudisch, P. The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, ACM (New York, NY, USA, 2010), 581–590.
16. Holz, C., and Baudisch, P. Understanding touch. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, ACM (New York, NY, USA, 2011), 2501–2510.
17. Holz, C., Buthpitiya, S., and Knaust, M. 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, 2015), 3011–3014.
18. Ioffe, S., and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR abs/1502.03167* (2015).
19. Kratz, S., Chiu, P., and Back, M. Pointpose: Finger pose estimation for touch input on mobile devices using a depth sensor. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '13, ACM (New York, NY, USA, 2013), 223–230.
20. Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, Curran Associates, Inc. (Lake Tahoe, NV, USA, Dec. 2012), 1097–1105.
21. Lawrence, N., Seeger, M., and Herbrich, R. Fast sparse gaussian process methods: The informative vector machine. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, MIT Press (Cambridge, MA, USA, 2002), 625–632.
22. Le, H. V., Bader, P., Kosch, T., and Henze, N. 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, 2016), 27–37.
23. Le, H. V., Mayer, S., Bader, P., Bastian, F., and Henze, N. 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, 2017), 2730–2737.
24. Le, H. V., Mayer, S., Bader, P., and Henze, N. 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 Adjunct*, MobileHCI '17, ACM (New York, NY, USA, 2017).
25. Le, H. V., Mayer, S., Wolf, K., and Henze, N. 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, 2016), 2576–2584.
26. Lorensen, W. E., and Cline, H. E. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '87, ACM (New York, NY, USA, 1987), 163–169.
27. Mayer, S., Gad, P., Wolf, K., Wozniak, P. W., and Henze, N. Understanding the ergonomic constraints in designing for touch surfaces. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services* (Vienna, 2017).
28. Mayer, S., Mayer, M., and Henze, N. Feasibility analysis of detecting the finger orientation with depth camera. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct*, MobileHCI'17, ACM (New York, NY, USA, 2017), 8.
29. Murray-Smith, R. Stratified, computational interaction via machine learning. In *Eighteenth Yale Workshop on Adaptive and Learning Systems* (New Haven, CT, USA, June 2017), 95–101.
30. Nair, V., and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning*, ICML'10, Omnipress (2010), 807–814.
31. Oakley, I., Lindahl, C., Le, K., Lee, D., and Islam, M. R. The flat finger: Exploring area touches on smartwatches. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, ACM (New York, NY, USA, 2016), 4238–4249.

32. Qian, N. On the momentum term in gradient descent learning algorithms. *Neural Networks 12*, 1 (1999), 145 – 151.

33. Ramos, G., Boulos, M., and Balakrishnan, R. Pressure widgets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, ACM (New York, NY, USA, 2004), 487–494.

34. Rogers, S., Williamson, J., Stewart, C., and Murray-Smith, R. Anglepose: Robust, precise capacitive touch tracking via 3d orientation estimation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, ACM (New York, NY, USA, 2011), 2575–2584.

35. Roudaut, A., Lecolinet, E., and Guiard, Y. Microrolls: Expanding touch-screen input vocabulary by distinguishing rolls vs. slides of the thumb. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, ACM (New York, NY, USA, 2009), 927–936.

36. Simard, P. Y., Steinkraus, D., and Platt, J. C. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, vol. 3 of *ICDAR '03*, IEEE Computer Society (Washington, DC, USA, August 2003), 958–962.

37. Weir, D., Rogers, S., Murray-Smith, R., and Löchtefeld, M. A user-specific machine learning approach for improving touch accuracy on mobile devices. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, ACM (New York, NY, USA, 2012), 465–476.

38. Wilkinson, G., Kharrufa, A., Hook, J., Pursglove, B., Wood, G., Haeuser, H., Hammerla, N. Y., Hodges, S., and Olivier, P. Expressy: Using a wrist-worn inertial measurement unit to add expressiveness to touch-based interactions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, ACM (New York, NY, USA, 2016), 2832–2844.

39. Williamson, J. Fingers of a hand oscillate together: Phase syncronisation of tremor in hover touch sensing. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, ACM (New York, NY, USA, 2016), 3433–3437.

40. Wong, P. C., Fu, H., and Zhu, K. 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, 2016), 10:1–10:5.

41. Xiao, R., Schwarz, J., and Harrison, C. 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, 2015), 47–50.

42. Zaliva, V. 3d finger posture detection and gesture recognition on touch surfaces. In *Control Automation Robotics & Vision (ICARCV), 2012 12th International Conference on*, IEEE (2012), 359–364.