SpeciFingers: Finger Identification and Error Correction on Capacitive Touchscreens

ZEYUAN HUANG, CANGJUN GAO, HAIYAN WANG, and XIAOMING DENG, Institute of Software, Chinese Academy of Sciences, China and University of Chinese Academy of Sciences, China
YU-KUN LAI, Cardiff University, United Kingdom
CUIXIA MA*, Institute of Software, Chinese Academy of Sciences, China and University of Chinese Academy of Sciences, China
SHENG-FENG QIN, Northumbria University, United Kingdom
YONG-JIN LIU, Tsinghua University, China
HONGAN WANG, Institute of Software, Chinese Academy of Sciences, China and University of Chinese Academy of Sciences, China

The inadequate use of finger properties has limited the input space of touch interaction. By leveraging the category of contacting fingers, finger-specific interaction is able to expand input vocabulary. However, accurate finger identification remains challenging, as it requires either additional sensors or limited sets of identifiable fingers to achieve ideal accuracy in previous works. We introduce SpeciFingers, a novel approach to identify fingers with the capacitive raw data on touchscreens. We apply a neural network of an encoder-decoder architecture, which captures the spatio-temporal features in capacitive image sequences. To assist users in recovering from misidentification, we propose a correction mechanism to replace the existing undo-redo process. Also, we present a design space of finger-specific interaction with example interaction techniques. In particular, we designed and implemented a use case of optimizing the performance in pointing on small targets. We evaluated our identification model and error correction mechanism in our use case.

CCS Concepts: • **Human-centered computing** → **Interaction techniques**; **Touch screens**; *Ubiquitous and mobile computing systems and tools.*

Additional Key Words and Phrases: Finger identification, Finger-specific interaction, Capacitive touchscreen, Deep learning, Error correction

*indicates the corresponding author

Authors' addresses: Zeyuan Huang, zeyuan2020@iscas.ac.cn; Cangjun Gao, gaocangjun23@mails.ucas.ac.cn; Haiyan Wang, wanghaiyan@iscas.ac.cn; Xiaoming Deng, xiaoming@iscas.ac.cn, Beijing Key Laboratory of Human-Computer Interaction, Institute of Software, Chinese Academy of Sciences, Beijing, China , School of Computer Science and Technology and University of Chinese Academy of Sciences, Beijing, China; Yu-Kun Lai, LaiY4@cardiff.ac.uk, School of Computer Science and Informatics, Cardiff University, Cardiff, Wales, United Kingdom; Cuixia Ma, cuixia@iscas.ac.cn, Beijing Key Laboratory of Human-Computer Interaction, Institute of Software, Chinese Academy of Sciences, Beijing, China , School of Computer Science and Technology and University of Chinese Academy of Sciences, Beijing, China; Sheng-feng Qin, sheng-feng.qin@northumbria.ac.uk, School of Design, Northumbria University, Newcastle, United Kingdom; Yong-Jin Liu, liuyongjin@tsinghua.edu.cn, Department of Computer Science and Technology, MOE-Key Laboratory of Pervasive Computing, Tsinghua University, Beijing, China; Hongan Wang, hongan@iscas.ac.cn, Beijing Key Laboratory of Human-Computer Interaction, Institute of Software, Chinese Academy of Sciences, Beijing, China , School of Computer Science and Technology and University of Chinese Academy of Sciences, Beijing, China.

ACM 2474-9567/2024/1-ART8
https://doi.org/10.1145/3643559

ACM Reference Format:
Zeyuan Huang, Cangjun Gao, Haiyan Wang, Xiaoming Deng, Yu-Kun Lai, Cuixia Ma, Sheng-feng Qin, Yong-Jin Liu, and Hongan Wang. 2024. SpeciFingers: Finger Identification and Error Correction on Capacitive Touchscreens . *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 8, 1, Article 8 (January 2024), 28 pages. https://doi.org/10.1145/3643559

1 INTRODUCTION

Over the past years, touch input has become one of the most common interactions on digital surfaces in our daily lives. Taking advantage of the intuitive interface assembling input and output, touchscreens enable natural and dynamic interactions on ubiquitous computing devices. On the other hand, touch input itself suffers from the drawback of limited vocabulary because touch points are treated equally without differentiation.

Traditional touchscreens translate touch input into 2D coordinates, which omits the properties of interacting hand and fingers. Researchers have extensively explored the possibilities to expand the input space from various sources of input (such as differentiated fingers [12, 17, 36], areas of finger [23], palm [34], nail [25], knuckle [55], ear [63], object [42], etc.) and states of input (such as pre-touch [21], finger shape [46], touch force [5], 3D hand pose [1, 7], 3D hand mesh [1], finger angle [19, 41, 65], etc.). Such additional input dimensions enable more awareness of users' input and support broader interactions.

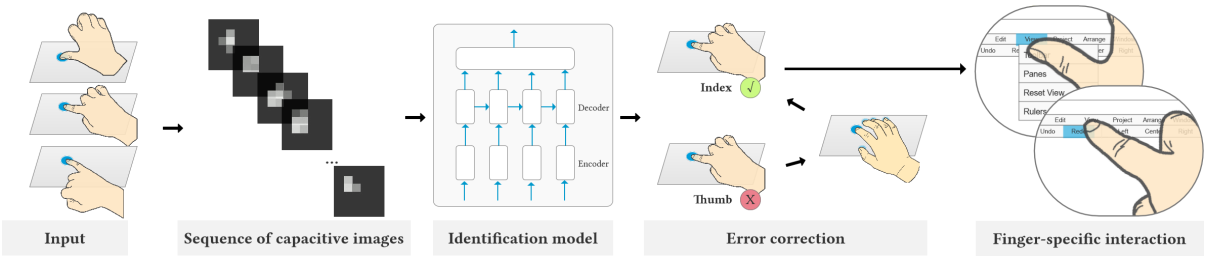


Fig. 1. Overview of SpeciFingers: (1) input on capacitive touchscreen with specific fingers, (2) capture the sequence of capacitive images, (3) identify the contact finger using spatial and sequential features, (4) error correction for misidentification results, (5) finger-specific interaction - pointing on small targets with finger-layout binding.

Among these techniques to access new input properties, finger identification has accumulated much attention from researchers. Previous studies have developed identification techniques with a variety of sensors to implement finger identification. The approaches using external sensors like vision sensors [16] and wearable sensors [39, 47, 48, 50, 60] achieve higher accuracy but require additional hardware. To achieve better usability, other studies apply built-in sensors to capture the signal of finger contact. Kim and Oakley [29] use the speaker and microphone of a smartwatch to identify thumb, index and middle fingers. Capacitive touchscreens produce capacitive raw data that can be regarded as "capacitive images" [12] for identification. Using capacitive images, Gil et al. [12] achieve a 90%+ accuracy for thumb, index and middle fingers but only in exaggerated gestures while Le et al. [36] identify thumb fingers of left or right hands with an accuracy of 90%. Some compromise has to be made between the requirement of additional sensors and limited identifiable finger sets for ideal accuracy.

Even though the accuracy of existing identification techniques can reach over 90%, it inevitably decreases in practice [36] and leads to misidentification results. Typically, the responses of interface to touch inputs assume that all inputs are correctly identified. Otherwise, it leads to errors in which the outcomes do not match the intent of input. Users need to undo the erroneous operation and redo the previous input to clarify the real intent. But if the error occurs repetitively, it is frustrating that the process needs to be repeated until the user's intent is correctly captured. Therefore, such interface requires toleration of incorrect identification results and convenient

recovery from misidentification errors. Correction of input errors on touch interfaces is less frequently proposed. Jung and Jang [28] proposed a two-step interaction to correct selection errors. It is necessary and valuable to investigate the correction of input recognition errors on touch interfaces.

Finger-specific interactions, or assigning different functionality to specific fingers, have been extensively investigated, illustrating their promising value and potential. Previous work bound fingers to functions on elements, or assigned functions directly to the fingers. The former allows for the multiplexing of interface elements, whereas the latter provides a more direct and convenient way of interaction. The proposed interaction techniques are mostly application-oriented. There has been a lack of interaction frameworks that can exploit the properties of input fingers as well as the task characteristics.

To try to address these issues, we introduce *SpeciFingers* in this paper, a novel approach to identifying fingers on capacitive touchscreens of tabletop devices (see Figure 1). Previous work revealed that capacitive raw data of touchscreen allows for accurate identification of a limited set of fingers or intentionally exaggerated gestures [12, 36]. Recognizing the extensive use and integrated hardware implementation of capacitive touchscreens, we argue that expanding the identifiable finger set is still a useful addition to enrich the variety of interaction. We apply an encoder-decoder architecture neural network to make full use of the sequential features throughout the contacting process rather than just within a single frame of capacitive images. We collected a dataset containing four typical single-finger actions, specifically including the pressing action as distinguished from the tapping action with shorter duration. Taking a user-specific strategy, we reach a mean accuracy of over 85% to differentiate between thumb, little finger and the other three fingers. We further design and implement an interactive correction mechanism to correct the misidentification errors by placing all five / ten fingers open on the screen. The correction mechanism is suitable not only for the model we propose, but also for correcting the misidentification errors of more fingers.

We present a design space for finger-specific interactions to inspire more interaction techniques. A use case of optimizing the pointing on small targets with finger-layout binding was also developed and implemented. We evaluate our pipeline with finger identification and error correction in the scenario of the use case. The evaluation results highlight the 99.45% accuracy and the advantages in preventing error and recovering from error with our pipeline. The proposed finger-layout binding interaction also demonstrates significantly less visual attention occupation and higher usefulness in the user study. In order to bootstrap further study in this area, we are publicly releasing our dataset, model and source code [1].

The contribution of this work can be summarized in three-fold: (1) a state-of-the-art model to identify fingers on capacitive touchscreens taking advantage of sequential features; (2) the first interactive correction mechanism to correct misidentification errors; (3) a novel interaction design space with example interaction techniques and a use case to optimize pointing performance on small targets with finger-layout binding.

2 RELATED WORK

2.1 Finger Identification on Ubiquitous Devices

Researchers have investigated various sensing techniques for recognizing the contacting finger on the screen and have developed corresponding interaction techniques on the target devices.

Vision sensors are predominantly used in finger identification. Researchers obtain the positions of hand and fingers with *external cameras*. Earlier efforts marked fingers with color tags [62] and color-tagged rings [13] for successful finger identification to explore the design of interaction techniques. More efforts identify the contacting fingers by estimating hand skeleton with various cameras like web cameras [68], fisheye cameras [49], depth cameras [44], and infrared cameras [9]. *Infrared cameras beneath the screens* are often deployed to facilitate human body and object interactions on desktop devices. Existing works implement finger identification

[1]Dataset, model, and source code for SpeciFingers are available at: https://github.com/iscas-MMSketch/SpeciFingers

Table 1. Overview of existing literature on finger-specific interactions and sensing techniques for identifying fingers. Fingers are abbreviated as the first letter of the word (i.e., thumb (T), index (I), middle (M), ring (R), and little (L)), and the subscripts indicate the left(L)/right(R) hand. The devices are categorized into *tabs, pads and boards* according to the basic forms for ubiquitous computing devices [64].

Literature	Fingers	Device	Sensing technique	Interactions
FingerSense [62]	T, I, M, R, L	Tabs (Button)	CMOS camera	Button multiplexing
Benko et al. [4]	I, M	Pads (Tablet)	Electromyography sensor	Painting
Au and Tai [2]	T, I, M, R, L	Boards (Tabletop)	Web camera	Palm menu, virtual mouse
Marquardt et al. [37, 38]	ALL	Boards (Tabletop)	Fiduciary-tagged gloves	Hand-part and posture aware interactions
Ewerling et al. [11]	ALL	Boards (Tabletop)	Infrared camera	None
Murugappan et al. [44]	ALL	Boards (Horizontal)	Depth camera	Pen+touch interactions
Colley and Häkkilä [9]	T, I, M, R, L	Pads (Phone)	Infrared camera	Command assignment
Glass + Skin [54]	T, I, M, R, L	Pads (Tablet)	iPad screen	Menu instances
DualKey [17]	I, M	Tabs (Watch)	Infrared sensor	Text entry
Zheng and Vogel [68]	All	Boards (Laptop)	Camera	Keyboard shortcuts
Gupta et al. [16]	I, M	Pads (Phone)	Infrared sensor	Multitasking
Goguey et al. [13, 14]	T, I, M, R, L	Pads (Phone, tablet) Boards (Tabletop)	Web camera Gametrack string	Integration of command selection and parameter manipulation
TriTap [12]	T, I, M	Tabs (Watch)	Capacitive touchscreen	Text entry
Matulic et al. [40]	T, L	Boards (Tabletop)	Infrared camera	Menu
WhichFingers [39]	T, I, M, R, L	Boards (Laptop)	Vibration sensor	Precision assignment, position specification
TouchSense [3]	T, I, M	Pads (Phone, tablet)	Electromyography sensor	Finger-to-command mapping
InfiniTouch [35]	T, I, M	Pads (Phone)	Capacitive touchscreen	Keyboards
Le et al. [36]	T_L, T_R	Pads (Phone)	Capacitive touchscreen	Multitasking, painting
FingMag [50]	I, M, R	Tabs (Watch)	Magnet equipment	None
TelemetRing [60]	T, I, M, R, L	Boards (PC)	Ring-shaped sensor	Chord keyboard
MagTouch [48]	I, M, R	Tabs (Watch)	Magnet equipment	None
DeepFisheye [49]	T, I, M, R, L	Pads (Tablet)	Fisheye camera	Painting, command assignment
TapID [43]	T, I, M, R, L	Tabs (VR)	Accelerometers	Command assignment
Oh et al. [47]	T, I, M	Pads (Tablet)	Vibration sensor	Painting, command assignment, button multiplexing
SonarID [29]	T, I, M	Tabs (Watch)	Sonar	None

by exploiting Microsoft Surface's capability of identifying fiduciary tags [37, 38], or directly making use of the raw touch images from infrared cameras [11, 40]. The setups of vision sensors have led to challenges in device convenience and hardware integration. While recent approaches [49] have tried to overcome this challenge through optimizing camera properties and settings, it still imposes limitations on practical applications.

Wearable sensors have been popular in recent years for finger identification. Some works [16, 17] attached *infrared sensors on users' fingers* while other works utilized *signals yielded by touch actions*, like vibration [39, 47], magnetic field [48, 50], motion acceleration [43], passive inductive telemetry [60], and so on. Although these solutions achieve high identification accuracy, it is barely feasible in practice to require users to wear sensors on their arms and fingers.

Sonic sensors deployed on the devices do not require external sensors or user-worn equipments. SonarID [29] made use of the small differences in bio-acoustic signals captured by the speaker and microphone of a smartwatch to identify the finger above the smartwatch from thumb, index and middle fingers. Despite the benefits of hardware integration, SonarID has limited gesture vocabulary and uncertain robustness in different ambient noise conditions.

Capacitive sensors on touchscreens are promising sources for receiving raw data of touch input. The low capacitive image resolution is the major limitation for commercial touchscreen devices to implement finger recognition applications. Therefore, researchers have worked on reconstructing low-resolution capacitive images [42, 58], or improving accuracy with different identification models [12, 35, 36]. Gil et al. [12] perform finger identification on capacitive touchscreens of smartwatches, reaching ideal results only when the gestures are

exaggerated. Le et al. [35] recognize finger contacts on the front and back of smartphones. Le et al. [36] support gestures commonly used on smartphones, but only recognize left and right thumbs with a high accuracy. In contrast to the recognition of each contact blobs, some recent work has attempted to reconstruct the structural information of hand utilizing all the contact blobs on the capacitive touchscreen. Choi et al. [7] estimated 3D hand pose while Ahuja et al. [1] predicted hand pose, depth image and contacting finger. While these efforts reconstruct 3D hand information to enable more interaction techniques, they are limited to a specific vocabulary of gestures and are only effective when multiple fingers are contacting the screen. Therefore, these approaches are not suitable for finger identification on general interactions.

We list an overview of the existing literature in Table 1. Existing efforts either require additional sensors like external or wearable sensors, or rely on a limited vocabulary of gestures to achieve usable accuracy. In this paper, we focus on the capacitive touchscreens, aiming to improve the performance of finger identification with the temporal features of touch input as a sequence of capacitive images.

Currently, finger identification based on capacitive touch data is mainly focused on smartphones [36] and smartwatches [12]. We believe that supporting finger-specific interactions on tabletop devices is also meaningful, as indicated by existing tabletop research [2, 11, 37, 38, 40]. However, existing finger identification methods on tabletop devices typically require additional sensors. Therefore, our work aims to enhance finger identification capabilities on tabletop devices with capacitive screens. Considering that tabletop devices generally have larger screens, the touch interaction methods may differ from smaller screen devices like smartphones. Hence, we propose an identification model trained on data collected from tabletop devices , along with an error correction mechanism, to establish a finger identification and correction pipeline.

2.2 Finger-specific Interactions

Finger-specific interaction assigns different functionality to different fingers, depending on the identification result of input finger. We present the interactions proposed by the existing literature in the last column of Table 1. We summarize the existing interactions from the following two perspectives:

Binding fingers to functions on elements allows users to perform the corresponding function on an element with a specific finger. Wang and Canny [62] and Oh et al. [47] built proof-of-concept prototypes for multiplexing a physical button. DualKey [17] and TriTap [12] bound different fingers to multiple characters of the same key on keyboard to realize text entry on smartwatches. Zheng and Vogel [68] demonstrated keyboard shortcuts triggered by specific fingers pressing physical keys. Other works define different application commands for different fingers on the target element [9, 43, 49]. Such interaction techniques enable the multiplexing of elements on the interface, thus supporting more functionality with a limited size of screen and number of elements.

Assigning functions directly to fingers allows users to perform the function with specific fingers on the whole interface. Some application examples of interaction correspond fingers to different painting tools [4, 36, 47, 49]. Gupta et al. [16] and Le et al. [36] assigned foreground or background applications to gestures of different fingers in porous multitasking interfaces. Masson et al. [39] define different precision of pointing or scrolling and positions of resizing window to gestures with different fingers. Takahashi et al. [60] presented a chord keyboard assigning orderly typing finger chords to different characters. Goguey et al. [13] and Murugappan et al. [44] explored bimanual finger-specific interactions.

The potential of finger-specific interactions has been indicated in early research. With the development of finger identification techniques, we take a deeper look at the possibilities in finger-specific interaction techniques from a view of design framework. In particular, we propose an interaction that binds fingers to interface layout, thus enabling more accurate pointing on small targets.

2.3 Input Error Correction

Since there might be mistakes in interpreting user's input on the interface, such as touch, gesture, voice, etc., an effective correction mechanism is necessary to avoid leading to an erroneous interaction flow. Researchers have proposed explicit and implicit correction approaches for different user interfaces.

Explicit correction approaches allow users to manually correct the mistaken result to bring the interaction flow back in accordance with the intention. Suhm et al. [59] proposed speech recognition error correction methods that allow users to correct errors multimodally. The work by Huggins-Daines and Rudnicky [24] assists users to correct speech recognition errors by visualizing and browsing the word lattice using gestures. Voice typing [33] enables real-time error identification by transcribing users' utterances and fast correction with a gesture-based marking menu to edit texts.

Implicit correction approaches automatically detect and correct the potential errors on the interface without manual intervention. Such approaches usually take advantage of the information from other modalities as a complement to detecting errors. Putze et al. [52] designed an error-aware gesture interface through detecting the Error Potential (ErrP) of user's brain activity with Electroencephalography (EEG) and thus recovering from the error pro-actively. The system promoted by Putze et al. [53] automatically identifies and repairs auto-correction errors in mobile text entry with multi-modal features from brain activity, gaze, and context. As a special scenario with contextual semantics, interfaces for text entry usually use language models [15, 61, 61, 66, 67] to detect and correct errors leveraging implicit patterns in the text content.

Fewer studies have addressed interactive error correction methods on touch interfaces. The existing approaches are designed with explicit interactions for users to correct errors. Jung and Jang [28] developed a two-step touch method for website navigation, in which a flick motion was conducted to correct the selection if the initial selection by press motion failed. No studies have yet addressed correction of errors in finger identification or other similar methods of enriching touch input vocabulary. In this work, we propose an explicit error correction procedure to assist in the convenient recovery from misidentification results.

3 DATA COLLECTION

The goal of this study is to collect the capacitive raw data of different fingers while performing typical single finger tasks: tapping, pressing, sliding, and dragging. With the data, we train and evaluate models to identify different fingers.

3.1 Design

3.1.1 Task Design. The participants were instructed to perform different tasks multiple times with each finger of both hands. We designed the tasks in the study considering the basic gesture vocabulary promoted by Clark [8] and previous work by Le et al. [34, 36]. The tasks contain four typical single finger actions including tap, press, slide, and drag, as shown in Figure 2.

Tapping task: Participants performed the tapping task by tapping the square target (50 × 50 px) on the touchscreen. The target appeared within the touch area randomly. The duration of action was not limited but the participants were required to complete the action close to real use.

Pressing task: Participants pressed the square target (50 × 50 px) on the touchscreen for a period of time (0.5 seconds). The participants were required not to finish the task before the signal area flashed red (time-up signal).

Sliding task: Participants completed the sliding task with the instruction of sliding direction. Four sliding directions were included: up, down, left, and right. The duration and position of sliding were not limited.

Dragging task: Participants were instructed to drag the square target to the destination position. The duration was not limited.

(a) Tap / Press (b) Slide (c) Drag

Fig. 2. Tasks of typical single finger actions on touchscreens.

3.1.2 Study Design. We adopted a 10×4 within-subjects design with the independent variables being the fingers and tasks. To obtain enough touch raw data and concrete behavior understanding on touch inputs, the four tasks were executed 40 times with each finger. In total, there were 10 FINGERS × 4 TASKS × 40 REPETITIONS = 1600 TRIALS per participant.

The order of fingers was balanced by a Balanced Latin Square. The tasks for each finger were randomly shuffled. The targets in tapping, pressing and dragging tasks appeared at randomized positions. Any part of the targets was ensured to be within the touch area. The 40 repetitions of sliding task were equally divided into up, down, left and right directions to cover the major sliding directions. The order of directions was also randomly shuffled.

3.2 Participants and Procedure

We recruited 20 participants (13 male and 7 female) with a mean age of 27.6 ($Std = 4.60$). Two of the participants are left-handed while the others are right-handed. All of the participants are familiar with touchscreen. The average hand size of participants ranged from 15.8cm to 19.8cm ($M = 18.37cm, Std = 1.12$).

We conducted the study in the laboratory environment to ensure the least distraction to the participants. After welcoming the participants, we introduced the goal of study and demonstrated the tasks in the study to them. The participants completed the informed consent form, in which they were told that they could take breaks or quit the study at any time. Afterwards, we collected their basic information including gender, age and dominant hand. We measured the participants' hand length from the base of the hand at the wrist crease to the tip of the middle finger [51].

The touchscreen device was laid on the desk at an angle of 20°. Participants were instructed to hover their hands over the screen when completing the tasks with each finger.

3.3 Apparatus

We used a Wacom DTH-2242 tabletop touchscreen device, which provides a 21.5" (475.2$mm \times 267.3mm$) display in a resolution of 1920$px \times 1080px$ with a density of 4.04px/mm. The touchscreen provides 122×70 raw touch data in a range of [0, 65535]. The sensitivity data was adapted as capacitive raw data and further generated into capacitive images. Some capacitive images are demonstrated in Figure 1.

The application for the user study was implemented based on the Wacom's Feel™ Multi-Touch SDK [2]. The raw touch data was logged every 10ms (100 FPS) to collect the sensitivity values in the sensitivity blocks. The log files were named with the respective participant ID, task and finger to label each action. The recording of each action automatically starts when participants' fingers contact the screen and ends when the fingers leave the screen. Therefore, participants do not need to manually separate each action.

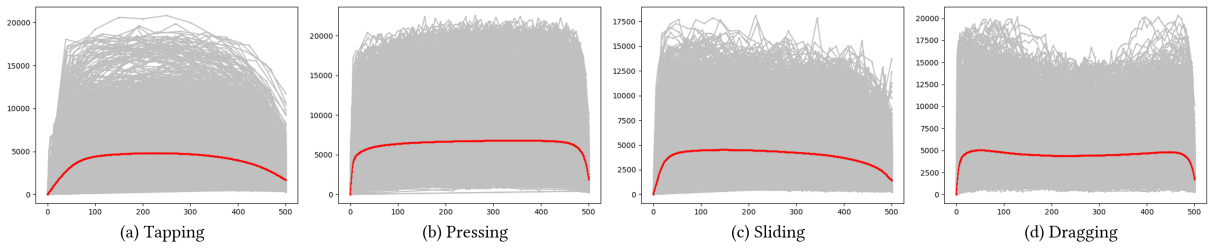
[2] https://developer-docs.wacom.com/intuos-cintiq-business-tablets/docs/wfmt-overview

Table 2. Details of datasets: SpeciFingers (ours) and CapFingerId [36].

		SpeciFingers	CapFingerId
Average sequence length	Tap	10.81	32.59
	Slide / Scroll	20.46	26.07
	Drag	74.50	19.55
	Press	84.26	–
Sensitivity range		0-1800	0-710
Sequence count		38153	17416
Actions		4	3
Participants		20	20
Device	Type	Tabletop	Phone
	Size	21.5"	4.96"
	Capacitive resolution	122×70	27×15

3.4 Data Exploration

To avoid wrong labels, we filtered out the wrongly labeled data that were manually recorded during the data collection, such as those of actions with wrong finger, wrong action, etc. We also removed the action data with unintentional touch by detecting the contact blobs in each frame of capacitive images. In total, our dataset consists of 38153 sequences of capacitive images. We report the details of our dataset in Table 2, compared with the only publicly released finger identification dataset on capacitive touchscreen [36]. We have found that the average duration of our task is shorter than the CapFingerId dataset [36]. We believe this is primarily because we differentiate between tap and press actions in our task, while the CapFingerId dataset only includes tap actions and requires users to "tap and hold the target for 1.5 seconds", which is more similar to our press task. As a result, the data sequences generated by our tap actions are much shorter.



(a) Tapping (b) Pressing (c) Sliding (d) Dragging

Fig. 3. Change of sum of sensitivity in different actions over time. The x-axis represents the position of the frame in sequence, and the y-axis represents the sum of sensitivity of this frame. Each gray line represents a sequence (with temporal dimension resampled to the same length), and the red curves are the average of sequences of the same action type.

As shown in Figure 3, we investigated the sequential pattern of sensitivity data during actions. There is a trend where the sum of sensitivity increases and then decreases over time during an action. Therefore we later captured and exploited the hidden sequence features by a deep learning approach.

4 METHOD OF FINGER IDENTIFICATION

In this section, we present our finger identification model and its performance on the datasets mentioned above. We tracked the contact finger between frames to access the sequence of capacitive images and used a model of encoder-decoder architecture to capture the features within and between frames. We empirically studied the finger combinations with usable accuracy and further adapted a user-specific strategy to improve model performance on individual users.

4.1 Data Preprocessing

Our model was trained and validated on two datasets: CapFingerId [36] and our SpeciFingers. For the SpeciFingers dataset, we generated the capacitive images with sensitivity raw data in a matrix of 122×70, and converted them to gray-scale images in a resolution of $1920px \times 1080px$. The contact blob was then cut out and resized to $224px \times 224px$. For the CapFingerId dataset, we generated capacitive images with capacitive raw data in 15×17. The contact blob was cropped out and resized to $224px \times 224px$.

We ensured that the capacitive image sequences of different actions share the same length. We used linear interpolation for the actions with shorter sequence length and average sampling for those with longer sequence length.

In order to improve the generalization of the prediction model, online augmentation was applied in the training process. In every epoch, every sequence of images has a probability of being flipped horizontally, vertically or being rotated by a random angle within $(-20°, 20°)$. The probability was set to be 50%.

4.2 Model in Encoder-Decoder Architecture

We propose a learning-based model to identify fingers from the capacitive images. Different from the CNN-based models applied in previous approaches [1, 36], the capacitive images in an action are regarded as a sequence instead of single images. Therefore, we adapted a model of an encoder-decoder architecture to learn the sequential features.

4.2.1 Architecture. The architecture of our model, as shown in Figure 4, consists of two parts: an encoder and a decoder. The CNN-based encoders encode each input image into a latent representation of spatial features. By comparing the performance of different CNN models (including AlexNet [32], VGGNet [57], ResNet [20] and model of previous work [36], etc.) on the dataset, we chose AlexNet as the encoder in our model. To process capacitive image sequences, the RNN (Recurrant Neural Network)-based decoder takes as input the output vectors of latent representations from the encoder, and extracts the sequence features among all frames into hidden layer vectors. We chose LSTM (Long Short-Term Memory) [22] as the decoder. The output vectors of the decoder were passed through a fully connected layer to produce the prediction result.

4.2.2 Training and results. In the training process, we used a batch size of 1024 and updated the weights using the Adam optimizer [30] with a learning rate of 0.001. The number of hidden layers of LSTM decoder is 2, reaching a best converged performance in experiments. Cross-Entropy Loss was used as the loss function.

The mean sequence length of actions in SpeciFingers was about 50 (0.5 second), while the time cost of identification model is about 0.2 second. To balance the prediction latency and data volume, the length of input sequence was set to be 30 in preprocessing, so that the latency is within the mean input duration.

We explored multiple finger combinations to achieve a usable accuracy with a diversity of fingers: (a) $ALL_{L\&R}$: all the ten fingers of both hands, (b) $ALL_{L/R}$: all the five fingers of either hand, (c) $TM^{+}L$: the thumb, little finger, and other fingers in the middle of either hand.

We split both datasets with a participant-wise split of 80% to 20% (16:4) to avoid samples of the same participant being in both training and validation sets. The confusion matrices of each finger combination are shown in

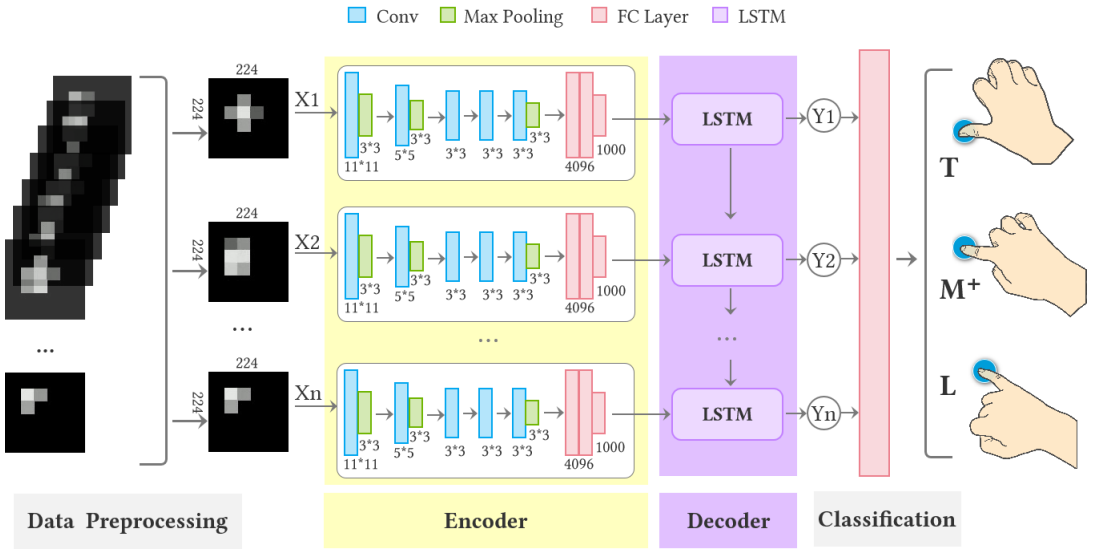
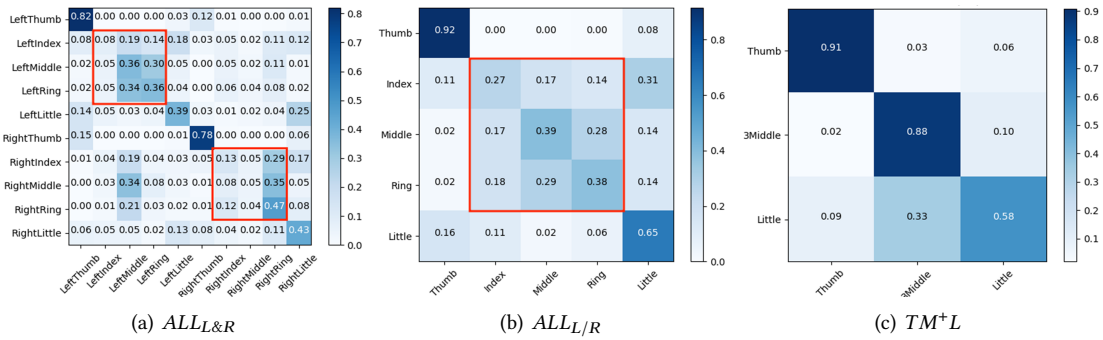


Fig. 4. Architecture of our model. The model takes sequence of in capacitive images as input and predicts the category of the contact finger. The model is an encoder-decoder neural network that learns both spatial and sequential features.

Figure 5. The motivation of combining the three fingers in the middle to one single category (M^+) is that it is difficult to differentiate these three fingers for the model (see the red boxes in Figure 5).



(a) $ALL_{L\&R}$ (b) $ALL_{L/R}$ (c) TM^+L

Fig. 5. Confusion matrices of different finger combinations. The data in red boxes illustrate the classification results of the three fingers in the middle.

The identification accuracies are reported in Table 3, showing a comparison between our model and the previous approach [36] in different finger combinations on both datasets. From the results, we witness higher accuracy of our model than the previous approach in each finger combination. In particular, our model achieved an accuracy of 82.13% in the TM^+L finger combination on our proposed dataset, showcasing a significant improvement of nearly 10% compared to the previous approach's accuracy of 72.21%. We also observed that although the CNN model's performance improved when using the majority voting of the top predictions from the first 30 frames as the final prediction (achieving 76.96% on our dataset), the accuracy was still insufficient. This indicates the

value of the decoder in our model and the trade-off between inference time and improved accuracy. Moreover, when evaluated on the previous CapFingerId dataset [36], our model demonstrated remarkable performance, surpassing the previous approach by more than 12% in accuracy for the same finger combination (78.01% vs. 65.73%). These results highlight the effectiveness of our model, which achieves a state-of-the-art performance on both datasets.

Table 3. The comparison of different identification models: SpeciFingers (ours) and CNN [36]. We explored three finger combinations on two datasets: SpeciFingers (ours) and CapFingerId [36].

Dataset	*SpeciFingers* (ours)			*CapFingerId* [36]		
Fingers	$All_{L\&R}$	$All_{L/R}$	$TM^{+}L$	$All_{L\&R}$	$All_{L/R}$	$TM^{+}L$
CNN [36]	30.71%	46.87%	72.21%	35.55%	46.13%	65.73%
Ours	38.73%	52.46%	**82.13%**	41.60%	51.06%	**78.01%**

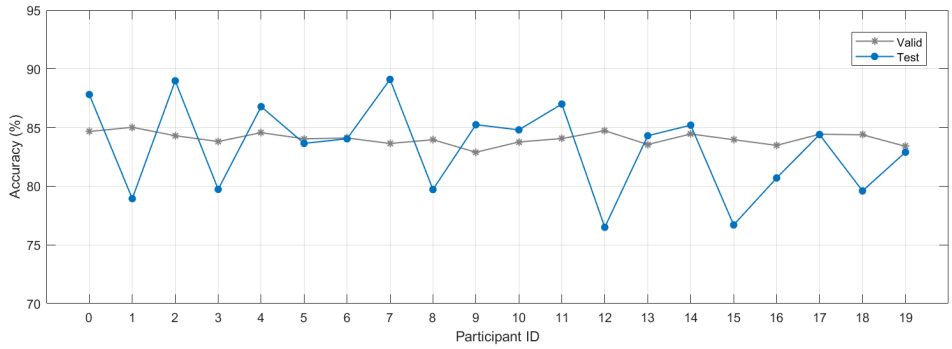


Fig. 6. Training results for each participant. Valid curve means the accuracy in validation set in each 19-participant groups. Test curve means the accuracy in each removed participant.

Although the result of our model could be considered sufficient in general [31] in the $TM^{+}L$ combination, we further investigated the robustness of prediction. We adopted a leave-one-out strategy, i.e. leaving the data of one participant as test set while those of other 19 participants as training set and validation set. The results shown in Figure 6 indicate the variation in accuracy between participants ($M = 83.30\%, Std = 3.84\%$). We speculate that the reason is due to the differences in users' fingers, which lead to the inconsistent features across users.

Regarding the number of frames, we tested its impact on accuracy. Using the same leave-one-out strategy, when using only 10 frames, the model achieved an average accuracy of 78.89% ($Std = 4.61\%$), which is lower compared to the results obtained with 30 frames. Therefore, we believe that maintaining a certain sequence length is necessary to improve accuracy.

4.3 User-specific Strategy

To improve the usability of our model for specific users, we propose a user-specific strategy, in which our model is pre-trained with the data of an amount of users and fine-tuned with the data of a specific user.

To determine the appropriate amount of fine-tuning data, since there is usually a trade-off between data volume and model accuracy, we designed an experiment to compare the performances at different data volumes. We adopted the previous leave-one-out strategy for pre-training. But data of the testing participant was split into

80% for training set and 20% for test set. We used the data in the training set of different proportions (10% - 100%) to fine-tune the pre-trained model and the data in the test set for testing. 5-fold cross-validation was used to ensure the robustness of results.

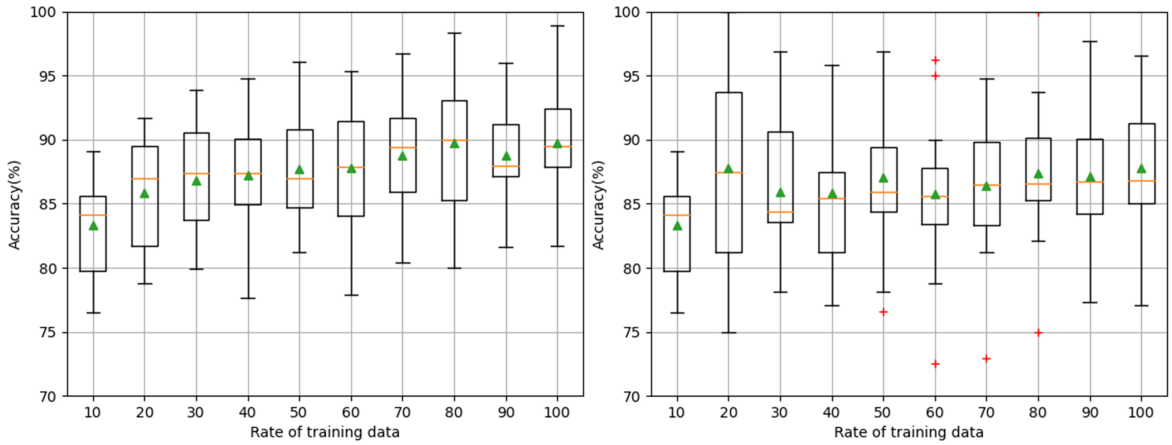


Fig. 7. Box plot of fine-tuning outcome in two different training strategies. (a) Training results using average sampling for sequences with more than 30 frames. (b) Training results using first 30 frames for sequences with more than 30 frames. Green triangles represent average, whereas yellow lines represent median. Red points represent exception values.

Figure 7(a) shows the results of fine-tuned model in testing. The accuracy gradually raises as the amount of data increases. The fine-tuned model obtained a mean accuracy of over 85% in most amount of data, showing the effectiveness of the user-specific strategy in improving model performance. With a balance of data volume and model performance, we recommend a ratio of 30% for fine-tuning, i.e. 12 repetitions per finger for each task in our previous data collection study. Our model attains a mean accuracy of 87.7% on the test set under this amount of fine-tuning data when employing this specific amount of fine-tuning data..

4.4 Implementation

If we sample the entire data of each action for identification, as we previously described, the result cannot be obtained until the action ends. Additionally, there will be a latency between the end of the action and the end of prediction. To address this, in our implementation, we introduced an input duration threshold (30 frames for our apparatus) to separate inputs with shorter or longer durations. When the input length exceeds the threshold, we utilize the first 30 frames for prediction. On the other hand, if the input ends before the threshold, we use the entire sequence for prediction after interpolation. Figure 7(b) presents the testing results of this implementation. While the model's performance, when sampling the first 30 frames, achieves a slightly lower accuracy compared to the average sampling of the entire action data, it is still suitable for use in applications [31]. In practical applications, when using the first 30 frames of the action's data and applying a user-specific strategy for fine-tuning, we achieved an average accuracy of 84.94%. As we increased the amount of fine-tuning data, the average accuracy consistently exceeded 85%.

5 ERROR CORRECTION MECHANISM

Existing research has tried to strike a balance between algorithmic accuracy and interaction variety to ensure that the user's interaction intention is correctly executed. The existing finger identification methods still inevitably suffer from recognition errors in practical use. Users may also make mistakes due to confusing their fingers. Under the existing interaction designs, users have to first undo the incorrect operation and then input again to achieve the expected result, which reduces the efficiency and experience of interaction. Hence, we are motivated to promote a novel design to bring the interaction flow into the right track as expected.

In this section, we introduce our design and implementation of correction mechanism for finger identification. The evaluation results of the mechanism are reported and discussed in section 7.

It is worth mentioning that, the correction mechanism is suitable not only for the model we propose, but also for correcting identification errors for more fingers (i.e. identifying ten fingers or five fingers). The accuracy of different identification models will make a difference to the probability of triggering the correction mechanism.

5.1 Design

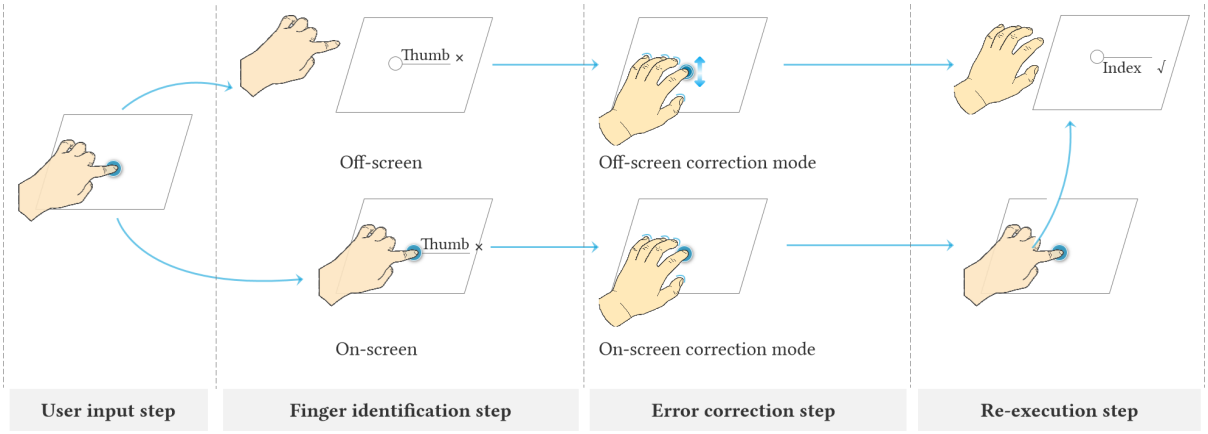


Fig. 8. The pipeline of the proposed identification correction mechanism.

The purpose of the promoted mechanism is to provide a convenient and flexible process for users to easily convey the actual intention and thus correct the interaction when the result of finger identification is not as expected. It is annoying for the user when the result in correction is wrong again, so the requirement to ensure that the finger is identified correctly in the correction mechanism is essential. We attempt to minimize the uncertainty of the correction results while reducing the complexity of interactions in correction.

In our design, the correction mode is activated by simultaneously pressing five fingers openly on the screen, so that a gesture with all five fingers assists in performing accurate finger differentiation while avoiding unintentional touches. If the user drops five fingers straight onto the screen after the previous gesture without lifting the hand, the system can correct the previous identification result by determining the fingers through finger tracking. Otherwise, if the user has lifted the hand after the previous gesture, the finger(s) previously used can be specified by tapping. After the correction, the user continues the interaction or just leaves the screens for the system to re-execute the interaction again with the corrected result.

While previous work has addressed the concept of recognizing all fingers placed on the screen [2], here we emphasize the proposal of an error correction process that serves the complete finger recognition interaction. As demonstrated in Figure 8, the touch input process with correction mechanism consists of four continuous steps:

(1) *User Input step:* The user inputs on the touchscreen with finger(s). The system receives the touch input and responds to it.
(2) *Finger identification step:* The user lifts the finger(s) for short actions like tap and slide or keeps it on the touchscreen for longer actions like press and drag as needed. The system identifies the input finger as long as the raw data is enough for classification. The result will be displayed visually and possible operation will be executed.
(3) *Error correction step:* The user puts all five fingers on the touchscreen to activate the correction mode. If the user has not lifted the finger, the system will trigger **on-screen correction mode** and automatically correct the result. Otherwise, **off-screen correction mode** will be triggered and the user should tap once to indicate the previous input finger. The system modifies the identification result and records the error.
(4) *Re-execution step:* The user continues interacting with the correct identification result. Or if the user simply lifts the fingers after the correction, the system executes the operation again.

5.2 Implementation

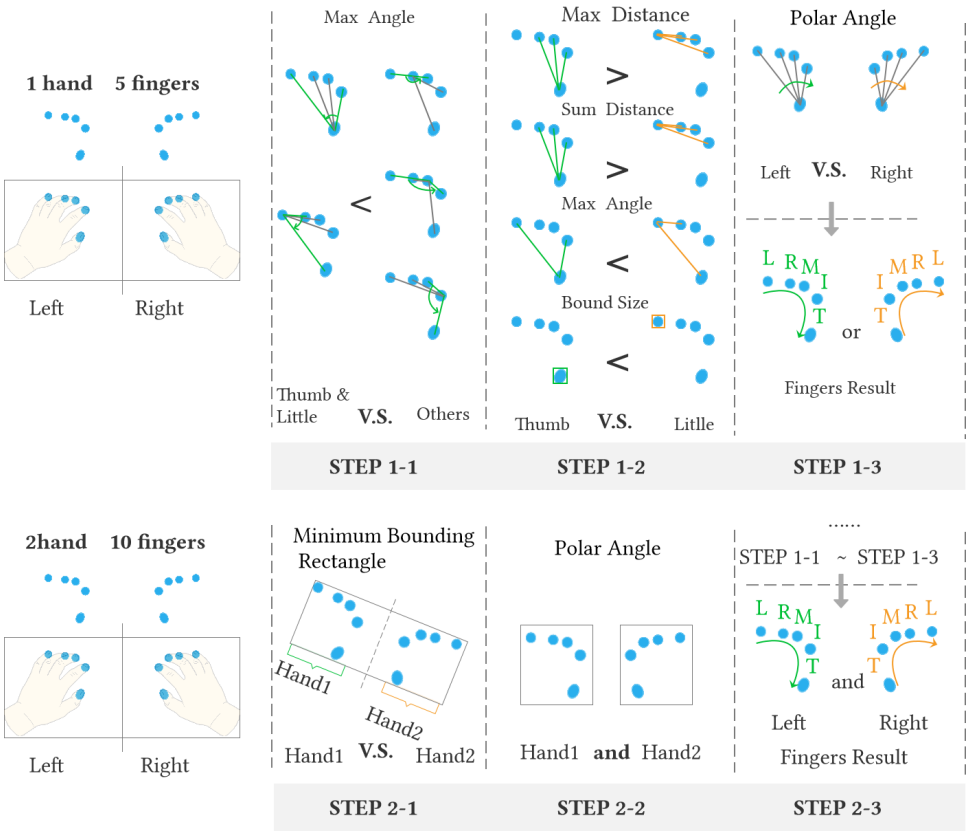


Fig. 9. Procedure for identifying each finger in the correction mode: single hand correction mode (above), double hands correction mode (below).

We improved the method of registering fingers proposed by Au and Tai [2] and designed a pipeline to distinguish between five or ten fingers in the correction mode as shown in Figure 9. Correction mode is triggered by the count of contacting fingers and the threshold value of sensitivity. When there exist 5 or 10 contacting fingers, the corresponding correction mode is activated: 5 fingers for single hand correction mode, 10 fingers for double hands correction mode.

Users are instructed to naturally place their hands on the screen with their fingers comfortably spread apart, avoiding fingers being positioned too closely together or deliberately placed in unnatural positions. This approach helps prevent finger tracking failures caused by inseparable fingers being positioned too closely together and ensures accurate finger distinguishing by avoiding unnatural finger placements.

The algorithm pipeline of single hand correction mode:

Step 1-1: *Separate the thumb, little finger and the other three fingers.* For each finger $F_i \in F$, the maximum angle of F_i and any other two fingers with F_i as the vertex $\theta_{F_i} = max\{\angle(\overrightarrow{F_iF_m}, \overrightarrow{F_iF_n})\}$ where $F_m, F_n \in F \setminus \{F_i\}$ are any two fingers of the other four fingers except F_i. We found by observation that θ_{F_T} and θ_{F_L} are larger than those of the other three fingers (θ_{F_I}, θ_{F_M}, and θ_{F_R}). So we can separate the thumb F_T and little finger F_L from the other fingers by simply sorting θ_{F_i}.

Step 1-2: *Distinguish between thumb and little finger.* We consider the following differences between thumb F_T and little finger F_L ($F_i \in F_T, F_L$): ① the maximum distance between F_i and the other three fingers: $m_{F_i} = max\{|\overrightarrow{F_iF_I}|, |\overrightarrow{F_iF_M}|, |\overrightarrow{F_iF_R}|\}$, usually $m_{F_T} > m_{F_L}$; ② the sum of distances between F_i and the other three fingers: $d_{F_i} = |\overrightarrow{F_iF_I}| + |\overrightarrow{F_iF_M}| + |\overrightarrow{F_iF_R}|$, usually $d_{F_T} > d_{F_L}$; ③ the maximum angle of F_i and any other two fingers with F_i as the vertex: θ_{F_i}, usually $\theta_{F_T} < \theta_{F_L}$; ④ the area of minimum bounding rectangle of F_i: $s_{F_i} = BW_{F_i} \times BH_{F_i}$, usually $s_{F_T} < s_{F_L}$. The differences are weighted for comparison based on trial to distinguish the two fingers.

Step 1-3: *Differentiate between right or left hand.* The thumb F_T is used as the pole to sort the polar angles of fingers in F. The finger order of right hand should be F_I, F_M, F_R, F_L, and F_T while that of left hand should be just reverse as F_L, F_R, F_M, F_I, and F_T. Different results after sorting indicate whether the hand is left or right.

Step 1-4: *Identify each of the five fingers.* With the result of sorting in *Step 1-3*, we can assign a specific finger to each finger in F_I, F_M, and F_R.

The algorithm pipeline of double hands correction mode:

Step 2-1: *Separate into two hands.* We first calculate the minimum bounding rectangle of the fingers F, and then derive the vector of long edges of the bounding rectangle. The projections from the position vectors of the each finger point to the vector of each long edge are sorted, and the first five projections are considered as belonging to one hand and the other five are considered as belonging to the other hand.

Step 2-2: *Identify specific fingers.* For each hand separated, we perform *Step 1-1* to *Step 1-4* respectively to identify the specific fingers of each hand.

5.3 Overall Interaction Process

In Figure 10, we demonstrate the overall process of interaction with finger identification and error correction. We divide the input actions into two categories according to the duration of the input:

Inputs with shorter duration (tapping and sliding): The user inputs end within the input duration threshold ($T_0 - T_1$). After the finger leaves the screen, the system performs finger identification ($T_1 - T_2$). According to the result, the system displays visual cues and executes the function based on the identification result (T_2). Noticing the inconsistencies with expectation, the user activates the off-screen correction mode (T_3). The system identifies the fingers in correction mode ($T_3 - T_4$) and prompts the user with the corrected result (T_4). When the corrected identification result is consistent with the intended finger, the system re-executes the previous function after the user ends the input (T_5).

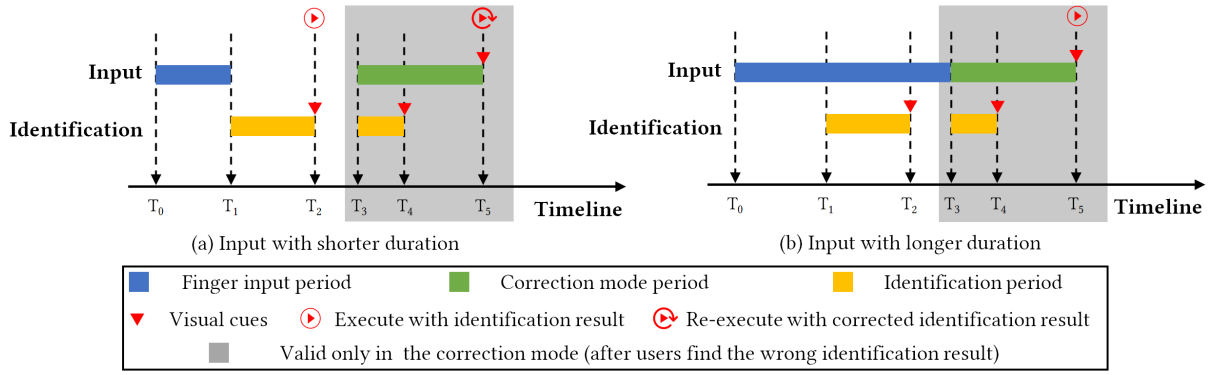


Fig. 10. The overall interaction process with error correction. The inputs are divided into shorter actions and longer actions.

Inputs with longer duration (pressing and dragging): The user starts input at T_0. When the input duration threshold is reached (T_1), the system performs finger identification ($T_1 - T_2$). The system displays visual cues of the result (T_2). The user activates the on-screen correction mode when the result is not as expected, otherwise simply ends the input to execute the function (T_3). The system identifies the fingers in correction mode ($T_3 - T_4$) and prompts the user with the corrected result (T_4). When the corrected identification result is consistent with the intended finger, the system executes the intended function after the user ends the input (T_5). If the user realizes the misidentification result after leaving the screen, off-screen correction mode is also available.

6 FINGER-SPECIFIC INTERACTION

In this section, we present a design space for finger-specific interactions, attempting to distill the existing interaction techniques and inspire more. Under this framework, we propose a novel interaction technique as a use case - using finger-layout binding to optimize the performance of pointing on small targets.

6.1 Design Space of Finger-specific Interactions

We propose the interaction design space to distill existing interaction techniques and to inspire more finger-specific interactions in a more systematic way. We consider the interaction design space through the perspective of bimanual interaction. There are two key interaction design considerations under the interaction technique of differentiated input fingers:

In performing interaction tasks on an interface, finger input can be considered as **explicit or implicit** (i.e., foreground / background framework proposed by Buxton [6]) depending on the user's degree of attention focused on the input. When the user performs a foreground task with their finger, such as dragging an object, the input occupying the primary attention is regarded as explicit. While implicit input occurs when it is in the background of the user's attention, usually as an auxiliary input like pressing to switch the mode of finger or pen input in bimanual interactions.

Another consideration is whether the interaction is **targeted or target-less**. Targeted interactions require users to input in a specified area or on an object of the interface, whereas target-less interactions allow users to interact in any area of the interface.

Figure 11 shows the design space with interaction techniques under the two considerations. It is worth noting that the two considerations are of matters of degrees instead of a strict dichotomy.

We now elaborate on several example interaction techniques from the framework of design space.

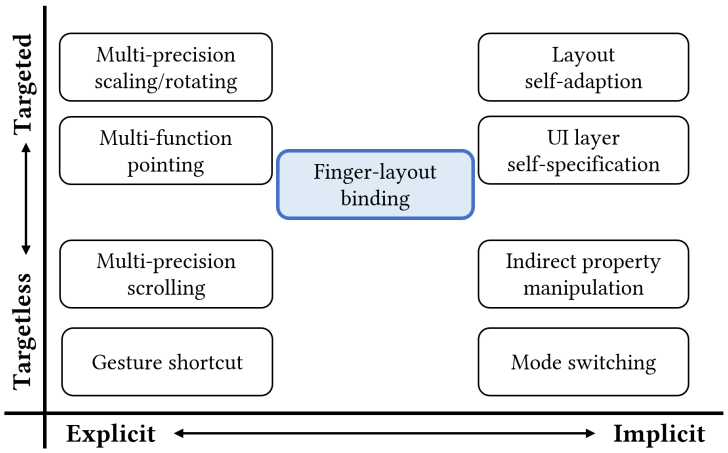


Fig. 11. The design space of finger-specific interactions and example interaction techniques.

Multi-function direct pointing (explicit + targeted) enables different fingers pointing on the same target to activate different functions. Similar interactions have been widely proposed in previous work, such as multiplexed buttons [62], text entry on touchscreen keyboards [12, 17, 35], shortcuts on physical keyboards [68], etc. This interaction technique makes better use of limited screen size or physical buttons.

Multi-precision indirect pointing (explicit + targeted) enables different fingers to control the movement of pointer. The precision of fingers can be achieved by setting the fingers to different cursor speeds, or to relative or absolute movement (similar to the idea of WhichFingers [39]). This interaction technique alleviates the inefficiencies of moving the cursor over a large screen size while maximizing the accuracy of pointing.

Multi-scale zooming/rotating (explicit + targeted) enables different fingers to manipulate with different scales. For example, we use the thumb and index finger for equal scale manipulation while the thumb and little finger for larger or smaller scale manipulation. This interaction technique can be utilized to overcome the difficulty of rotating and zooming in a wide range due to the limited range of finger movement.

Multi-speed scrolling (explicit + targetless) allow users to scroll pages in different speeds with different fingers. For example, scrolling with the thumb or little finger jumps to the end or the beginning of documents and the other three fingers correspond different speeds like normal, faster and slower.

Gesture shortcut (explicit + targetless) defines different shortcut functions for the same gesture with different fingers, thus expanding the gesture vocabulary.

Interface layer specification (implicit + targeted) enables users to input with different fingers corresponding to different layers, such as porous application layers [16], layers in design or drawing applications, etc. This interaction technique improves the efficiency of frequently switching between layers.

Layout self-adaption (implicit + targeted) allows the interface to automatically adjust to the size of the input finger. For example, when thumb input is recognized, the interface elements become larger and more sparse to fit the size of the thumb. This interaction technique provides more accurate and comfortable interaction with different fingers on the interface.

Indirect parameter manipulation (implicit + targetless) allows different fingers to manipulate the parameters of different selected targets or different parameters of the same selected target. For example, users can change the transparency and thickness of brush in a painting application with the thumb and index finger respectively.

Mode switching (implicit + targetless) allows users to change input modes of another input (like finger or pen input in bimanual interaction) with different fingers, such as holding with the thumb to trigger the eraser mode and the index finger to trigger the highlighter mode. This interaction technique accelerates the switching between frequently used modes and reduces the repetitive operations of switching back and forth.

6.2 Use Case: Pointing on small targets with finger-layout binding

Fat finger problem [56] is an essential consideration in the interface design of touchscreen devices. The minimum size of touch targets is recommended in the guidelines of interface design: 44×44pt in Apple's Human Interface Guidelines [26] and 48×48dp in Android Material Design Guidelines [27]. Under such design principles, more screen space is occupied to ensure the accessibility of the interactive targets. The limited elements caused by the small screen size of mobile devices have to be offset by higher frequency of interaction. Additionally, elements on tabletop devices may not be specifically optimized for touch interactions, which can present further challenges in their usage. We hope that the proposed design can also contribute to improving the usability of tabletop device interfaces.

By assigning different functions to different fingers on one UI element, the elements are multiplexed to simplify the interface. Unlike existing work where fingers and functions are bound to a single UI element, we propose to **bind specific fingers to interface layout**, so that multiple elements in an interface layer are mapped to a single finger. The elements in the interface layer should have the same functional implications, such as menu bar and navigation bar. Thus, interfaces that take advantage of this interaction technique can be made more compact between interface layers, which only need to be distinguished by different fingers, while elements within interface layers should be sparser to reduce input errors.

We show the application demonstrations in Figure 12: tab bar of web browsers and menu bar of general applications. In the web browser example, user can hit the tab bar by tapping on the corresponding area with the thumb, and the address bar by tapping with the index finger. In the menu bar example, user can hit the menu bar by pressing on the corresponding area with the thumb, and the shortcuts by tapping with the index finger. With finger-layout binding, these areas in applications can be designed with smaller heights, thus saving screen space.

7 EVALUATION STUDY

We conducted the evaluation study (1) to evaluate the proposed finger identification and error correction pipeline, and (2) to evaluate the use case with users. The evaluation study is conducted on the use case scenario, where the primary task for participants is to point on small targets.

The action of clicking smaller targets may differ from the data collected during normal target clicks in our dataset. By designing experiments that incorporate these diverse interaction scenarios, we effectively test the efficacy of our complete pipeline for finger identification and correction. This enables us to assess the pipeline's effectiveness in challenging interaction contexts, gather insights into its performance under realistic conditions, and identify areas for improvement.

7.1 Study Design

The evaluation study is divided into two parts: data collection, and evaluation experiment.

Data collection: We adopted a user-specific strategy to improve the accuracy of identification model on individual participants. So we collected touch raw data as in our data collection study. Fewer task repetitions for each finger are required, leading to 10 FINGERS × 4 TASKS × 12 REPETITIONS = 480 TRIALS per participant.

Evaluation experiment: We design the interaction tasks of the experiment in the context of our proposed use case. The participants point at specific blocks by tapping or pressing as required. The blocks are organized in 3 rows × 5 columns. Each row corresponds to an interface layer, named as A, B, and C, while the columns are

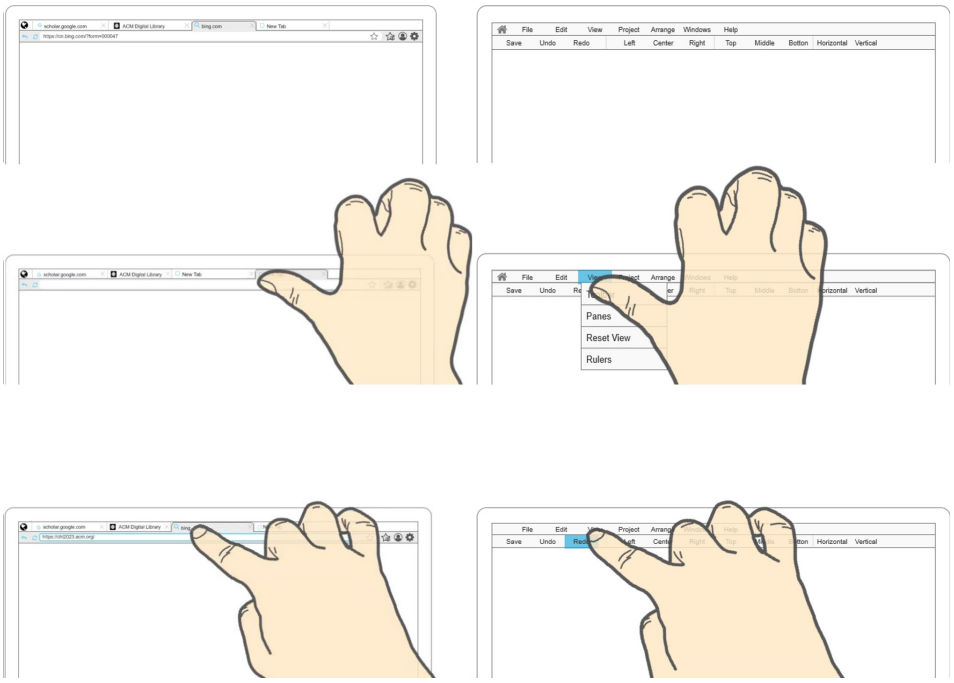


Fig. 12. Application demonstration of pointing small targets with finger-layout binding in web browser (left) and menu bar (right).

named from 1 to 5. The positions and layout of the target blocks on the interface is shown in Figure 13(D). We set the total height of the input area to be close to the recommended value in the design guidelines, which shows the spatial differences between the two designs. The target block of each trial is indicated on the interface with its name, like A1, B2, etc. The order of the two actions is randomly shuffled. For both actions, each block is required to be pointed at for 4 times. Therefore, there are 15 BLOCKS × 2 ACTIONS × 4 REPETITIONS = 120 TRIALS per participant. Notice that the target block in each trial is randomly selected.

We used a one-factor within-subject design. The independent factor was pointing targets with/without finger identification (whether finger-layout binding is available). The study consisted of two sessions: one session with finger identification (namely *with* session) and one session without finger identification (namely *without* session). The order of the two sessions was counterbalanced. Both sessions share the same interaction task design and interface appearance but differ in the way of input and error correction.

In the *with* session, fingers are bound to different interface layers: the thumb, index finger and little finger correspond to A, B and C respectively. Participants should point within the column's boundary where the target block is located. When the pointing position falls outside the boundary, participants were asked to tap the "undo" button (Figure 13 (C)) and repeat this trial. When the interface mistakes the interface layer due to the misidentification result, participants were asked to correct the trial with our correction mechanism.

In the *without* session, there is no restrictions on which hand or finger is used to perform the task. Participants should point within the boundary of the target block. When the pointing location falls outside the boundary, participants were asked to tap on the "undo" button (Figure 13 (C)) and repeat this trial.

At the end of each session, participants scored their subjective experience with a questionnaire, including (1) cognitive load from NASA-TLX [18], (2) demand for visual attention, (3) efficiency and learnability from Perceived Usefulness and Ease of Use Questionnaire [10], (4) capability of error prevention and error recovery from Nielsen's Heuristic Evaluation [45], (5) intuitiveness of interaction, (6) overall feeling. All questions are rated on 7-point Likert Scale (1: Strongly negative - 7: Strongly positive).

7.2 Apparatus

The prototype for evaluation was implemented in a front-end and back-end architecture. The back-end server was running the identification model on a GeForce GTX 2080Ti GPU and providing web service with Python Flask[3]. The front-end was running on a PC with an Intel i7-3770 CPU and 16GB RAM, connected with a Wacom touchscreen (the same device as in the data collection study), a Tobii Eye Tracker 5[4], and a keypad. The front-end provides two main functions: running the interface for evaluation and capturing eye movement data. Front-end and back-end devices were connected by a LAN.

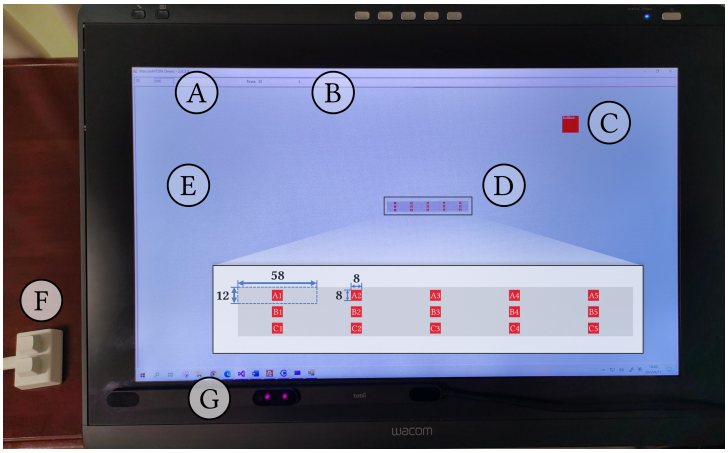


Fig. 13. The interface for the evaluation study consists of a text box (A) to enter participant ID, a text label (B) to indicate current action and target block, undo button (C), input area (D) with target blocks, background (E) showing the status of eye tracker, keypad (F) to switch tasks, and eye tracker (G) to collect eyegaze data.

Figure 13 demonstrates the interface for evaluation and the external devices connected to the front-end. For right-handed participants, the input area (Figure 13 (D)) was located in the slightly right of center to avoid blocking the eye tracker which was installed in the lower left corner of the screen. The background color of the interface (Figure 13 (E)) indicated the detection status of eye movement data: blue for successful and pink for failure. The status was synchronized between a program for fetching eye movement data and the interface by a local Redis[5] server. The eye movement data was logged every 30ms.

In our apparatus interface design, we set each interaction layer to be 12px in height. The operational area, identified as Figure 13 (D), spans approximately 36px in total height. Within this region, we organized three interface layers (rows), each measuring 12px in height and accommodating five blocks per layer. There's a vertical spacing of 4px and a horizontal spacing of 25px between these blocks to ensure that the horizontal gap doesn't

[3] https://flask.palletsprojects.com/en/2.2.x/
[4] https://gaming.tobii.com/product/eye-tracker-5/
[5] https://redis.io/

interfere with the interactions. The design is aimed at highlighting the distinctions in performance and user experience between finger-specific interaction and standard interaction within smaller touch areas. It also serves as an evaluation of the "finger-layout binding" design principle.

7.3 Participants & Procedure

We recruited 16 participants (9 male and 7 female) with an average age of 25 ($Std = 4.41$). None of them participated in the previous data collection. All participants were right-handed. The average hand size ranged from $16cm$ to $20.3cm$ ($M = 17.83cm, Std = 1.22$).

The experiment took place in a separate room that was free from distractions. Participants sat in a chair, which was adjusted to a comfortable height for each participant. The touchscreen was set up on a stable desk.

We first obtained the informed consent from the participants, and then introduced the background and procedure to them. We collected their personal information and calibrated the eye tracker.

Afterwards, the participants completed the tasks in each session. Before each session's experiment began, they were instructed how to complete the task and were given 5 minutes for practice. They took a 5-minute break between sessions and were allowed to rest at any time during a session. After completing each session, participants were asked to answer the questionnaire for subjective ratings. At the end of the study, the participants were briefly interviewed for their subjective feedback.

7.4 Results and Discussion

7.4.1 Accuracy of general identification pipeline. We report a mean accuracy of 99.45% of finger identification result. The result confirmed the effectiveness and reliability of the proposed pipeline with finger identification and error correction. The reason for the few unsuccessful corrections is that the participant's fingers were in particular positions, causing the algorithm to mistakenly reverse the thumb and little finger. The probability of triggering the finger correction mechanism during our experiments was 37.64% ($Std = 13.30\%$), indicating a decrease in the model's recognition accuracy compared to the result obtained from the dataset. We attribute this difference to potential variations in the participants' clicking behavior when clicking smaller targets in our task compared to the data collection process. This finding aligns with the objective of our experimental design, demonstrating that our finger identification and correction pipeline remains useful even with relatively lower finger recognition accuracy. It opens up possibilities for users to engage in a wider range of interactions. We also believe that it is necessary to collect finger contact data from a broader range of scenarios to enhance the generalizability of the identification model. This will reduce the probability of triggering finger correction and further improve the efficiency of interactions.

7.4.2 Visual attention. Since it is not necessary to precisely point on the block when using finger-layout binding, we hypothesize that the participants use less visual attention to complete the tasks in the *with* session. We employed an eye tracker to obtain the eye movement data. We focus on the interval from the moment the participant received the prompt (starting of the task) to the first frame of the first pointing action (starting of the input). During this interval, the participant went through the process from reading the prompt, then locating the target block and proceeding to point on the block. We measure the visual attention by calculating the ratio of the duration of eye in input area (Figure 13 (D)) to total duration. The assumption of homogeneity of variance cannot be rejected by Levene's test ($F_{1,2812} = 1.280, p = 0.258 > 0.05$). One-way ANOVA showed that the visual attention variation in different sessions was significant ($F = 35.365, p < 0.0001$). The result proves our hypothesis and shows the benefits of our finger-layout binding in finger-specific interaction.

7.4.3 Pointing error. Due to the different size of target area for pointing, we built the hypothesis that there is less pointing error in the *with* session than in the *without* session. We asked participants to recover from

a pointing error by undo-redo method in *without* session or by error correction mechanism in *with* session. Therefore, the total number of all touch inputs while completing a task can be used to measure pointing errors, i.e. how many trials in total to reach the goal. The mean trial is 1.46 ($Std = 0.792$) in the *with* session and 1.60 ($Std = 1.281$) in the *without* session. The assumption of homogeneity of variance is rejected by Levene's test ($F_{1,3599} = 4.474, p < 0.001$). The result of Welch test ($t_{1,3215.095} = 15.684, p < 0.001$) and Brown-Forsythe test ($t_{1,3215.095} = 15.684, p < 0.001$) both indicates the significant difference between the *with* session and *without* session in error rate. The result proves the hypothesis and indicates the advantage of finger-layout binding in reducing pointing errors.

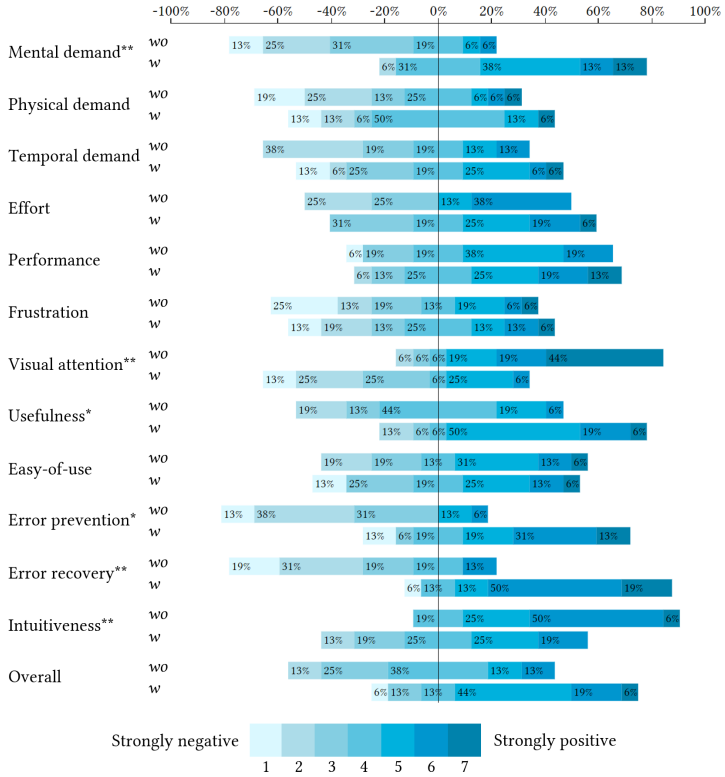


Fig. 14. Result of subjective ratings. "*w*" indicates the *with* session and "*wo*" indicates the *without* sessions in the experiment. * means significant ($p < 0.05$) and ** means very significant ($p < 0.01$).

7.4.4 Subjective ratings. We ran Wilcoxon signed-rank tests on the subjective rating results (as shown in Figure 14). The ratings of visual attention shows that participants perceived very significantly lower visual attention cost in the *with* session than in the *without* session ($Z = -3.050, p = 0.002$). The ratings of error recovery indicates that participants agreed that our error correction mechanism in the *with* session was very significantly more helpful than the traditional "undo-redo method in the *without* session ($Z = -3.213, p = 0.001$). Several participants stated that they were "very willing to use such a method without having to repeatedly perform undo and redo". The scores of error prevention shows that participants experienced the significantly higher effectiveness of finger-layout binding in reducing input errors ($Z = -2.471, p = 0.013$). The scores of

usefulness indicates the significantly higher perceived usefulness of finger-layout binding in the *with* session ($Z = -2.137, p = 0.033$). However, participants gave negative feedback to the *with* session on two metrics: mental demand ($Z = -2.944, p = 0.003$) and intuitiveness ($Z = -2.961, p = 0.003$). Several participants noted the need to recall the correspondences of specific fingers to the layout by memory, which could be effectively improved by hinting the user on the interface. Participants also stated that the finger-layout binding takes longer to learn, but better performance can be achieved with a little more practice. There is no significance in other metrics of subjective ratings.

8 DISCUSSION

8.1 Improving Identification Performance

In the dataset-based evaluation, our model achieved an accuracy over 85%, nearly 10% increase than the current baseline model, which could be considered sufficient in general [31]. Our model was available under natural interaction actions, including tapping, pressing, sliding and dragging.

However, the data collected in advance only covers selected typical scenarios. During our experiments, we observed a decrease in recognition accuracy specifically for clicking small targets. Therefore, to improve the learning-based model, it is crucial to expand the dataset by including data from a wider range of diverse interactions in daily usage. The proposed error correction method can be used to record the misidentification data with their real finger labels to iteratively optimize the model. Additionally, while our current data includes participants with average hand sizes, it is crucial to enhance data diversity for the model's effectiveness in a broader population. Hand size variations can influence capacitive contact characteristics, thereby affecting the model's performance. Expanding the dataset to encompass a wider range of hand sizes can improve the model's robustness and adaptability across different individuals.

The main challenge for the identification model is the low resolution of the capacitive images. With the advances in deep learning, the performance of our model could be improved by new models. Strategies such as few-shot learning can be used to make the best use of the limited data available. To overcome the difference of fingers between users, modules like domain discriminator can be used to modify the feature representation space and generate domain-invariant features. Super-resolution of capacitive images in existing work [42, 58] can be applied to reconstruct the low-resolution capacitive images to achieve better performance. Data from other sources of input (e.g., IMUs, microphones, vibration sensors, etc.) can be used to complement the insufficient signals in capacitive raw data for increasing the identification accuracy.

Identifying specific fingers in a multi-finger gesture is also a topic that worth further consideration. Although existing method [1] enables the classification of multi-finger gesture sets with different fingers, they do not identify specific fingers in the gesture set. Utilizing the relationship between the states of multiple fingers (position, orientation, etc.) might improve the classification of individual fingers. Our current work primarily focuses on single-finger interaction gestures. Therefore, further validation is necessary to evaluate the performance of our model in handling multi-touch gestures. Nonetheless, our workflow incorporates finger contact area segmentation and tracking methods, making it potentially applicable for multi-finger recognition. In addition, aside from simultaneous multi-finger interaction, there may be variations in interaction data when multiple fingers interchange rapidly in real usage. This is also worth further exploration.

8.2 Refining Error Correction Mechanism

In the evaluation study, we observed a positive acceptance of the error correction mechanism among most participants. Several participants expressed their views on its effectiveness compared to the existing "undo-redo" process, and the quantitative results from the experiment aligned with their opinions. Importantly, our experiments also demonstrated that even when there was a decrease in finger recognition accuracy, the error

correction mechanism was able to make finger recognition interactions usable and effective. However, there is room for further refinement in our error correction mechanism.

One aspect that requires consideration is the requirement for a sufficiently large device screen size to accommodate five spread-out fingers on the screen. It is also worth exploring alternative gestures involving fewer fingers that can robustly identify specific fingers for correcting previous errors. For instance, a gesture like thumb + X could be employed to achieve the same correction task if each finger can be accurately recognized. However, as discussed in the previous section, there is currently no existing method for identifying specific fingers in multi-finger gestures. Furthermore, it is important to acknowledge that our method relies on users naturally placing their fingers on the screen and may still face challenges in distinguishing very close fingers or addressing the issue of "fat fingers". Future work could consider developing more universal and robust finger registration methods to overcome these limitations.

In addition, exploring other implicit correction methods is worthwhile. Such approaches involve utilizing additional signals, such as physiological signals (e.g., EMG, EEG, etc.) or eye gaze, to identify and correct errors. These signals can provide valuable information for improving the accuracy and effectiveness of error correction.

We believe that an error correction mechanism that requires fewer fingers, offers more flexible correction methods, and incorporates more implicit interaction modalities has the potential to expand the capability of finger recognition error recovery to a broader range of ubiquitous devices.

8.3 Potentials in Finger-specific Interaction

We proposed a design space for finger-specific interaction and evaluated the use case with our prototype. The evaluation experiment showed a positive performance and preference in completing the task of pointing on small targets with finger-layout binding. Our experiment could be further improved by, such as changing the size and margin of the targets to compare the performance of pointing in different target conditions.

The interface design for finger-specific interaction is also an area worthy of study. The traditional interface design does not distinguish the correspondence between the layout or UI elements and different fingers. Therefore, users cannot intuitively understand the binding between fingers and functions, which increases cognitive load during operation. When designing the binding of UI elements and layouts with specific fingers, intuitiveness needs to be considered when choosing the finger. For example, in our use case, the correspondence between interface layout and fingers can be designed in reference to the order of finger arrangement and layout.

8.4 Generalizability to Other Devices

In this work, we employed Wacom's capacitive touchscreen and used the raw data it provided to identify the contacting finger. Our work is not restricted to the device we used, and instead can be generalized to other devices providing touch capacitive data like LG Nexus 5 [34, 36, 55], Samsung Galaxy Tab S2 tablet [7, 42], etc. We have shown that our model improved the identification accuracy on the raw data from other capacitive touchscreen devices. Therefore, the feasibility and potential of deploying our model to identify fingers on different devices is obviously promising. We acknowledge that when applying our identification method to other devices, it is important to consider the device-specific characteristics, such as finger angles or device orientations.

The design space of finger-specific interaction is also available for any device that supports finger identification. We believe it will assist designers in creating more interactions with application scenarios on various devices. For devices with smaller screens such as mobile phones and smartwatches, the use case we proposed - pointing at small targets with finger-layout binding - may contribute to better utilization of the very limited screen space.

The correction mechanism proposed in this paper is applicable to devices with larger screens like desktops and tablets, since it requires all five fingers to be placed on the screen. As mentioned in the last section, more explicit or implicit approaches of correcting errors are expected for further exploration.

9 CONCLUSION

In this paper, we investigated finger identification and error correction to enhance finger-specific interactions on capacitive touchscreens. We adopted a neural network of an encoder-decoder architecture to improve the performance leveraging the sequence features in touch inputs. Our model differentiated between thumb, little finger and the other three fingers in a mean accuracy over 85%. To minimize the impact of inevitable misclassification, we proposed an approach to facilitate explicit error correction by users. Moreover, a design space of finger-specific interaction was presented to inspire more novel interactions. In particular, we discovered a novel interaction binding fingers and interface layout to improve the pointing performance on small targets. The evaluation results indicated the benefits of finger-layout binding, together with the performance of our pipeline with finger identification and error correction.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under Grant No.62272447, National Key R&D Program of China under Grant No.2022ZD0117900, Beijing Natural Science Foundation under Grant No.L232028 and No.L222008, Beijing Hospitals Authority Clinical Medicine Development of Special Funding Support under Grant No.ZLRK202330. We express our gratitude to the reviewers for their valuable comments and insights. Our sincere appreciation goes to all the participants who actively contributed to the data collection and user evaluation study. Special thanks to Junjiao Sun, Qiang He, and everyone who dedicated their efforts and offered valuable feedback throughout the entire process.

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* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 997–1009. https://doi.org/10.1145/3472749.3474801

[2] Oscar Kin-Chung Au and Chiew-Lan Tai. 2010. Multitouch finger registration and its applications. In *Proceedings of the 22nd Conference of the Computer-Human Interaction Special Interest Group of Australia on Computer-Human Interaction* (Brisbane, Australia) *(OZCHI '10)*. Association for Computing Machinery, New York, NY, USA, 41–48. https://doi.org/10.1145/1952222.1952233

[3] Vincent Becker, Pietro Oldrati, Liliana Barrios, and Gábor Sörös. 2018. Touchsense: Classifying Finger Touches and Measuring Their Force with an Electromyography Armband. In *Proceedings of the 2018 ACM International Symposium on Wearable Computers* (Singapore, Singapore) *(ISWC '18)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3267242.3267250

[4] Hrvoje Benko, T. Scott Saponas, Dan Morris, and Desney Tan. 2009. Enhancing Input on and above the Interactive Surface with Muscle Sensing. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces* (Banff, Alberta, Canada) *(ITS '09)*. Association for Computing Machinery, New York, NY, USA, 93–100. https://doi.org/10.1145/1731903.1731924

[5] 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

[6] Bill Buxton. 1995. Integrating the Periphery, Context: A New Model of Telematics. In *Proceedings of Graphics Interface '95* (Quebec, Quebec, Canada) *(GI '95)*. Canadian Human-Computer Communications Society, Toronto, Ontario, Canada, 239–246. https://doi.org/10.20380/GI1995.28

[7] 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* (Toulouse & Virtual, France) *(MobileHCI '21)*. Association for Computing Machinery, New York, NY, USA, Article 3, 13 pages. https://doi.org/10.1145/3447526.3472045

[8] Josh Clark. 2015. *Designing for touch.* A Book Apart New York.

[9] Ashley Colley and Jonna Häkkilä. 2014. Exploring Finger Specific Touch Screen Interaction for Mobile Phone User Interfaces. In *Proceedings of the 26th Australian Computer-Human Interaction Conference on Designing Futures: The Future of Design* (Sydney, New South Wales, Australia) *(OzCHI '14)*. Association for Computing Machinery, New York, NY, USA, 539–548. https://doi.org/10.1145/2686612.2686699

[10] Fred D Davis. 1989. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS quarterly* (1989), 319–340.

[11] Philipp Ewerling, Alexander Kulik, and Bernd Froehlich. 2012. Finger and Hand Detection for Multi-Touch Interfaces Based on Maximally Stable Extremal Regions. In *Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces* (Cambridge, Massachusetts, USA) *(ITS '12)*. Association for Computing Machinery, New York, NY, USA, 173–182. https://doi.org/10.1145/2396636.2396663

[12] Hyunjae Gil, DoYoung Lee, Seunggyu Im, and Ian Oakley. 2017. TriTap: Identifying Finger Touches on Smartwatches. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 3879–3890. https://doi.org/10.1145/3025453.3025561

[13] Alix Goguey, Géry Casiez, Daniel Vogel, Fanny Chevalier, Thomas Pietrzak, and Nicolas Roussel. 2014. A Three-Step Interaction Pattern for Improving Discoverability in Finger Identification Techniques. In *Proceedings of the Adjunct Publication of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA) *(UIST'14 Adjunct)*. Association for Computing Machinery, New York, NY, USA, 33–34. https://doi.org/10.1145/2658779.2659100

[14] Alix Goguey, Daniel Vogel, Fanny Chevalier, Thomas Pietrzak, Nicolas Roussel, and Géry Casiez. 2017. Leveraging finger identification to integrate multi-touch command selection and parameter manipulation. *International Journal of Human Computer Studies* 99 (1 March 2017), 21–36. https://doi.org/10.1016/j.ijhcs.2016.11.002 Publisher Copyright: © 2016 Elsevier Ltd.

[15] Mitchell Gordon, Tom Ouyang, and Shumin Zhai. 2016. WatchWriter: Tap and Gesture Typing on a Smartwatch Miniature Keyboard with Statistical Decoding. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 3817–3821. https://doi.org/10.1145/2858036.2858242

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

[17] Aakar Gupta and Ravin Balakrishnan. 2016. *DualKey: Miniature Screen Text Entry via Finger Identification.* Association for Computing Machinery, New York, NY, USA, 59–70. https://doi.org/10.1145/2858036.2858052

[18] Sandra G. Hart and Lowell E. Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Human Mental Workload*, Peter A. Hancock and Najmedin Meshkati (Eds.). Advances in Psychology, Vol. 52. North-Holland, 139–183. https://doi.org/10.1016/S0166-4115(08)62386-9

[19] Ke He, Yongjie Duan, Jianjiang Feng, and Jie Zhou. 2022. Estimating 3D Finger Angle via Fingerprint Image. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 1, Article 14 (mar 2022), 22 pages. https://doi.org/10.1145/3517243

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778. https://doi.org/10.1109/CVPR.2016.90

[21] Ken Hinckley, Seongkook Heo, Michel Pahud, Christian Holz, Hrvoje Benko, Abigail Sellen, Richard Banks, Kenton O'Hara, Gavin Smyth, and William Buxton. 2016. Pre-Touch Sensing for Mobile Interaction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 2869–2881. https://doi.org/10.1145/2858036.2858095

[22] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[23] Da-Yuan Huang, Ming-Chang Tsai, Ying-Chao Tung, Min-Lun Tsai, Yen-Ting Yeh, Liwei Chan, Yi-Ping Hung, and Mike Y. Chen. 2014. TouchSense: Expanding Touchscreen Input Vocabulary Using Different Areas of Users' Finger Pads. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 189–192. https://doi.org/10.1145/2556288.2557258

[24] David Huggins-Daines and Alexander I. Rudnicky. 2008. Interactive ASR Error Correction for Touchscreen Devices. In *Proceedings of the ACL-08: HLT Demo Session*, Jimmy Lin (Ed.). Association for Computational Linguistics, Columbus, Ohio, 17–19. https://aclanthology.org/P08-4005

[25] Kaori Ikematsu and Shota Yamanaka. 2020. ScraTouch: Extending Interaction Technique Using Fingernail on Unmodified Capacitive Touch Surfaces. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 3, Article 81 (sep 2020), 19 pages. https://doi.org/10.1145/3411831

[26] Apple Inc. 2022. Human interface guidelines: Layout. https://developer.apple.com/design/human-interface-guidelines/foundations/layout/. Accessed August 27, 2022.

[27] Google Inc. 2022. Material Design Guidelines: Accessibility. https://material.io/design/usability/accessibility.html#layout-and-typography. Accessed August 27, 2022.

[28] Kihyo Jung and Jinah Jang. 2015. Development of a two-step touch method for website navigation on smartphones. *Applied Ergonomics* 48 (2015), 148–153. https://doi.org/10.1016/j.apergo.2014.11.006

[29] Jiwan Kim and Ian Oakley. 2022. SonarID: Using Sonar to Identify Fingers on a Smartwatch. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 287, 10 pages. https://doi.org/10.1145/3491102.3501935

[30] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

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

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).

[33] Anuj Kumar, Tim Paek, and Bongshin Lee. 2012. Voice Typing: A New Speech Interaction Model for Dictation on Touchscreen Devices. 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, 2277–2286. https://doi.org/10.1145/2207676.2208386

[34] Huy Viet Le, Thomas Kosch, Patrick Bader, Sven Mayer, and Niels Henze. 2018. PalmTouch: Using the Palm as an Additional Input Modality on Commodity Smartphones. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3173934

[35] Huy Viet Le, Sven Mayer, and Niels Henze. 2018. InfiniTouch: Finger-Aware Interaction on Fully Touch Sensitive Smartphones. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 779–792. https://doi.org/10.1145/3242587.3242605

[36] 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* (Marina del Ray, California) *(IUI '19)*. Association for Computing Machinery, New York, NY, USA, 637–649. https://doi.org/10.1145/3301275.3302295

[37] Nicolai Marquardt, Johannes Kiemer, and Saul Greenberg. 2010. What Caused That Touch? Expressive Interaction with a Surface through Fiduciary-Tagged Gloves. In *ACM International Conference on Interactive Tabletops and Surfaces* (Saarbrücken, Germany) *(ITS '10)*. Association for Computing Machinery, New York, NY, USA, 139–142. https://doi.org/10.1145/1936652.1936680

[38] Nicolai Marquardt, Johannes Kiemer, David Ledo, Sebastian Boring, and Saul Greenberg. 2011. Designing User-, Hand-, and Handpart-Aware Tabletop Interactions with the TouchID Toolkit. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces* (Kobe, Japan) *(ITS '11)*. Association for Computing Machinery, New York, NY, USA, 21–30. https://doi.org/10.1145/2076354.2076358

[39] Damien Masson, Alix Goguey, Sylvain Malacria, and Géry Casiez. 2017. WhichFingers: Identifying Fingers on Touch Surfaces and Keyboards Using Vibration Sensors. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 41–48. https://doi.org/10.1145/3126594.3126619

[40] Fabrice Matulic, Daniel Vogel, and Raimund Dachselt. 2017. Hand Contact Shape Recognition for Posture-Based Tabletop Widgets and Interaction. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces* (Brighton, United Kingdom) *(ISS '17)*. Association for Computing Machinery, New York, NY, USA, 3–11. https://doi.org/10.1145/3132272.3134126

[41] 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* (Brighton, United Kingdom) *(ISS '17)*. Association for Computing Machinery, New York, NY, USA, 220–229. https://doi.org/10.1145/3132272.3134130

[42] 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* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 12, 10 pages. https://doi.org/10.1145/3411764.3445703

[43] Manuel Meier, Paul Streli, Andreas Fender, and Christian Holz. 2021. TapID: Rapid Touch Interaction in Virtual Reality using Wearable Sensing. In *2021 IEEE Virtual Reality and 3D User Interfaces (VR)*. 519–528. https://doi.org/10.1109/VR50410.2021.00076

[44] Sundar Murugappan, Vinayak, Niklas Elmqvist, and Karthik Ramani. 2012. *Extended Multitouch: Recovering Touch Posture and Differentiating Users Using a Depth Camera.* Association for Computing Machinery, New York, NY, USA, 487–496. https://doi.org/10.1145/2380116.2380177

[45] Jakob Nielsen. 1994. Usability inspection methods. In *Conference companion on Human factors in computing systems*. 413–414.

[46] Ian Oakley, Carina Lindahl, Khanh Le, DoYoung Lee, and MD. Rasel Islam. 2016. The Flat Finger: Exploring Area Touches on Smartwatches. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 4238–4249. https://doi.org/10.1145/2858036.2858179

[47] Seungjae Oh, Chaeyong Park, Yo-Seb Jeon, and Seungmoon Choi. 2021. Identifying Contact Fingers on Touch Sensitive Surfaces by Ring-Based Vibratory Communication. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 208–222. https://doi.org/10.1145/3472749.3474745

[48] Keunwoo Park, Daehwa Kim, Seongkook Heo, and Geehyuk Lee. 2020. *MagTouch: Robust Finger Identification for a Smartwatch Using a Magnet Ring and a Built-in Magnetometer.* Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376234

[49] Keunwoo Park, Sunbum Kim, Youngwoo Yoon, Tae-Kyun Kim, and Geehyuk Lee. 2020. *DeepFisheye: Near-Surface Multi-Finger Tracking Technology Using Fisheye Camera.* Association for Computing Machinery, New York, NY, USA, 1132–1146. https://doi.org/10.1145/3379337.3415818

[50] Keunwoo Park and Geehyuk Lee. 2019. FingMag: Finger Identification Method for Smartwatch. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI EA '19)*. Association for Computing Machinery, New

York, NY, USA, 1–6. https://doi.org/10.1145/3290607.3312982
[51] Alan Poston. 2000. Human engineering design data digest. *Washington, DC: Department of Defense Human Factors Engineering Technical Advisory Group* (2000), 90.
[52] Felix Putze, Christoph Amma, and Tanja Schultz. 2015. Design and Evaluation of a Self-Correcting Gesture Interface Based on Error Potentials from EEG. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 3375–3384. https://doi.org/10.1145/2702123.2702184
[53] Felix Putze, Tilman Ihrig, Tanja Schultz, and Wolfgang Stuerzlinger. 2020. *Platform for Studying Self-Repairing Auto-Corrections in Mobile Text Entry Based on Brain Activity, Gaze, and Context.* Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376815
[54] Quentin Roy, Yves Guiard, Gilles Bailly, Éric Lecolinet, and Olivier Rioul. 2022. Glass+Skin: An Empirical Evaluation of the Added Value of Finger Identification to Basic Single-Touch Interaction on Touch Screens. In *Human-Computer Interaction – INTERACT 2015*. Springer-Verlag, Berlin, Heidelberg, 55–71. https://doi.org/10.1007/978-3-319-22723-8_5
[55] 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
[56] Katie A. Siek, Yvonne Rogers, and Kay H. Connelly. 2005. Fat Finger Worries: How Older and Younger Users Physically Interact with PDAs. In *Proceedings of the 2005 IFIP TC13 International Conference on Human-Computer Interaction* (Rome, Italy) *(INTERACT'05)*. Springer-Verlag, Berlin, Heidelberg, 267–280. https://doi.org/10.1007/11555261_24
[57] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1409.1556
[58] 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* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 289, 14 pages. https://doi.org/10.1145/3411764.3445621
[59] Bernhard Suhm, Brad Myers, and Alex Waibel. 2001. Multimodal Error Correction for Speech User Interfaces. *ACM Trans. Comput.-Hum. Interact.* 8, 1 (mar 2001), 60–98. https://doi.org/10.1145/371127.371166
[60] Ryo Takahashi, Masaaki Fukumoto, Changyo Han, Takuya Sasatani, Yoshiaki Narusue, and Yoshihiro Kawahara. 2020. *TelemetRing: A Batteryless and Wireless Ring-Shaped Keyboard Using Passive Inductive Telemetry.* Association for Computing Machinery, New York, NY, USA, 1161–1168. https://doi.org/10.1145/3379337.3415873
[61] Keith Vertanen, Haythem Memmi, Justin Emge, Shyam Reyal, and Per Ola Kristensson. 2015. VelociTap: Investigating Fast Mobile Text Entry using Sentence-Based Decoding of Touchscreen Keyboard Input. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 659–668. https://doi.org/10.1145/2702123.2702135
[62] Jingtao Wang and John Canny. 2004. FingerSense: Augmenting Expressiveness to Physical Pushing Button by Fingertip Identification. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems* (Vienna, Austria) *(CHI EA '04)*. Association for Computing Machinery, New York, NY, USA, 1267–1270. https://doi.org/10.1145/985921.986040
[63] Ruolin Wang, Chun Yu, Xing-Dong Yang, Weijie He, and Yuanchun Shi. 2019. EarTouch: Facilitating Smartphone Use for Visually Impaired People in Mobile and Public Scenarios. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300254
[64] Mark Weiser. 1999. The computer for the 21st century. *ACM SIGMOBILE mobile computing and communications review* 3, 3 (1999), 3–11.
[65] Robert Xiao, Julia Schwarz, and Chris Harrison. 2015. Estimating 3D Finger Angle on Commodity Touchscreens. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces* (Madeira, Portugal) *(ITS '15)*. Association for Computing Machinery, New York, NY, USA, 47–50. https://doi.org/10.1145/2817721.2817737
[66] Mingrui Ray Zhang and Shumin Zhai. 2021. PhraseFlow: Designs and Empirical Studies of Phrase-Level Input. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 197, 13 pages. https://doi.org/10.1145/3411764.3445166
[67] Mingrui Ray Zhang, Shumin Zhai, and Jacob O. Wobbrock. 2022. TypeAnywhere: A QWERTY-Based Text Entry Solution for Ubiquitous Computing. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 339, 16 pages. https://doi.org/10.1145/3491102.3517686
[68] Jingjie Zheng and Daniel Vogel. 2016. Finger-Aware Shortcuts. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI '16)*. Association for Computing Machinery, New York, NY, USA, 4274–4285. https://doi.org/10.1145/2858036.2858355