# TrackPose: Towards Stable and User Adaptive Finger Pose Estimation on Capacitive Touchscreens

KE HE, Department of Automation, BNRist, Tsinghua University, China
CHENTAO LI, Department of Automation, BNRist, Tsinghua University, China
YONGJIE DUAN, Department of Automation, BNRist, Tsinghua University, China
JIANJIANG FENG*, Department of Automation, BNRist, Tsinghua University, China
JIE ZHOU, Department of Automation, BNRist, Tsinghua University, China

Several studies have explored the estimation of finger pose/angle to enhance the expressiveness of touchscreens. However, the accuracy of previous algorithms is limited by large estimation errors, and the sequential output angles are unstable, making it difficult to meet the demands of practical applications. We believe the defect arises from improper rotation representation, the lack of time-series modeling, and the difficulty in accommodating individual differences among users. To address these issues, we conduct in-depth study of rotation representation for the 2D pose problem by minimizing the errors between representation space and original space. A deep learning model, TrackPose, using a self-attention mechanism is proposed for time-series modeling to improve accuracy and stability of finger pose. A registration application on a mobile phone is developed to collect touchscreen images of each new user without the use of optical tracking device. The combination of the three measures mentioned above has resulted in a 33% reduction in the angle estimation error, 47% for the yaw angle especially. Additionally, the instability of sequential estimations, measured by the proposed metric $MAE_{\Delta}$, is reduced by 62%. User study further confirms the effectiveness of our proposed algorithm.

CCS Concepts: • **Human-centered computing → Touch screens**.

Additional Key Words and Phrases: finger angle, touchscreen, deep neural network

**ACM Reference Format:**
Ke He, Chentao Li, Yongjie Duan, Jianjiang Feng, and Jie Zhou. 2023. TrackPose: Towards Stable and User Adaptive Finger Pose Estimation on Capacitive Touchscreens. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 4, Article 161 (December 2023), 22 pages. https://doi.org/10.1145/3631459

## 1 INTRODUCTION

Capacitive touchscreens, widely available on smartphones, tablets, and smartwatches, are the primary input devices of many consumer electronics. Yet, the expressiveness and functionality are constrained since only the 2D locations of touch events are reported by the touch controllers. Consequently, researchers have concentrated

*Corresponding author

Authors' addresses: Ke He, he-k18@mails.tsinghua.edu.cn, Department of Automation, BNRist, Tsinghua University, Beijing, China; Chentao Li, lict23@mails.tsinghua.edu.cn, Department of Automation, BNRist, Tsinghua University, Beijing, China; Yongjie Duan, dyj17@mails.tsinghua.edu.cn, Department of Automation, BNRist, Tsinghua University, Beijing, China; Jianjiang Feng, jfeng@tsinghua.edu.cn, Department of Automation, BNRist, Tsinghua University, Beijing, China; Jie Zhou, jzhou@tsinghua.edu.cn, Department of Automation, BNRist, Tsinghua University, Beijing, China.

2474-9567/2023/12-ART161
https://doi.org/10.1145/3631459

on enriching the input vocabulary of touchscreens. Manipulatory force, contact area, finger angles, etc. are explored to extend the capability of touchscreens.

Finger angles offer three additional dimensions: yaw, pitch, and roll, as shown in Figure 1. It is one of the most prominent methods to enlarge the input space for the touchscreens. A large body of work has presented various algorithms for determining finger angles [15, 19, 32, 36, 37]. Several studies [1, 6] have attempted to reconstruct hand poses from capacitive data. In addition to capacitive images, some studies [8, 14] have proposed algorithms to predict finger pose via fingerprint images. Among these studies, capacitive touchscreen-based technologies are particularly attractive because of the ease of implementation on a variety of existing consumer electronic devices.

Despite recent advances, estimating finger angles based on capacitive touchscreens remains a challenging task. We first dived into the rotation representation, which has been proved very important for learning-based algorithms in computer vision, robotics, and graphics[9, 29, 40]. In the problem of finger pose estimation based on capacitive images, existing researches[19, 32, 37] use vanilla yaw and pitch angles to represent finger pose, whose values are taken from 3D Euler angles directly. We believe this representation will introduce systematic errors inherently for finger pose estimation algorithms because it discards roll angle simply. What this means for human-computer interactions is that the represented finger poses are far from the user-perceived finger poses in some cases. This creates confusion for users, degrades user experience, and cannot be solved by finger angle prediction algorithms. In this work, we proposed a 2D rotation representation to deal with the missing degree of freedom by minimizing the errors between the representation space and original space. The proposed rotation representation are superior in two aspects: 1) it is the closest 2D representation compared to the ground-truth finger pose, which means that algorithms trained with this representation will achieve higher user-perceived accuracy inherently. 2) it is non-singular, which makes it easier for neural networks to converge.

Digital pens (styluses) have similar application scenarios as finger pose. The angle error of digital pens is smaller than 0.01 degrees [18]. Compared to that, the accuracy of finger angle estimation still has a lot of room for improvement. The lowest mean average error for yaw angle of existing researches is 18°, and 10° for pitch angle [19], which is far more larger than the error of pens and cannot meet the needs of practical applications. Another problem which hinders the practical use of finger angles is that prior studies lack attention to stability. Without the time-series modeling, prior algorithms cannot take advantage of the fact that the finger angle varies continuously along the time axis. Haran [11] defines several requirements for pens including jitter, accuracy, linearity, tilt-error, and pen lag. Jitter hurts the user experience of interaction significantly [2, 3, 23].

Existing approaches do not generalize well to a new user, since different finger shapes, sizes, and pressing habits can significantly impact performance. To mitigate this gap, we present a feasible and effective method for collecting representative user data and fine-tuning the trained model with no ground truth poses needed. Quantitative results and user studies demonstrate the significant improvement compared to the state-of-the-art study.

To summarize, our contributions include:

- a 2D rotation representation, which is optimal in the sense of representation errors and is superior to be used for training learning-based algorithms.
- a time-series deep neural network with self-attention mechanism, which estimates sequential finger angles accurately and steadily.
- a user adaptive fine-tuning method, which reduces the performance gap for new users. An application installed on the mobile phone is the only requirement to finish the user-specific registration.

We believe that with the more accurate, stable, and non-singular finger pose estimation model proposed in this study, interactive applications that use finger angle as extended dimensions will be more user-friendly.

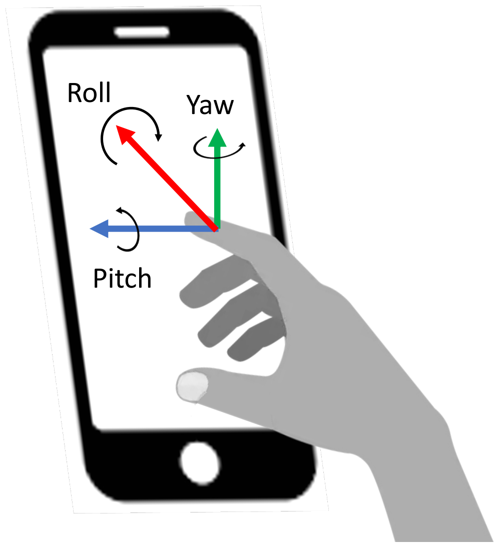


Fig. 1. Definition of finger angle: yaw (green), pitch (blue), and roll (red).

## 2 RELATED WORK

Touch controllers only extract the 2D location of the contacting finger nowadays, which is becoming increasingly insufficient as mobile applications become more complex. Various innovative modalities have been introduced to improve the touch input on touchscreen devices, including finger contact area [5, 24], manipulatory force [4, 26, 36], part of the hand [13], shear forces [12], and finger angle [34, 37]. Mayer et al. [21], and Streli and Holz [31] proposed algorithms and applications by reconstructing high-resolution contact area wth touchscreen. Le et al. [16] identified different fingers from capacitive data. Vogelsang et al. [33] presented expert interviews to describe the wide range of finger orientation input opportunities. Among these new touch inputs, finger angle has gained a lot of attention because it can offer two or three additional, continuous degrees of freedom (yaw, pitch, and roll) for interactive functions. Considering the limited information from touch sensors, researchers have proposed the use of additional sensors to provide more information. Watanabe et al. [36] used a RGB camera. Mayer et al. [20] constructed a prototype using a depth camera mounted on a tablet and reduced the systematic error using ground truth data to improve the initial approach of Kratz et al. [15]. Dang and André [7] used infrared images captured by a camera sensor inside a tabletop as input and processed the contour of the contacting area to estimate the finger angle. However, requirements for additional sensors to obtain finger posture information became the primary barrier for these algorithms to be applied in realistic applications.

Estimating finger angles on off-the-shelf smartphones without the use of auxiliary devices is an exciting area of research. Capacitive images reflect the disturbances in the projected electric field caused by finger touch [10]. It can be obtained directly from any commercial smartphones and tablets equipped with touchscreens. Limited by the low resolution of capacitive touchscreens (take for example LG Nexus 5 smartphone, 27×15 capacitors for a 137.9×69.2 mm$^2$ touchscreen), their typical usage is simply reporting the 2D location of touch events to the operating system. Wang et al. [34] used the shape of a capacitive image to calculate the yaw angle of a finger. Rogers et al. [27] further expanded the pitch angle by presenting a finger-tracking system for touch-based interaction. Roudaut et al. [28] proposed MicroRolls, which was characterized by zero tangential velocity of the skin relative to the screen surface, to enrich more gestures. Zaliva [39] demonstrated several useful features including the contact area, average intensity, centroids, and the shape asymmetry. They used these characteristics in conjunction with an artificial neural network to estimate the pitch and yaw orientations. Xiao et al. [37] extended the feature set and trained a Gaussian Process Estimator (GPR) to regress the finger angles. Their work demonstrated that combining machine learning algorithms with handcrafted features could improve estimation. Mayer et al. [19] proposed a convolutional neural network (CNN), that was trained on a large scale dataset with capacitive images and corresponding finger angles. The estimation accuracy improved further as a result of deep neural networks' powerful feature representation ability. ThumbPitch [32] used a CNN model to predict the pitch angle of thumbs when holding the phone in one hand. Recently, several studies have shown the possibility to estimate hand pose from capacitive images. Ahuja et al. [1] designed TouchPose, a multi-task network, to predict depth image, finger angle, and hand pose. Choi et al. [6] proposed an algorithm to estimate hand pose by comparing input capacitive image with a reference library. In additional to capacitive data, fingerprint images are also explored for finger pose estimation. He et al. [14] presented a multi-task neural network to estimate 3D finger pose given fingerprint images. Duan et al. [8] estimated finger angles by matching keypoints between fingerprint images and reconstructed 3D point cloud. Their methods were tested using fingerprint images captured by large optical fingerprint sensors, and the adoption in smartphones is still not practical due to low frame rate and small sensor area.

Previous studies, such as [30] and [25], have also utilized temporal dependencies in sequential capacitive data to recognize gestures. However, our study differs in the following aspects: (1) Target task. The target in this study is to estimate continuous finger angles, which is different from the gesture classification task in [30], and [25]. (2) Model architecture. The proposed TrackPose uses self-attention mechanism to implement time-series modeling instead of LSTM, which is used in [30], and [25]. (3) Design for stable estimation. We proposed a dedicated tracking loss function for instructing the model to forecast smoother angles. No similar design appears in prior researches. (4) User-adaptive fine-tuning. With the application developed in this study, pre-trained TrackPose could adapt to unseen user with a better performance. To the best of our knowledge, this was not proposed in prior studies.

While existing finger pose estimation methods have achieved significant reduction in quantitative errors, they are still not widely used in real-world applications. This is partly due to the lack of consideration of stability in algorithms' output angles for a sequence of continuous capacitive images. In scenarios where users must constantly press their fingers on the screen, such as manipulating a 3D model, unexpected vibrations of the operated objects can reduce users' willingness to use finger pose. Therefore, both accuracy and stability should be considered when selecting pose estimation algorithms for practical applications. To assess stability quantitatively, we propose a metric in addition to absolute yaw and pitch errors. We designed a specialized neural network with multihead self-attention layers to predict the most recent finger angles. Moreover, the tracking loss is designed to focus on the network's output stability along the time axis.

## 3 DATA COLLECTION

To the best of our knowledge, no prior research has released large-scale time-series capacitive touchscreen dataset. The most related one is the dataset used in [19]. However, it is not constructed in sequences. We try our best to regroup separate frames into sequences based on the available timestamp log, but the finger angles and capacitive images in many recovered sequences are very close to each other, which means participants barely move their fingers in these sessions during the data capture process. The variation ranges on average are 6°, and

4° for regrouped sequences. Such sequences contain limited information about fingers' motion profiles. Therefore, the dataset in [19] is not suitable for training and evaluating time-series models for this study. Another important reason for gathering a new dataset is that we contribute a user adaptive fine-tuning method which needs participants' representative data while operating on the smartphone. Therefore, instead of building upon the dataset of [19], we collect a new time-series dataset to verify the contributions of time-series modeling and user-specific fine-tuning. The dataset comprises 1630 sequences of capacitive images and corresponding finger angles from 12 volunteers for 72 different fingers. The total number of images is 75,152.

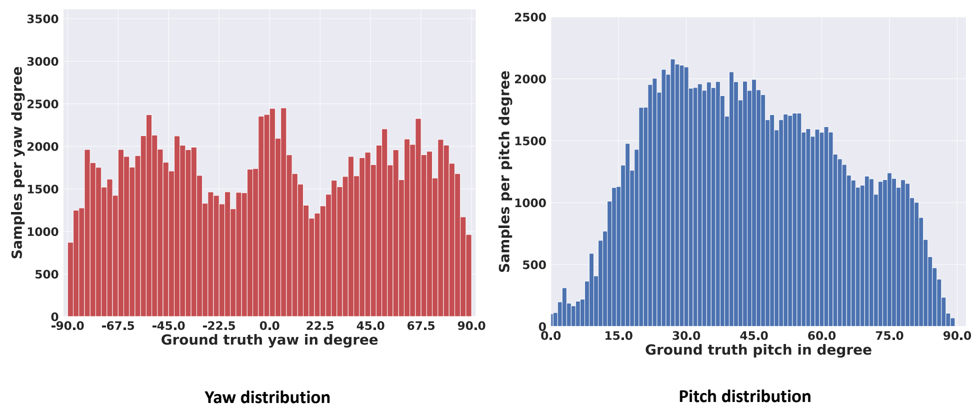


Fig. 2. Distributions of yaw and pitch angles in our dataset.

## 3.1 Apparatus

Our data acquisition system, shown in Figure 3, consists of a Realme C11 smartphone, an optical tracking system (PST-Iris from PS-Tech[1]), and several reflective markers for finger pose tracking. The touch-sensitive area of the screen is 160×75 mm$^2$. The touchscreen controller is ICNT8962 from Chipone Technology[2], which records 31× 16 lines capacitive images at 60 FPS. The input driver of the Android kernel in this phone was modified to send raw capacitive values to the operating system, which are available to us with the support of Chipone. The optical tracking system is utilized to record 3D finger angles as ground truth at 120 FPS, whose orientation tracking error is lower than 0.5°. A rigid body with four reflective optical markers is attached to the fingertips of participants. Its 3D pose relative to the markers installed on the screen is calculated as the finger pose. A data acquisition software is developed to synchronize sequential capacitive images and 3D finger angles.

## 3.2 Participants

Totally 12 volunteers were invited to participate in the data collection procedure (10 male, 2 females, aged from 19 to 28, $M = 24.22, SD = 3.74$). Each volunteer was advised to use 6 frequently used fingers, thumb, index, and middle for both the left and right hands. None of the participants had any movement impairment. Participants in this study had diverse finger length, finger width, finger shapes and showed different characteristics in capacitive images.

[1]https://www.ps-tech.com/products-pst-iris
[2]http://www.chiponeic.com/en/TT/229

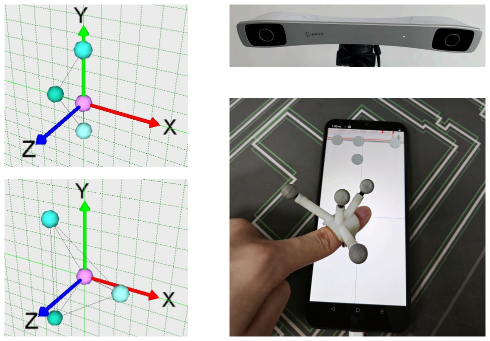


Fig. 3. The data-acquisition system. The left panel shows the tracked postures of the markers attached to the fingers and markers on the smartphone. The upper right panel shows the PST-Iris. The bottom right panel shows the Realme C11 smartphone with markers installed on the screen. Reflective rigid body is attached to the volunteer's finger near the fingertip.

### 3.3 Procedure

The volunteers were encouraged to produce various finger angles freely while pressing on any location on the touchscreen. The lifting and rotation speed was not limited. Both landscape and portrait phone layouts were used during the procedure. Volunteers could touch the screen and remove the finger at any time. The joint distribution of the yaw and pitch angles was monitored on the software interface, therefore the experimenter could instruct the volunteers to cover all finger angles. But the experimenter would not interfere with the procedure to make sure the volunteers performed the operations at their own pace. A capture session for a finger was complete when all angles between $0°$ and $90°$ for pitch and $-90°$ and $90°$ degrees for yaw had been covered and a minimum count of sequences (50) was met. Totally 6 sessions for thumb, index, middle finger of both left and right hands were conducted for each volunteer.

### 3.4 Data Filtering

The capacitive images and finger angles were cached and synchronized in sequences. In total, around 2100 sequences and 97000 images were captured. Sequences with less than 3 frames were removed. Blob detection was done following [19] by finding contours through marching cubes algorithm [17]. Contour area threshold was set to 4 pixels to include small capacitive images. Empty capacitive images with no blobs (no touch) were discarded. After filtering, 1630 sequences from 12 person for 72 fingers, which contained 75152 pairs of captured images and finger angles, were left. The mean angle for yaw was $8.9°$ ($SD = 65°$), and the mean angle for pitch was $43°$ ($SD = 21°$). The distributions of the yaw and pitch angles are shown in Figure 2.

## 4 ROTATION REPRESENTATION

The Euler angles are three angles that describe the orientation of a rigid body with respect to a fixed coordinate system. They are more intuitive than other rotation formulations. The angles consist of yaw, pitch, and roll, where pitch is the angle between the finger and the horizontal touch surface, roll is the angle around the finger's longitudinal axis, and yaw is the angle between the finger and the vertical axis, as illustrated in Figure 1.

Due to the limitations of low-resolution capacitive images, it is difficult to accurately calculate the roll angle. Previous studies [19, 34, 37, 39] estimated yaw and pitch angles only. MicroRolls [28] introduced several rolling gestures through the recognition of sequential X-Y locations, but the output was limited to a few categories and did not estimate the analog roll angle. In the case of missing roll angles, there is a question that has not been addressed in previous studies: is it correct to use vanilla yaw and pitch angles to represent finger pose? We examine this question from two perspectives. Firstly, ignoring the roll angle introduces systematic errors in the finger pose. For example, the product

$$R = R_z(\alpha)R_y(\beta)R_x(\gamma) \tag{1}$$

represents a rotation whose yaw, pitch, and roll angles are $\alpha$, $\beta$, $\gamma$, respectively. However, if we ignore the roll angle simply (which is equivalent to setting the roll angle to $0°$), the rotation matrix becomes

$$\hat{R} = R_z(\alpha)R_y(\beta), \tag{2}$$

which is different from $R$. The difference between $\hat{R}$ and $R$ becomes larger as the roll angle increases. This means that even if a perfect pose estimation model exists that can output exact vanilla yaw and pitch angles as the ground truth, the reconstructed finger pose by it may be far from the actual 3D finger pose due to the systematic errors. Secondly, Saxena et al. [29] proposed that the 3D Euler angles cause learning problems due to discontinuities. In our problem, this shortcoming is magnified by the absence of roll angle. Zhou et al. [40] suggested the use of representations in 5D and 6D spaces with good continuity. However, their conclusions do not solve our problem because, as mentioned before, any formulation containing more than two degrees of freedom is not appropriate in this context limited by the information-poor input modality.

From the perspective of maintaining the closest finger pose, we propose the optimal 2D rotation representation. Given the full 3D finger pose below:

$$R = \begin{bmatrix} c(\alpha)c(\beta) & c(\alpha)s(\beta)s(\gamma) - s(\alpha)c(\gamma) & c(\alpha)s(\beta)c(\gamma) + s(\alpha)s(\gamma) \\ s(\alpha)c(\beta) & s(\alpha)s(\beta)s(\gamma) + c(\alpha)c(\gamma) & s(\alpha)s(\beta)c(\gamma) - c(\alpha)s(\gamma) \\ -s(\beta) & c(\beta)s(\gamma) & c(\beta)c(\gamma) \end{bmatrix},$$

where 'c' is shorthand for the cos function, and 's' is shorthand for sin function. Keeping the pitch angle constant, the optimal yaw angle can be solved by

$$\min_{\bar{\alpha}\in(-90,90)} \|(R_z(\bar{\alpha})R_y(\beta)) - R\|, \tag{3}$$

where $\|...\||$ represents the squared Frobenius norm of a matrix. For simplicity, we use $f(\bar{\alpha})$ to represent the objective function $\|(R_z(\bar{\alpha})R_y(\beta)) - R\|$. By analyzing the first and second derivatives $f(\bar{\alpha})$, we get the solution as below:

$$\alpha^* = \begin{cases} \arctan(\frac{s(\alpha)c(\beta)^2+s(\alpha)c(\gamma)s(\beta)^2-c(\alpha)s(\gamma)s(\beta)}{c(\alpha)c(\beta)^2+c(\alpha)c(\gamma)s(\beta)^2+s(\alpha)s(\beta)s(\gamma)}), & \text{if } \frac{\partial^2 f}{\partial \bar{\alpha}^2} > 0. \\ 90 \times \text{sign}(\alpha), & \text{otherwise.} \end{cases} \tag{4}$$

The proposed representation has two advantages compared with the vanilla yaw and pitch angles: 1) it considers roll angle and makes the representation optimal in the sense of minimizing errors between the representation space and the original space. What this means for finger-pose based interaction is the rotation representation itself has less systematic errors inherently and is closer to user-perceived finger pose. 2) the solution has no singularities. This means that the representation is smooth and continuous even when users lifts the finger up to operate. While in this case, the vanilla yaw suffers very significantly due to the Gimbal Lock. Considering the good properties of the proposed representation, we use it as the ground truth for learning-based algorithms. It is worth emphasizing that both our model and baseline models were trained and evaluated with it for a fair comparison.

## 5 TRACKPOSE MODEL

In this section, we present a deep neural network with self-attention layers that assembles time-series features extracted from continuous capacitive images. Then, we show how a loss function can be used to help the model track finger angles smoothly.

### 5.1 Input Processing

To assemble time-series features of continuous capacitive images, $N$ frames were fed into the network as a whole for angle predictions. The hyperparameter $N$ was set to 5 in the final experiment considering the trade-off between memory footprint and model accuracy. When the available frames were less than 5, we padded the sequence to 5 frames with "static frames" at the beginning by repeating the first frame. This only happened when the finger started touching the screen (the initial 0.08 seconds at 60 FPS). Each frame in a sequence was padded into $32 \times 32$. Min-max intensity normalization was applied. For the training stage, we further performed random translation as a method of data augmentation. Translation was not performed at inference.

### 5.2 Self-attention Model Architecture

Mayer et al. [19] and Ullerich et al. [32] used a convolutional neural network (CNN) to estimate finger angles based on a single capacitive image. The information included was limited. Motivated by the observation that humans can guess the finger pose by looking at the changes of sequential capacitive images, we develop a time-series model, called TrackPose, that utilizes a CNN as the backbone network to extract features from adjacent images and self-attention layers to fuse features along the time axis.

An overview of the TrackPose architecture is shown in Figure 4. The overall network was divided into two parts: CNN backbone and attention module. In Figure 4, $[I_1, I_2, ..., I_N]$ is an example input sequence, For each frame (i.e., $I_1$, $32 \times 32 \times 1$ in size) in this sequence, the CNN backbone extracts feature $X_1$ from it. It begins with a convolutional layer which has 64 kernels sized $3 \times 3$, and $2 \times 2$ average-pooling layer. Subsequently, three residual convolutional layers are applied. The output channels are 64, 128, and 128, respectively. The kernel size is $3 \times 3$ for all layers. A fully connected (FC) layer is followed to extract a 64-dimensional feature for each frame. $\{X_1, X_2, ..., X_N\}$ are fed into the self-attention module. They are first normalized using layer normalization, and then regrouped by three multi-head self-attention layers. To encode the temporal sequential relationship, we mask the attention matrix to be a lower triangular matrix (See Figure 5). An older frame should not get any information from newer frames. Finally, a single concatenated feature describing the whole sequence is passed through a FC layer to predict the yaw and pitch for the latest frame $I_N$ (denoted as $\bar{\alpha}_N$ and $\bar{\beta}_N$).

### 5.3 Model Training

Existing studies [1, 19, 32, 34, 37, 39] focused on improving the accuracy of finger angle estimation. We argue that stability of continuous estimation is also important for practical applications. However, simple smoothing of the predictions results in hysteresis, particularly when the finger pose changes rapidly, making the operations feeling unresponsive for users. Similar to the velocity loss applied in [38], we used a temporal tracking loss function for guiding the TrackPose model to put more attention to the stability of adjacent estimations.

Given the outputs of a sequence $\bar{\alpha}_1, \bar{\alpha}_2, \ldots, \bar{\alpha}_N$ and $\bar{\beta}_1, \bar{\beta}_2, \ldots, \bar{\beta}_N$, we first compare them with the ground truth $\alpha_1, \alpha_2, \ldots, \alpha_N$ and $\beta_1, \beta_2, \ldots, \beta_N$, and the Mean Squared Error (MSE) loss function is calculated as follows:

$$L_{\text{pose}} = \frac{1}{N}\left(\sum_{k=1}^{N} L_{\text{MSE}}(\bar{\alpha}_k, \alpha_k) + \sum_{k=1}^{N} L_{\text{MSE}}(\bar{\beta}_k, \beta_k)\right), \tag{5}$$

Then, a tracking loss is applied to instruct the neural network to forecast smoother angles for subsequent frames:

$$L_{\text{tracking}} = \frac{1}{N-1}\left(\sum_{k=2}^{N} L_{\text{MSE}}(\bar{\rho}_k, \rho_k) + \sum_{k=2}^{N} L_{\text{MSE}}(\bar{\sigma}_k, \sigma_k)\right), \tag{6}$$

where $\bar{\rho}_k = \bar{\alpha}_k - \bar{\alpha}_{k-1}, \rho_k = \alpha_k - \alpha_{k-1}, \bar{\sigma}_k = \bar{\beta}_k - \bar{\beta}_{k-1}, \sigma_k = \beta_k - \beta_{k-1}$, which can be considered constraints on the increments. Figure 6 shows the motivation for designing this loss function. The tracking loss function above

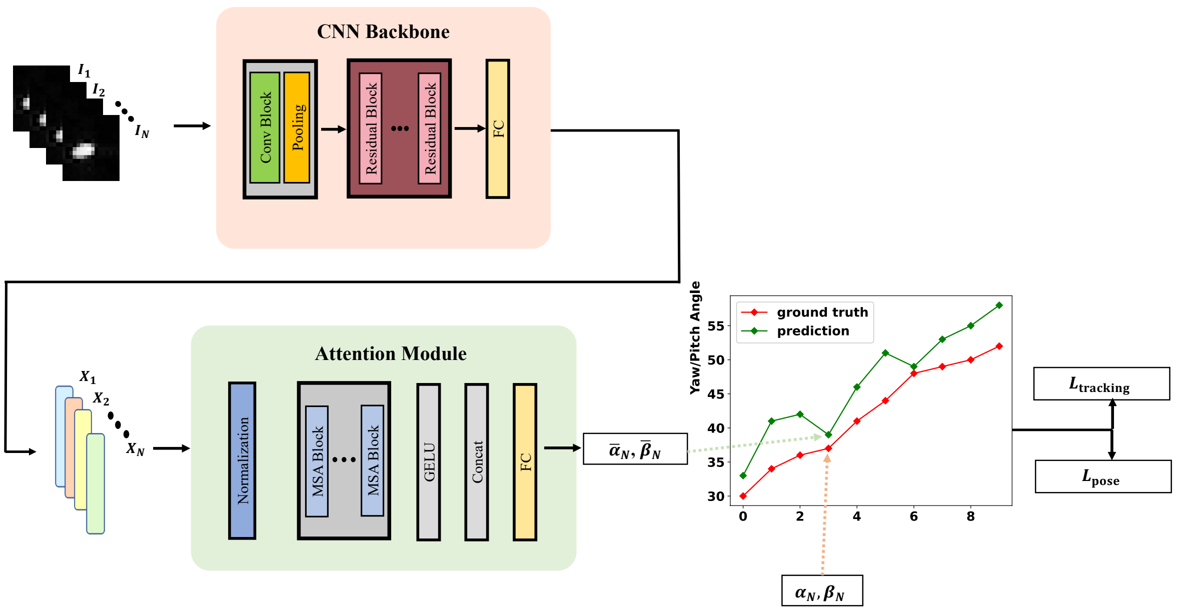


Fig. 4. Illustration of the architecture of the proposed TrackPose model for sequential finger angle estimation. Features extracted by CNN backbone are mixed by the self-attention module. Sequential predictions are supervised by both MSE loss and the proposed tracking loss.

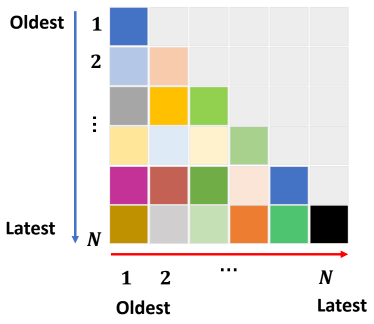


Fig. 5. Illustration of how we masked out the scaled dot-product attention matrix to represent the temporal relationship along the time axis. The gray elements in the upper-right corner indicate that an older frame should not obtain any information from newer frames. Generally, frame $k$ only accepts features from 1 to $k$.

drives the neural network to prefer to the right curve in Figure 6, which is more stable along the time axis. The overall loss function is represented as

$$L = L_{\text{pose}} + \delta L_{\text{tracking}}, \tag{7}$$

where $\delta$ is a hyperparameter, which is set to 5 in our experiment.

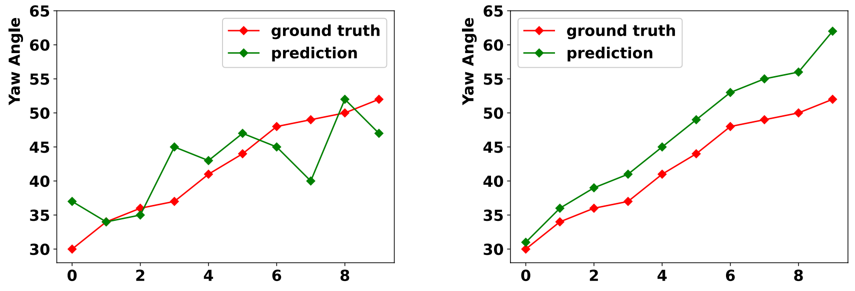


Fig. 6. The left panel shows a typical curve of the predicted yaw angles by a single-frame model, and the right is a curve of the predicted yaw angles by our approach. Even though the average estimation error is close in this example, the estimation curve on the right is preferable for interaction applications because it is smoother and the overall trend is more consistent with the ground truth.

## 6 USER ADAPTIVE FINE-TUNING

Deep learning models often exhibit a performance gap at inference. The generalization problem exacerbates this issue in the context of finger angle estimation because fingers come in different shapes and sizes, and individuals have different habits when pressing their fingers. This causes existing approaches to perform worse on new fingers.

A possible solution is to collect capacitive images along with finger angles for new users. However, this can be impractical if an auxiliary device such as a 3D pose tracking system is needed. Therefore, a registration application was developed that collected representative data of an unseen finger, including the capacitive image sequences and the corresponding finger angles. During the registration procedure, no extra devices were required except for the phone with the registration application installed. The process of registering a new finger cost less than 3 minutes on average, and improved accuracy by 17% for yaw angle compared to the general TrackPose model (see MAE metric in Table 1).

### 6.1 Apparatus

We used the same smartphone as 3.1, but the 3D pose tracking system was no longer needed. Its role was taken by the application we developed.

### 6.2 Participants

The same volunteers in Section 3.2 completed the registration. To clarify, data collected in this section would only be used in the fine-tuning stage (part of the training stage). No ground-truth finger angles could be obtained by this measure, therefore we did not use them for any quantitative evaluation. More details about how to use the dataset collected by the application will be introduced in Section 7.2.

### 6.3 Procedure

The User Interface is shown in Figure 7. Once a user clicked the "START" button, an arrow with a specific direction $D$ would be displayed on the screen. Users were then guided to lift their finger evenly from the fat posture to the steep posture (see Figure 8). They were also required to keep the yaw angle unaltered by following the arrow's direction (see Figure 7), which was easy with the visual guidance of the arrow. The arrow went from $0°$ to $180°$ at intervals of $22.5°$. Totally 9 sequences and about 200 frames were registered for a finger.

## 6.4 Data Processing

Note that our application was landscape-oriented. To be consistent with the data acquired in Section 3, the arrow's direction (see Figure 7) was converted to angles as follows:

$$\alpha_k = \begin{cases} -D, & \text{if } D \leq 90 \\ 180 - D, & \text{otherwise} \end{cases} \qquad k \in 0, ..., T-1 \tag{8}$$

With the calculated angles and collected capacitive image sequences, the general TrackPose model trained by Section 5.3 was fine-tuned to a user-specific customized TrackPose model. The technical details will be illustrated in Section 7.2.

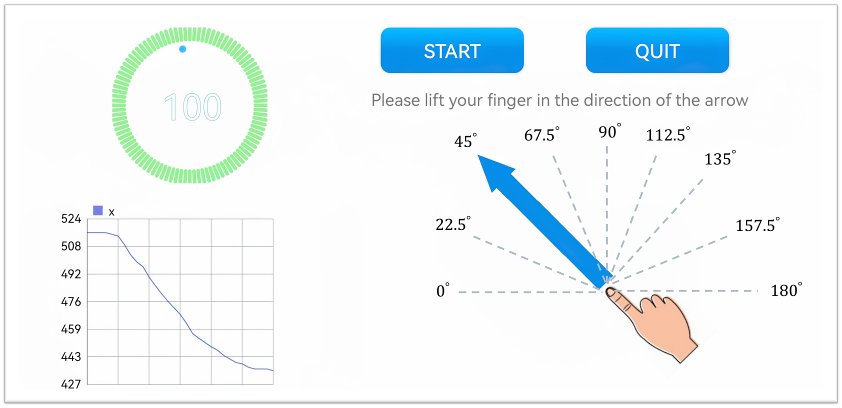


Fig. 7. Our registration Android application. Users were guided to lift their fingers along the arrow's direction in each session. Totally 9 sequences will be captured in order.

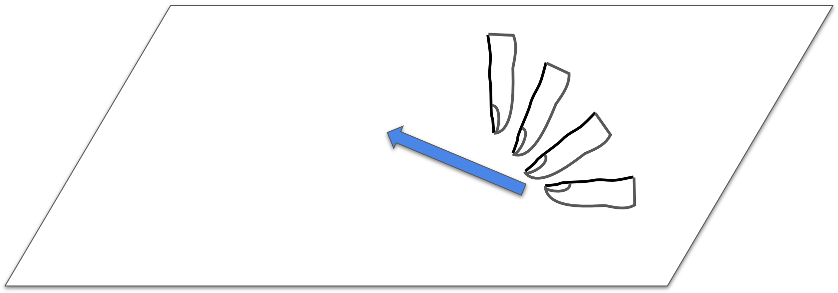


Fig. 8. Side view of the registration procedure. Users were guided to lift their fingers evenly from fat posture to steep posture.

## 7 EXPERIMENTS

In this section, we first introduce the experiment settings including dataset split, and evaluation metrics. Two proposed models are evaluated, which are the general TrackPose model without user adaptive fine-tuning and the customized TrackPose model with fine-tuning through user-specific registration. A single-frame CNN model inspired by [19] and a Gaussian Process Regression model inspired by [37] were implemented and tested on the collected dataset to serve as baselines. Additionally, performance analysis is conducted to illustrate the contributions of time-series modeling and user-specific fine-tuning.

## 7.1 Dataset Splitting

As said in Section 3, we collected dataset from 12 person. We used a 12-fold 'leave-two-person-out' cross-validation to evaluate both TrackPose and baseline methods. Through this evaluation protocal, we assured that every person would be treated as a totally new user as least in one fold. In each fold, sequences from 8 person were used for training, 2 person were used for validation, and the remaining 2 person were used for testing. Results in Table 1 were reported as the average on all folds.

## 7.2 Train General and Customized TrackPose

The proposed model was implemented with Pytorch. The general TrackPose model was optimized using the loss function illustrated in 5.3 by AdamW optimizer, whose $beta = (0.9, 0.999)$, $weight\_decay = 0.001$. The batch size was set to 256. Early stopping was used to avoid overfitting. Maximum epochs was 200. The learning rate was 0.001 initially and was scheduled by the ReduceLROnPlateau scheduler. The experiments were performed on a computer with an Intel Xeon E5 CPU and two NVIDIA GTX 3090 GPUs. It took approximately 4.5 hours to train the model.

For customized TrackPose model, let us take a specific fold as an example. For instance, person $P1, P2, \ldots, P8$ were the training dataset for the current fold, $P9, P10$ were the validation dataset, and $P11, P12$ were the testing dataset. We first loaded the weights of the general model trained on $P1, P2, \ldots, P8$. And then we used the application in Section 6 to collect user-specific data for person $P11$ and $P12$ respectively. Finally we got the customized TrackPose model for person $P11$ and $P12$ respectively by fine-tuning the general model for one epoch with a moderate learning rate 0.0005. In this process, no ground-truth finger angles acquired by 3D pose tracking system of $P11$ and $P12$ were used to train the model. Only the values we recorded by the arrows in Section 6 were fed into the neural network. Therefore, there was no data leakage issue in this process. The overall process was conceptually similar to on-device fine-tuning techniques in the area of face recognition.

## 7.3 Evaluation Metrics

Besides the Mean Absolute Error (MAE) metric, the Root Mean Squared Errors (RMSE), and the Standard Deviation (SD) used in existing studies, we propose a new metric, the MAE of increments (or speeds) for both yaw and pitch, to reveal the stability of sequential finger angle estimation. Given yaw angles of adjacent frames $\alpha_k$ and $\alpha_{k-1}$, the increment, which is denoted as $\Delta\alpha$, reveals the rotation speed (including the changing direction and magnitude), so as for the pitch angle:

$$\Delta\alpha_k = \alpha_k - \alpha_{k-1}, \tag{9}$$

$$\Delta\beta_k = \beta_k - \beta_{k-1}. \tag{10}$$

For a sequence whose ground truth yaw angles are $\alpha_1, \alpha_2, ..., \alpha_N$ and pitch angles are $\beta_1, \beta_2, ..., \beta_N$, the ground truth for incremental yaw angles and incremental pitch angles can be computed based on Equation 9 and 10. The predictions of the increments can be calculated in the same way given the sequential estimated finger angles estimated. Then, the mean average errors of the yaw and pitch increments can be evaluated as

$$MAE_\Delta = \frac{1}{N-1}\left(\sum_{k=2}^{N} |\Delta\alpha_k - \Delta\bar{\alpha}_k| + \sum_{k=2}^{N} |\Delta\beta_k - \Delta\bar{\beta}_k|\right). \tag{11}$$

As shown in Figure 6, the MAEs of the yaw angle for the two curves are similar, but the MAEs of the incremental yaw angle are 5.9° for the left and 1.2° for the right. The new metric can reveal the preference for more stable estimations in the context of continuous prediction.

### 7.4 Baselines Implementation

We re-implemented a CNN inspired by the best model in [19]. Model architecture, weights initialization, optimizer choice, and data pre-processing were kept the same as that in [19]. The same L2 regularization and batch normalization were used for best performance. For each fold, hyperparameters were initially set to the values in [19] and were tuned based on the performance on the validation dataset. Gaussian Process Regression (GPR) was re-implemented referring to the model in [37] using sklearn toolkits[3]. Due to the limit of memory footprint and training time, a smaller subset of the original training set (8.3%, 6200 images) was used to train GPR. The same dataset splitting was used for baseline models for a fair comparison.

### 7.5 Results

Results are reported as the average of all 'leave-two-person-out' sessions. The standard deviation of MAE among different folds is 2.6°. Besides the baseline methods inspired by [19] and [37], the performance of the cross-person experiment (protocol 2) reported in [1] is also listed in Table 1. Our best model is able to reduce the overall MAE by 33% compared with CNN baseline inspired by [19], 47% for the yaw angle especially. Additionally, considering the instability of sequential estimations which can be revealed by the metric $MAE_\Delta$, we reduce the error of it by 62%. MAE is also smaller than that reported in [1], though the experiments are conducted on two totally different dataset. As for the SD of errors, TouchPose [1] is much smaller than all algorithms in this study and in [19]. We ponder that the dataset distributions make the difference. As shown in Figure 2, distributions are more balanced in our collected dataset and images at high pitches account for a larger proportion, which are the difficult samples for pose estimation algorithms. ThumbPitch [32] achieves a mean error of 11.9° and is targeted at pitch angle estimation for thumbs only. Our best model can be applied to thumbs, index, and middle fingers with a smaller pitch MAE (reduced by 24%). He et al. [14] focuses on the finger pose estimation given fingerprint images. They report a MAE of 6.6° for yaw, and 7.1° for pitch. However, the input modality is quite different from that of this study. They use fingerprint images with a resolution of 500 ppi, which is much higher than that of capacitive images (typically <10 ppi). Considering the big difference in input resolution, the performance gap between this study and [14] is understandable.

### 7.6 Ablation Study

To verify the contributions of each measure in our study, we conducted ablation studies.

- **rotation representation.** As said in Section 4, the proposed rotatation representation is optimal in the sense of keeping the overall finger pose closest. The MAE of yaw and pitch could not reveal the characteristics of the representation itself. Therefore, the squared Frobenius norm of the rotation matrix error $\|(R_z(\alpha)R_y(\beta))-R\|$ was used as the evaluation metric. To demonstrate this, we trained two kinds of models: TrackPose model supervised by vanilla angles, and TrackPose model supervised by the proposed rectified angles. The rotation matrix errors were 0.10 for proposed representation and 0.17 for vanilla representation. It was also observed that models trained with the proposed representation converged significantly faster. Figure 9 illustrates the typical training processes of the CNN model inspired by [19] when using vanilla angles as ground-truth and using our proposed rectified angles as ground-truth. Here we used the CNN model to prove that the representation benefitted other learning-based models as well. This convergency experiment further verified that the representation was superior for learning-based algorithms.
- **time-series modeling.** The error distributions of the yaw and pitch are shown in Figure 10. Among these, the rightmost figure is noteworthy. In terms of the yaw angle, all three models performed worse with the the increase of the pitch angle. The phenomenon was illustrated by Xiao et al. [37]. This is because the circular appearance of capacitive images at high pitches (larger than 50°) contains little information

[3]https://scikit-learn.org

Table 1. Best results for all methods. Errors are reported as angular errors. The general TrackPose is the sequential model trained without user-specific fine-tuning (see Section 5.2). The customized TrackPose is the fine-tuned model based on the general one with representative user data collected by our developed registration application (see Section 6).

| | **Yaw** | | | | **Pitch** | | | | **Overall** | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | **RMSE** | **MAE** | **SD** | **MAE$_\Delta$** | **RMSE** | **MAE** | **SD** | **MAE$_\Delta$** | **MAE** | **MAE$_\Delta$** |
| GPR inspired by [37]* | 35.7 | 29.8 | 19.7 | 6.7 | 14.7 | 12.3 | 8.1 | 2.5 | 21.5 | 4.6 |
| CNN inspired by [19]** | 23.8 | 17.3 | 16.3 | 6.1 | 12.3 | 9.7 | 7.5 | 2.3 | 13.5 | 4.2 |
| General TrackPose | 16.1 | 10.9 | 11.8 | 2.2 | **11.5** | **8.9** | 7.3 | 1.4 | 9.9 | 1.8 |
| Customized TrackPose | **14.7** | **9.1** | **11.5** | **2.0** | 11.7 | 9.1 | **7.2** | **1.2** | **9.1** | **1.6** |
| TouchPose [1]*** | - | 11.4 | 3.2 | - | - | 9.9 | 2.5 | - | 10.6 | - |

* *re-implemented on a small subset of the training set (8.3%).*
** *re-implemented on our collected dataset using Pytorch*
*** *results reported in their original paper. '-' means that metric is not reported.*

for inferring the yaw. The difficulty of accurate estimation about the yaw angle at high pitches can be understood through an intuitive example. As shown in Figure 11, yaw angle estimation given a single frame is unreliable at high pitches. However, with the motional clue provided by sequential frames and the feature aggregation by the self-attention module, the error for yaw angle estimation can be reduced significantly. A simple smoothing strategy was also evaluated by averaging the adjacent five predicted angles of the CNN model inspired by [19]. The MAE of pitch angle was 9.4° (reduced by 0.2° compared to the CNN model), but the MAE of yaw angle was degraded to 17.8° (increased by 0.5°). We analyzed the sequential output and found that the smoothed angles lagged behind the ground-truth when the finger rotated quickly. The angles predicted by the CNN model were more accurate without smoothing in this case. Simple smoothing does not improve accuracy for challenging scenarios either (shown in Figure 11) when the CNN model struggled since the predicted angle for each frame was inaccurate.

- **tracking loss.** We removed the tracking loss function in Equation 6 and only used the angular MSE loss to train the TrackPose model. The performance was shown in Table 2. Angular errors were lower when using the tracking loss. The model trained with tracking loss was more stable, which was revealed by the MAE of increments (speeds). We also noticed an important fact that the $MAE_\delta$ of the TrackPose model without tracking loss was much smaller than that of the CNN model. This proved that the self-attention time-series modeling in the TrackPose architecture helped a lot for stable estimations.
- **user adaptive fine-tuning.** The effectiveness of per-user fine-tuning could be revealed by the comparisons of general TrackPose model and customized TrackPose model in Table 1. We found that the improvement of per-user adaption was significant for thumbs in terms of yaw angle. The MAE of the customized TrackPose model for thumbs was 9.6°, reduced by 33% compared with that of the general TrackPose model ($MAE = 14.3$). We pondered that this was because the difference of participates' thumbs was more distinct than that of other fingers. Among the volunteers, there was a tall man who has much larger fingers. The MAE of the general TrackPose model for him was 19°. And it was reduced to 13° after registration. We noticed that the customized model performed slightly worse for pitch angle. This may be because the model focused more on optimizing the performance of yaw angle during the fine-tuning process. However, the overall MAE of the customized model was significantly lower.

## 8 USER STUDY

Finger angle has many potential applications [19], including new gestures, 3D manipulation and new user interfaces. With the support of more accurate and stable TrackPose model, using finger poses for interactions becomes easier and more reliable (see Figure 12). For instance, we can now manipulate 3D models by performing yaw and pitch, which is more intuitive than the traditional swiping and dragging gestures. It is also possible

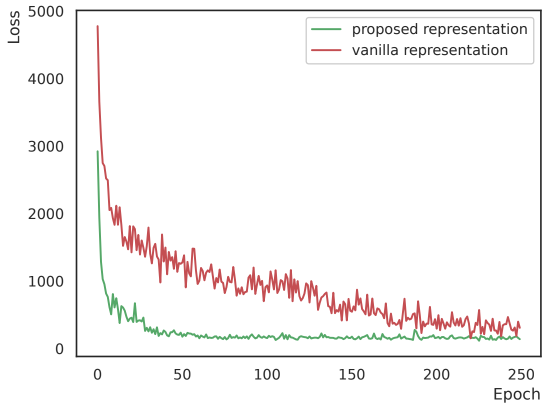


Fig. 9. Convergency speeds comparison. The model trained with the proposed rotatation representation converged much faster than that trained with vanilla representation.

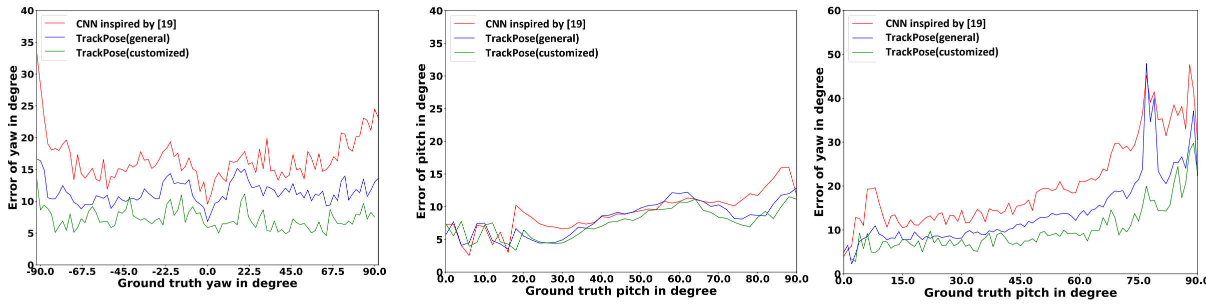
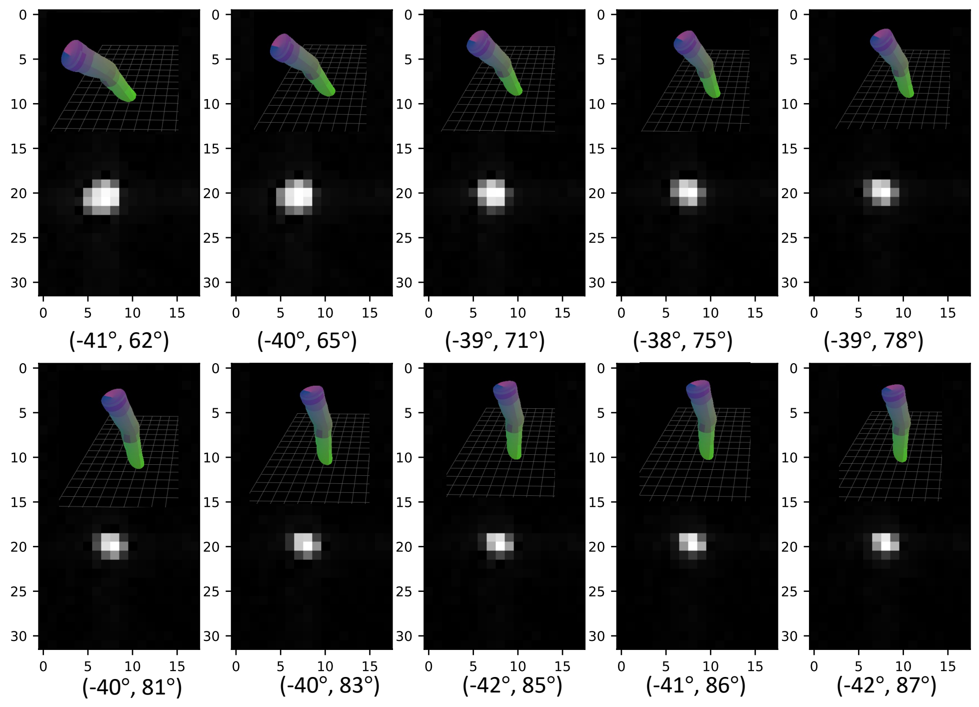


Fig. 10. Error distributions of yaw and pitch angles in our dataset. Observe the rightmost figure. Yaw estimation at high pitch ($> 60°$) is very difficult for single-frame models since the capacitive blobs are very small in this case. Errors are reduced a lot by using time-series modeling.

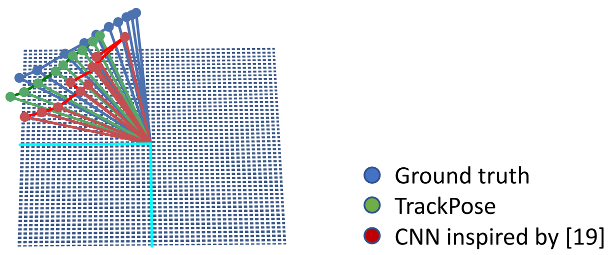
Table 2. Performances with and without tracking loss.

| | Yaw | | | | Pitch | | | | Overall | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Method** | **RMSE** | **MAE** | **SD** | **MAE$_\Delta$** | **RMSE** | **MAE** | **SD** | **MAE$_\Delta$** | **MAE** | **MAE$_\Delta$** |
| General TrackPose w/ tracking loss | **16.1** | **10.9** | **11.8** | **2.2** | **11.5** | **8.9** | **7.3** | **1.4** | **9.9** | **1.8** |
| General TrackPose w/o tracking loss | 18.4 | 11.3 | 14.5 | 3.7 | 12.0 | 9.3 | 7.6 | 1.8 | 10.3 | 2.7 |

to directly use a smartphone to control a toy car with a robot arm because we now have two additional reliable degrees of freedom. Our algorithm exhibits strong universality to diverse populations, finger conditions, and holding gestures. For more details, we recommend the audience to watch the supplemental demo video.
To evaluate the performance of different finger angle estimation algorithms in realistic interactive scenarios, we designed an interaction task following [22], which is a standard experiment to evaluate 3D object manipulation



(a)



(b)

Fig. 11. (a) images and ground-truth finger poses, (b) trajectories of the finger movement (The finger is represented by a line segment for visualization). A single capacitive image at high pitch contains limited information. It is not a "comet" shape any more (mentioned in [37]). While the changes of the blob shape contain useful context for finger pose estimation. Ground-truth angles are labeled under images and plotted in blue trajectories. Predicted trajectories of the single-frame model are marked in red, and the trajectories of TrackPose are marked in green. The TrackPose trajectory is more accurate and smoother.

techniques. The general TrackPose model, customized TrackPose model, and the CNN model inspired by [19] were tested.

### 8.1 Participants

We recruited 12 participants (10 male and 2 female, aged from 18 to 32) who are college students and workers. All participants are right-handed except for one. The fingerbreadths range from 1.2 cm to 2.5 cm.

### 8.2 Apparatus

The experiment was conducted in various places, including classrooms, meeting rooms, outdoors, etc. The smartphone used was a Realme C11. Since the finger angle estimation algorithms ran on Pytorch, we connected the phone to a computer (Figure 13A). Participants hold the smartphone in different gestures to complete the task without restraint.

### 8.3 Procedure

For each participant, the experiment was conducted through the following steps:

(1) We taught each participant how to use his/her finger to manipulate an airplane object on the screen (see Figure 13). Participants spent 2 minutes on average to get familiar with the operations thanks to the the intuitiveness of the angular mapping.
(2) We showed the participant that the target of the task was to rotate the moving airplane (red in Figure 13) until it matched the target pose of the reference airplane (white in Figure 13).
(3) A tested model was selected randomly to estimate the finger angles for operations from the following: customized TrackPose, general TrackPose and the CNN inspired by [19]. The participant did not know which model was used currently.
(4) Ten different poses were initialized for the moving airplane. The completion times were recorded.
(5) Repeat step (3) and step (4) until all models were tested.
(6) The participant was invited to fill out a questionnaire about the preference of different models and the feeling of using finger angles for 3D manipulation.

### 8.4 Results: Quantitative Comparisons

As shown in Figure 14, the average completion time in seconds for our best model was 16 ($SD = 5$), which outperformed the CNN model inspired by [19] ($M = 24, SD = 8$) by 33%. This result is consistent with the performance in Table 1. And it suggests that the design factors in our study, including time-series modeling and per-user fine-tuning, contribute to the improved user experience by providing more accurate and smoother angle estimation.

### 8.5 Results: Subjective Evaluation

We conducted a subjective study of the preferences of the three tested models by filling out a demographic questionnaire. The vast majority of users said that the customized TrackPose model ”feels smoother and stable” so that they prefer that model. While the single-frame model ”has some waggles” that makes them ”confused about how to fix it for twisting the airplane”. We also observed that participants intended to avoid yawing at high pitches when using the single-frame model. One participant said ”I can hardly perform rotation (yaw) at high pitches because it waggles randomly and makes the airplane far from the target place”. This phenomenon was consistent to the performance analysis mentioned in Section 7.6 (see Figure 10 and 11). Across participants, the customized TrackPose model was ranked as the most preferred model, followed by the general TrackPose model (see Table 3).

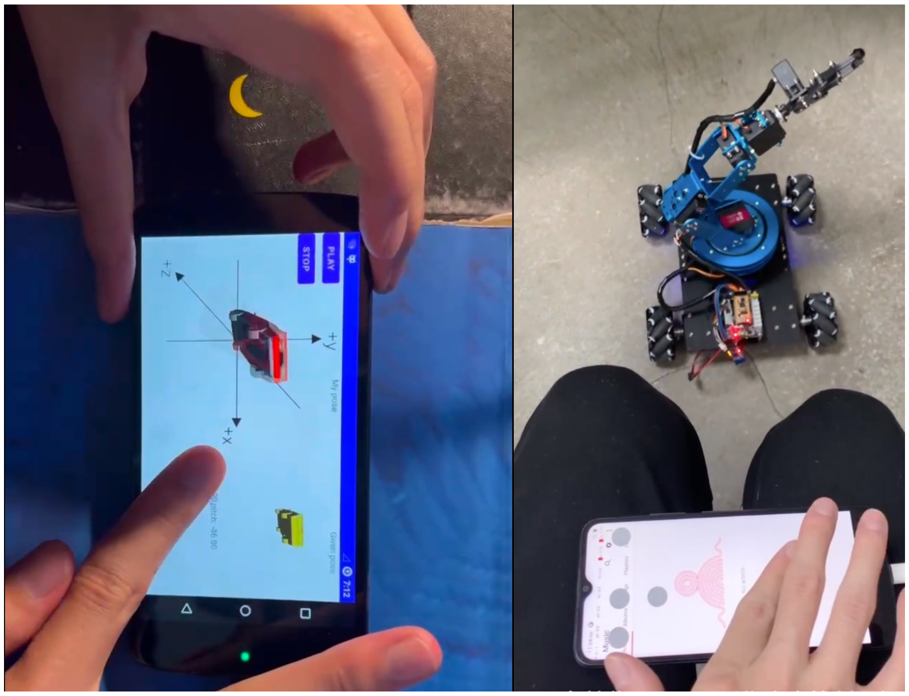


Fig. 12. Use TrackPose to manipulate 3D models and control a toy car with a robot arm.

Table 3. Subjective preferences (number of users assigning the rank). The lower the ranking, the stronger the preference.

| | **Rank 1 (best)** | **Rank 2 (middle)** | **Rank 3 (worst)** | **Mean Rank** |
|---|---|---|---|---|
| Customized TrackPose | 8 | 2 | 1 | 1.4 |
| General TrackPose | 3 | 7 | 1 | 1.8 |
| CNN inspired by [19] | 0 | 2 | 9 | 2.8 |

## 9 LIMITATION AND FUTURE WORK

This study has several limitations. As for the model architecture, using 5 adjacent frames of images as input causes a minor delay (0.08 seconds at 60 FPS). This effect needs to be further explored. Additionally, the bottleneck of runtime speed is the self-attention matrix computation (see Section 5.2). This may increase the power comsuption. It could be improved by utilizing a linear self-attention mechanism in [35]. The memory footprint is still a bit large when running on mobile devices. This could be improved by model pruning and distillation. Secondly, we find that all models including baselines and TrackPose performed poorly for wet fingers. We guess this is because the distributions of capacitive values is different in this case, and similar data is lacked in the training dataset. This issue could be mitigated by collecting capacitive images and finger poses for wet fingers specifically. In addition, finger tips are generally less conductive for older people. Further work needs to be done to verify the performance for TrackPose on their fingers. Evaluations across devices were not done since modification of Android kernel was difficult for recent mobile phones without the official permission. As for the registration application, an effective method for recording accurate pitch values should be investigated in the future. To use

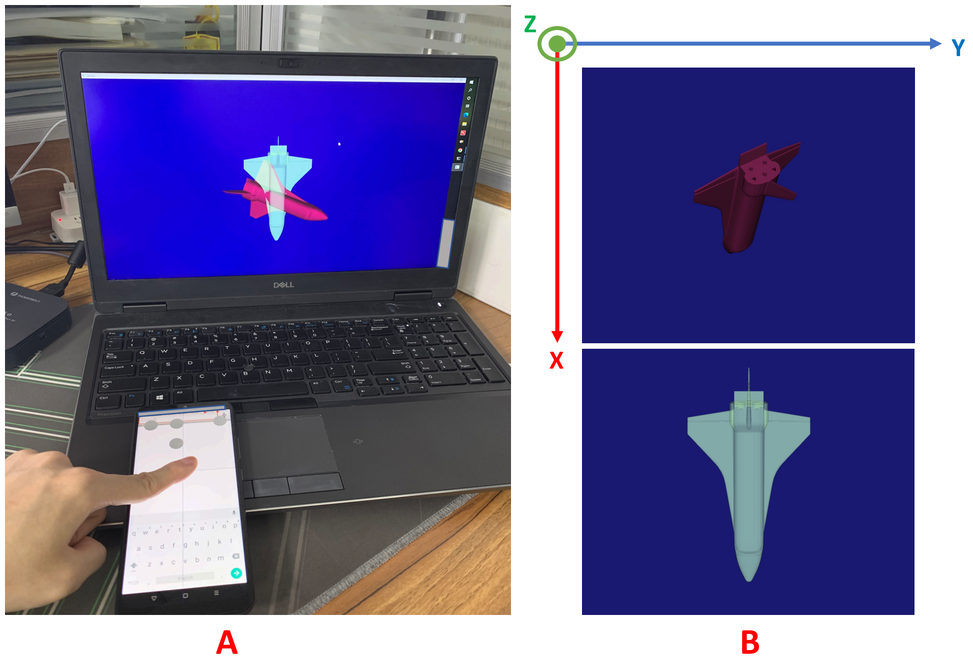


Fig. 13. A) Experiment apparatus. B) Illustration of the task. The user must rotate the airplane from the pose above to the target pose below by rotating the fingers. The finger rotation angles were estimated by baselines or TrackPose. Right pane shows that 3D object rotation is implemented by separability of manipulation.

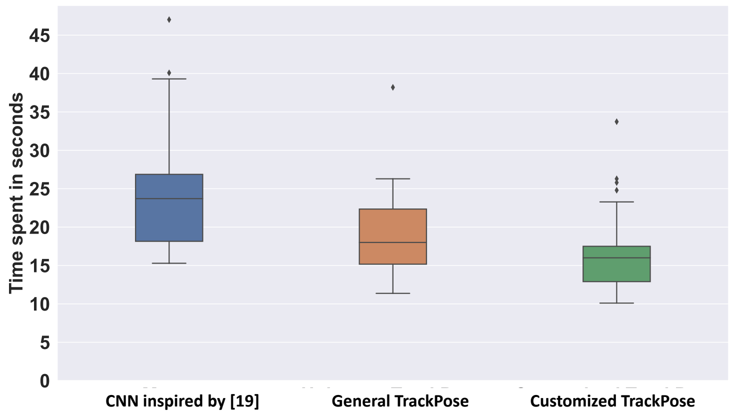


Fig. 14. Task completion time in seconds. Error bars: 95% CI.

customized TrackPose, now we rely on the speculation of usage scenarios to load the specific model for a finger. For example, when the user holds the smartphone sideways, the customized TrackPose model for his/her thumb will be used to boost the experience. In the future, we will integrate finger recognition technology to automatically determine which customized model to use for a certain finger. Additionally, user privacy problem should be considered in this stage.

## 10 CONCLUSION

Measuring the finger angle from the touchscreen is a very promising interaction technology. Due to the low accuracy and instability of existing researches, this technology has not been widely used in practice. In this study, we focused on improving the accuracy and stability of finger pose estimation using capacitive images. We achieved this through three measures. Firstly, we proposed an accurate and non-singular 2D rotation representation by minimizing errors between the representation space and the original space. This set a reasonable and easy-to-learn objective for learning-based algorithms. Secondly, we contributed a sequential model, TrackPose, which used continuous capacitive images as input and embedded the relationship of sequential images using self-attention modules. Thirdly, we advanced a new strategy to fine-tune TrackPose model for specific user using the representative data collected by a registration application. We demonstrated the effectiveness of our method through quantitative and subjective experiments. The angle estimation error was reduced by 33% compared with baselines, 47% for the yaw angle especially. The $MAE_{\Delta}$ which revealed instability was reduced by 62%. We also performed a 3D object manipulating experiment to prove that design factors contributed to the improved experience. The time cost of our best model was 33% less than that of the baseline method. The subjective preferences of the participants verified the effectiveness of the proposed method.

## ACKNOWLEDGMENTS

This research was supported in part by the National Natural Science Foundation of China under Grant 61976121 and 62376132. We would like to thank all the anonymous reviewers for their constructive suggestions.

## REFERENCES

[1] Karan Ahuja, Paul Streli, and Christian Holz. 2021. TouchPose: hand pose prediction, depth estimation, and touch classification from capacitive images. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. 997–1009.

[2] Anil Ufuk Batmaz, Mohammad Rajabi Seraji, Johanna Kneifel, and Wolfgang Stuerzlinger. 2020. No jitter please: Effects of rotational and positional jitter on 3D mid-air interaction. In *Proceedings of the Future Technologies Conference*. Springer, 792–808.

[3] Anil Ufuk Batmaz and Wolfgang Stuerzlinger. 2019. The effect of rotational jitter on 3D pointing tasks. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems*. 1–6.

[4] Tobias Boceck, Sascha Sprott, Huy Viet Le, and Sven Mayer. 2019. Force Touch Detection on Capacitive Sensors Using Deep Neural Networks. In *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services* (Taipei, Taiwan) *(MobileHCI '19)*. Association for Computing Machinery, New York, NY, USA, Article 42, 6 pages. https://doi.org/10.1145/3338286.3344389

[5] Sebastian Boring, David Ledo, Xiang'Anthony' Chen, Nicolai Marquardt, Anthony Tang, and Saul Greenberg. 2012. The fat thumb: using the thumb's contact size for single-handed mobile interaction. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services*. 39–48. https://doi.org/10.1145/2371574.2371582

[6] Frederick Choi, Sven Mayer, and Chris Harrison. 2021. 3D Hand Pose Estimation on Conventional Capacitive Touchscreens. In *Proceedings of the 23rd International Conference on Mobile Human-Computer Interaction*. 1–13.

[7] Chi Tai Dang and Elisabeth André. 2011. Usage and recognition of finger orientation for multi-touch tabletop interaction. In *INTERACT'11 Proceedings of the 13th IFIP TC 13 International Conference on Human-computer Interaction - Volume Part III*. 409–426.

[8] Yongjie Duan, Ke He, Jianjiang Feng, Jiwen Lu, and Jie Zhou. 2022. Estimating 3D Finger Pose via 2D-3D Fingerprint Matching. In *27th International Conference on Intelligent User Interfaces*. 459–469.

[9] F Sebastian Grassia. 1998. Practical parameterization of rotations using the exponential map. *Journal of graphics tools* 3, 3 (1998), 29–48.

[10] Tobias Grosse-Puppendahl, Christian Holz, Gabe Cohn, Raphael Wimmer, Oskar Bechtold, Steve Hodges, Matthew S. Reynolds, and Joshua R. Smith. 2017. *Finding Common Ground: A Survey of Capacitive Sensing in Human-Computer Interaction*. Association for Computing Machinery, New York, NY, USA, 3293–3315. https://doi.org/10.1145/3025453.3025808

[11] On Haran. 2014. Technologies and Requirements for Digital Pens. *Information Display* 30, 4 (2014), 6–11.

[12] Chris Harrison and Scott Hudson. 2012. 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* (Austin, Texas, USA) *(CHI '12)*. Association for Computing Machinery, New York, NY, USA, 3149–3152. https://doi.org/10.1145/2207676.2208730

[13] Chris Harrison, Julia Schwarz, and Scott E. Hudson. 2011. TapSense: Enhancing Finger Interaction on Touch Surfaces. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association

for Computing Machinery, New York, NY, USA, 627–636. https://doi.org/10.1145/2047196.2047279

[14] Ke He, Yongjie Duan, Jianjiang Feng, and Jie Zhou. 2022. Estimating 3D Finger Angle via Fingerprint Image. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 1 (2022), 1–22.

[15] Sven Kratz, Patrick Chiu, and Maribeth Back. 2013. 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* (St. Andrews, Scotland, United Kingdom) *(ITS '13)*. Association for Computing Machinery, New York, NY, USA, 223–230. https://doi.org/10.1145/2512349.2512824

[16] Huy Viet Le, Sven Mayer, and Niels Henze. 2019. Investigating the feasibility of finger identification on capacitive touchscreens using deep learning. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. 637–649.

[17] William E Lorensen and Harvey E Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *ACM siggraph computer graphics* 21, 4 (1987), 163–169.

[18] Guy Lüthi, Andreas Rene Fender, and Christian Holz. 2022. DeltaPen: A Device with Integrated High-Precision Translation and Rotation Sensing on Passive Surfaces. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*. 1–12.

[19] 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*. ACM. https://doi.org/10.1145/3132272.3134130

[20] Sven Mayer, Michael Mayer, and Niels Henze. 2017. Feasibility analysis of detecting the finger orientation with depth cameras. In *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM. https://doi.org/10.1145/3098279.3122125

[21] Sven Mayer, Xiangyu Xu, and Chris Harrison. 2021. Super-resolution capacitive touchscreens. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–10.

[22] Daniel Mendes, Fabio Marco Caputo, Andrea Giachetti, Alfredo Ferreira, and Joaquim Jorge. 2019. A survey on 3d virtual object manipulation: From the desktop to immersive virtual environments. In *Computer graphics forum*, Vol. 38. Wiley Online Library, 21–45.

[23] Moaaz Hudhud Mughrabi, Aunnoy K Mutasim, Wolfgang Stuerzlinger, and Anil Ufuk Batmaz. 2022. My eyes hurt: Effects of jitter in 3d gaze tracking. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. IEEE, 310–315.

[24] Ian Oakley, Carina Lindahl, Khanh Le, DoYoung Lee, and MD. Rasel Islam. 2016. *The Flat Finger: Exploring Area Touches on Smartwatches*. Association for Computing Machinery, New York, NY, USA, 4238–4249. https://doi.org/10.1145/2858036.2858179

[25] Philip Quinn, Wenxin Feng, and Shumin Zhai. 2021. *Deep Touch: Sensing Press Gestures from Touch Image Sequences*. Springer International Publishing, Cham, 169–192. https://doi.org/10.1007/978-3-030-82681-9_6

[26] Gonzalo Ramos, Matthew Boulos, and Ravin Balakrishnan. 2004. Pressure Widgets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria) *(CHI '04)*. Association for Computing Machinery, New York, NY, USA, 487–494. https://doi.org/10.1145/985692.985754

[27] Simon Rogers, John Williamson, Craig Stewart, and Roderick Murray-Smith. 2011. *AnglePose: Robust, Precise Capacitive Touch Tracking via 3d Orientation Estimation*. Association for Computing Machinery, New York, NY, USA, 2575–2584. https://doi.org/10.1145/1978942.1979318

[28] Anne Roudaut, Eric Lecolinet, and Yves Guiard. 2009. *MicroRolls: Expanding Touch-Screen Input Vocabulary by Distinguishing Rolls vs. Slides of the Thumb*. Association for Computing Machinery, New York, NY, USA, 927–936. https://doi.org/10.1145/1518701.1518843

[29] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. 2009. Learning 3-d object orientation from images. In *2009 IEEE International conference on robotics and automation*. IEEE, 794–800.

[30] Robin Schweigert, Jan Leusmann, Simon Hagenmayer, Maximilian Weiß, Huy Viet Le, Sven Mayer, and Andreas Bulling. 2019. KnuckleTouch: Enabling Knuckle Gestures on Capacitive Touchscreens Using Deep Learning. In *Proceedings of Mensch Und Computer 2019* (Hamburg, Germany) *(MuC'19)*. Association for Computing Machinery, New York, NY, USA, 387–397. https://doi.org/10.1145/3340764.3340767

[31] Paul Streli and Christian Holz. 2021. CapContact: Super-Resolution Contact Areas from Capacitive Touchscreens. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–14.

[32] Jamie Ullerich, Maximiliane Windl, Andreas Bulling, and Sven Mayer. 2022. ThumbPitch: Enriching Thumb Interaction on Mobile Touchscreens using Deep Learning. In *33nd Australian Conference on Human-Computer Interaction* (2022-11-29) *(OzCHI22)*. Association for Computing Machinery, Canberra, NSW, Australia. https://sven-mayer.com/wp-content/uploads/2022/08/ullerich2022thumbpitch.pdf

[33] Jonas Vogelsang, Francisco Kiss, and Sven Mayer. 2021. A Design Space for User Interface Elements using Finger Orientation Input. In *Proceedings of Mensch und Computer 2021*. 1–10.

[34] Feng Wang, Xiang Cao, Xiangshi Ren, and Pourang Irani. 2009. Detecting and leveraging finger orientation for interaction with direct-touch surfaces. In *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology - UIST '09*. ACM Press. https://doi.org/10.1145/1622176.1622182

[35] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768* (2020).

[36] Yoichi Watanabe, Yasutoshi Makino, Katsunari Sato, and Takashi Maeno. 2012. Contact force and finger angles estimation for touch panel by detecting transmitted light on fingernail. In *EuroHaptics'12 Proceedings of the 2012 International Conference on Haptics: Perception, Devices, Mobility, and Communication - Volume Part I*. 601–612.

[37] 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 Press. https://doi.org/10.1145/2817721.2817737

[38] Dongseok Yang, Doyeon Kim, and Sung-Hee Lee. 2021. Lobstr: Real-time lower-body pose prediction from sparse upper-body tracking signals. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 265–275.

[39] Vadim Zaliva. 2012. 3D finger posture detection and gesture recognition on touch surfaces. In *2012 12th International Conference on Control Automation Robotics & Vision (ICARCV)*. IEEE. https://doi.org/10.1109/icarcv.2012.6485185

[40] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. 2019. On the Continuity of Rotation Representations in Neural Networks. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 5738–5746.