FingerGlass: Enhancing Smart Glasses Interaction via Fingerprint Sensing

Zhanwei Xu
zw-xu18@mails.tsinghua.edu.cn
Department of Automation, BNRist, Tsinghua University
Beijing, China

Haoxiang Pei
phx19@mails.tsinghua.edu.cn
Department of Automation, BNRist, Tsinghua University
Beijing, China

Jianjiang Feng*
jfeng@tsinghua.edu.cn
Department of Automation, BNRist, Tsinghua University
Beijing, China

Jie Zhou
jzhou@tsinghua.edu.cn
Department of Automation, BNRist, Tsinghua University
Beijing, China

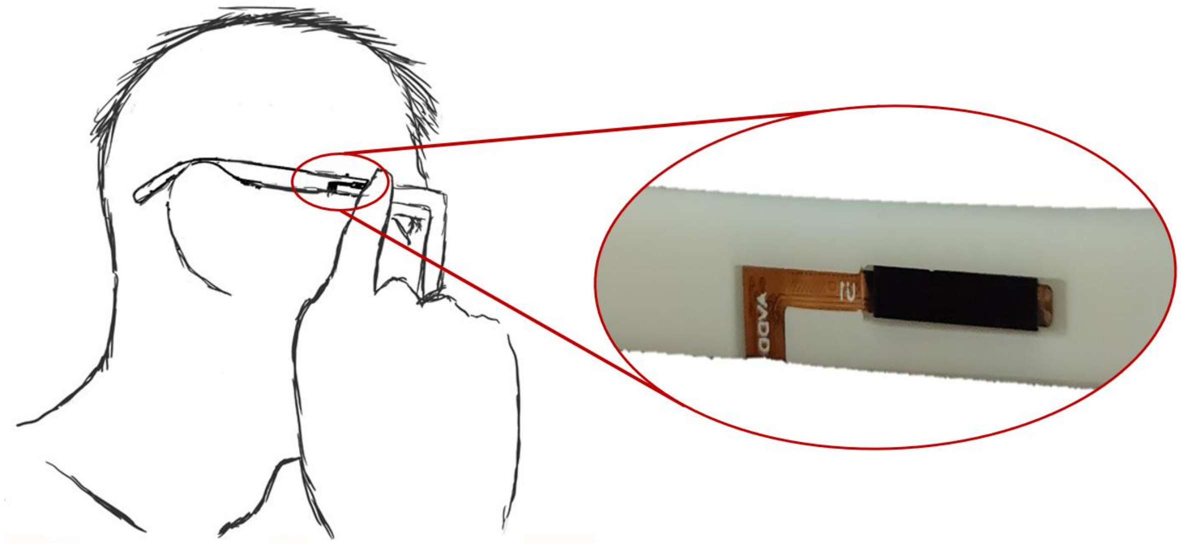


Figure 1: FingerGlass, an interaction technique for smart glasses. A fingerprint sensor, highlighted in the red circle, is seamlessly integrated onto the temple arm of the smart glasses. This strategic placement allows users to perform various finger gestures, providing a discreet, efficient, and ergonomic input solution for controlling smart glasses functions.

Abstract

Smart glasses hold immense potential, but existing input methods often hinder their seamless integration into everyday life. Touchpads integrated into the smart glasses suffer from limited input space and precision; voice commands raise privacy concerns and are contextually constrained; vision-based or IMU-based gesture recognition faces challenges in computational cost or privacy concerns. We present FingerGlass, an interaction technique for smart glasses that leverages side-mounted fingerprint sensors to capture fingerprint images. With a combined CNN and LSTM network, FingerGlass identifies finger identity and recognizes four types of gestures (nine in total): sliding, rolling, rotating, and tapping. These gestures, coupled with finger identification, are mapped to common smart glasses commands, enabling comprehensive and fluid text entry and application control. A user study reveals that FingerGlass represents a promising step towards a fresh, discreet, ergonomic, and efficient input interaction with smart glasses, potentially contributing to their wider adoption and integration into daily life.

*indicates the corresponding author.

CHI '25, April 26-May 1, 2025, Yokohama, Japan
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1394-1/25/04
https://doi.org/10.1145/3706598.3713929

CCS Concepts

• **Human-centered computing** → **Interaction techniques**; **Gestural input**; **Text input**; • **Computing methodologies** → *Supervised learning*.

Keywords

Smart Glasses, Fingerprint Sensing, Gesture Recognition, CNN, LSTM

ACM Reference Format:
Zhanwei Xu, Haoxiang Pei, Jianjiang Feng, and Jie Zhou. 2025. FingerGlass: Enhancing Smart Glasses Interaction via Fingerprint Sensing. In *CHI Conference on Human Factors in Computing Systems (CHI '25), April 26-May 1, 2025, Yokohama, Japan.* ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3706598.3713929

1 Introduction

Smart glasses, unlike immersive Virtual Reality (VR) devices such as Meta Quest and Apple Vision Pro or Augmented Reality (AR) devices like Microsoft HoloLens, prioritize lightweight design and everyday usability. While VR and AR devices typically involve bulky headsets that limit mobility and social acceptability, smart glasses aim to seamlessly integrate digital information into the user's natural field of view with minimal form factor impact, offering a truly wearable experience comparable to traditional eyeglasses. This design philosophy, exemplified by products like the Ray-Ban Meta Smart Glasses, highlights the potential of smart glasses for continuous, unobtrusive use in daily life. It is anticipated that smart glasses will incorporate an increasingly diverse array of sensors to enhance functionalities and gather detailed information from the external environment. As the capabilities of smart glasses expand, the complexity of input control requirements will naturally escalate, calling for more sophisticated and versatile interaction techniques. However, realizing this potential hinges on developing intuitive, efficient, and socially acceptable input methods, a significant challenge that current solutions struggle to address.

Existing integrated solutions primarily rely on touchpads and voice commands, both with significant drawbacks. Touchpads integrated onto the temple arms of smart glasses, constrained by their limited size, often lead to user frustration and errors due to poor accuracy and limited gesture complexity [27]. Voice input, while seemingly natural, raises significant privacy concerns and proves impractical in noisy environments or during sensitive conversations [34]. Relying solely on voice commands means sacrificing privacy in quiet environments or being rendered unable to interact with your device in crowded spaces.

Other avenues that may be integrated into smart glasses, such as hand gesture recognition [10, 13, 15], head gesture recognition [47], and eye tracking [2, 11, 37, 40], while promising more natural interaction, present significant challenges. These methods can be either vision-based or IMU-based, each with its own set of complexities and limitations. Vision-based systems generally require significant computational resources, which can lead to increased demand for processing power and may ultimately reduce battery life—an essential consideration for all-day wearable devices. On the other hand, IMU-based systems used for head gesture recognition often lead to discomfort and offer a limited range of gestures.

This work introduces FingerGlass, an interaction technique for smart glasses that leverages a commonly overlooked yet readily available sensor: the side-mounted fingerprint sensor. The miniaturization of fingerprint sensor technology, driven by the widespread adoption of smartphones, has advanced to the point where integration into the temple arms of smart glasses is feasible. Through a combined CNN and LSTM network, FingerGlass accurately identifies the user's finger and recognizes four main gestures: sliding, rolling, rotating, and tapping. Sliding includes four directional movements—up, down, left, and right. Rolling consists of two movements: left and right. Rotating encompasses two actions: clockwise and counterclockwise. Finally, tapping is a single gesture accomplished by tapping the screen. In total, FingerGlass can recognize nine distinct actions across these four types of gestures, offering a versatile and user-friendly interaction experience.

FingerGlass offers several key advantages over existing methods:

- **Ergonomics and Discreetness:** The side-mounted fingerprint sensor is strategically positioned on a smart glasses' arm, aligning perfectly with where the user's finger naturally falls during interaction. This design allows for comfortable and subtle input, eliminating the need for exaggerated hand movements. Compared to voice commands, which raise privacy concerns and are publicly audible, and vision-based gesture recognition, which can involve more overt hand movements, FingerGlass provides a more private and subtle interaction method. Users can enjoy the familiar ease of touch-based interaction with enhanced discretion, a crucial factor for increasing the social acceptability of smart glasses and driving their wider adoption. While our NASA-TLX results showed slightly higher physical demand compared to 1D Handwriting in our study, participants reported subjective comfort and natural hand positioning with FingerGlass.
- **Rich Input Space:** Unlike limited touchpad areas, the fingerprint sensor captures rich fingerprint information and is sensitive to subtle pressure and movement variations. This allows for a wider range of recognizable gestures, unlocking possibilities for complex command input and text editing. As smart glasses evolve to include more sensors and capabilities, the extensive input space offered by FingerGlass will become increasingly valuable.
- **Design Simplicity and Enhanced Comfort:** By integrating readily available and mature fingerprint sensors into smart glasses, FingerGlass avoids the need for additional independent hardware, minimizing cost and design complexity. Compared to eye-tracking sensors that require placement in front of the eyes, the fingerprint sensor integrated into the temple arm offers superior comfort for extended wear.

Through a user study comparing FingerGlass to traditional touchpad interaction, we demonstrate significant improvements in task completion time and user satisfaction. Our FingerGlass implementation achieves a gesture recognition accuracy exceeding 96% with a real-time response time of average 73 ms, enabling a text entry speed of 12.72 WPM. Our findings suggest that FingerGlass provides a compelling alternative for intuitive, efficient, and socially acceptable smart glasses control, paving the way for more seamless integration of these devices into our daily lives.

2 Related Work

Smart glasses are an emerging wearable device that offers users convenient access to information and interaction but also presents challenges in input method design. The ideal input method for smart glasses should be efficient, easy to use, private, and socially acceptable. However, existing input methods, such as touchpads and voice recognition, have significant limitations in these areas [27, 34]. To address these challenges, researchers have explored various novel input techniques, which can be broadly categorized into two types: using external devices for input and input methods that do not require external devices.

2.1 Using External Devices for Input

To expand the input capabilities of smart glasses, some studies have explored using external devices as input auxiliaries. For example, the touchscreens and physical buttons on smartphones and smartwatches [1] can provide a relatively comfortable text input experience. Ahn et al. [1] explored the feasibility of using a smartwatch as an input device for smart glasses and designed two input methods based on the touchscreen and physical buttons: SwipeBoard and HoldBoard. Additionally, smart glasses like Sony SmartEyeglass and Epson Moverio can be controlled using connected remote touchpads. Studies have explored more efficient text input layouts for these remote controllers [31, 43].

Besides smartphones and smartwatches, some research has explored other types of external devices. For example, Hsieh et al. [23] designed a system combining a haptic glove and smart glasses, allowing users to perform various interactive operations such as text input, icon selection, and scrolling by touching specific areas on the glove. Researchers have designed rings equipped with sensors to track finger movements and positions for auxiliary input to smart glasses [4, 5, 25, 32, 46]. Devices such as smart wristbands and sleeves, like the smart wristband by Ham et al. [16], use capacitive sensors and inertial measurement units to detect muscle tension and wrist movements, or Gesturewrist and Gesturepad [35] providing a more natural input method. TapType [38] uses wrist-worn sensors to enable full-size QWERTY typing on any surface by interpreting finger taps, while TypeAnywhere [51] achieves a similar feat through a wearable device that decodes finger-tap sequences for ubiquitous text entry.

In addition to these devices, some studies have also explored using the body itself as an input [14, 17, 18, 29, 41, 42]. For example, Liu et al. [29] proposed PrinType, a system that uses a thumb-worn fingerprint sensor to recognize different fingerprint areas for text input. Gustafson et al. [14] studied a palm-based virtual interface, where users can obtain information by rubbing the palm and receive instructions through an auditory system. This method utilizes the proprioception of the human body to provide additional feedback. Wang et al. [41] proposed using the palm as a keyboard (PalmType), where users can input characters by touching different areas of their palms. Xu et al. [45] proposed TipText, a system that enables text entry by tapping on a miniature QWERTY keyboard overlaid on the user's fingertip. This approach leverages the user's spatial memory of the QWERTY layout and uses a statistical decoder to ensure accurate text input.

2.2 Input Methods Without External Devices

To eliminate dependency on external devices, some studies have explored directly implementing input methods on smart glasses.

2.2.1 Touch-Based Input. Touch-based input methods on smart glasses are typically implemented through touch panels on the frame or temple.

Google Glass's touch-sensitive frame supports sliding operations, allowing users to select characters by sliding. For instance, Yu et al. [48] proposed a single-stroke gesture system that utilizes the sliding area of the frame for character input. Islam et al. [24] proposed GlassPass, a smart glasses authentication system where users tap specific locations on the glasses' temple to input passwords.

2.2.2 Touchless Input. Touchless input methods aim to control smart glasses through natural behaviors (e.g., head movements, eye tracking) without physical contact.

Voice Recognition: Voice recognition has been widely adopted by Google Glass and Microsoft HoloLens, although it has limitations in noisy environments [34].

Head Movements: Head movement is a relatively natural interaction method that can be used for simple navigation and control operations. Yi et al. [47] explored using head movements as an input method for smart glasses. They designed the GlassGesture system, which utilizes the accelerometer and gyroscope built into Google Glass, to recognize six distinct head gestures: nodding, shaking, turning left, turning right, looking up, and looking down.

Eye Tracking: Eye-tracking technology can be used to achieve more precise and efficient interactions, such as text input and target selection. Guo et al. [11] proposed EyeClick, which combines eye-tracking and a handheld controller to enable efficient text input within the limited field of view and accuracy of smart glasses. Ahn and Lee [2] introduced Gaze-Assisted Typing, which combines eye-tracking and touchpad input to enhance text input efficiency. The multimodal system designed by Slambekova et al. [37] uses eye tracking for object selection, combined with hand gestures for manipulation.

Hand Tracking: Beyond head movements and eye-tracking, virtual keyboards using hand tracking have been explored as an input method for both virtual and augmented reality headsets. These systems typically utilize hand tracking to detect finger presses on a virtual keyboard projected within the user's field of view. Studies have investigated the feasibility and performance of different hand-based input methods, such as touch typing and gesture typing, on mid-air virtual keyboards in VR [6].

However, these input methods also have some limitations. For example, the recognition accuracy and speed of head movements are limited, and in certain cases, it may not feel natural. Eye-tracking technology typically requires additional hardware to be mounted on the front of the glasses, adding extra weight to the forefront. This not only increases the overall cost but also makes the system susceptible to inaccuracies due to variations in environmental lighting conditions. Similarly, implementing robust hand tracking for virtual keyboards on smart glasses presents challenges due to size and power constraints, potentially impacting accuracy and user experience.

2.3 Fingerprint-Based Interaction Techniques

In addition to the PrinType [29], other fingerprint-based interaction techniques have been explored in various contexts to enhance input methods and user authentication, though these works are not designed for smart glasses and cannot be directly applied to this context. Sugiura and Koseki [39] introduced the Fingerprint User Interface (FUI), utilizing fingerprint recognition to distinguish between different fingers of the same user, allowing each finger to hold specific commands or data objects. This method enables intuitive interactions, such as executing commands by touching with a designated finger. However, their work focused solely on differentiating fingers without incorporating gestures, limiting the range of possible interactions.

Holz and Baudisch [21] developed Fiberio, a rear-projected multi-touch table system capable of displaying images and simultaneously capturing fingerprints during touch interactions. By integrating fingerprint recognition with touch input, Fiberio provides secure and seamless user authentication in collaborative environments. However, their system is huge, and does not combine identity recognition with interaction, thus it still relies on basic swipe gestures for input, similar to conventional touchscreens.

Ostberg et al. [33] explored user perceptions of repurposing smartphone fingerprint sensors for gestural input, such as taps and swipes. While users generally favored this concept, their work lacked support for rotational and scrolling gestures and did not address the unique interaction challenges posed by smart glasses.

Ferrari and Tartagni [8] and Gust [12] both proposed systems utilizing the skin texture of fingertips for cursor control, though with some differences in implementation. Ferrari and Tartagni combined fingerprint recognition with cursor control, where identity recognition is achieved through skin texture analysis, and the on-screen pointer is controlled by the rolling and pitching of the finger. Gust's patent describes a compact optical pointing device that employs optical sensors to detect motion by tracking the finger's skin texture on a touch surface, offering an efficient and space-saving input method suited to portable devices. Both approaches demonstrate innovative use of fingertip texture for input; however, they remain limited to cursor control and do not develop algorithms necessary to recognize complex discrete gestures to execute complex commands.

To address the limitations of the aforementioned studies, we presents *FingerGlass*, an input technique specifically designed for smart glasses. FingerGlass leverages a side-mounted fingerprint sensor to recognize **finger identity** and a set of **distinct discrete gestures**. By combining finger identity with gesture recognition into a comprehensive command set, FingerGlass enables efficient, convenient, and private input interactions on smart glasses, including complex tasks like text input.

3 FingerGlass: Design and Implementation

This section details the design and implementation of FingerGlass, encompassing the gesture set, finger identity utilization, data acquisition, and the machine learning pipeline for robust finger gesture recognition.

3.1 Gesture Design and Finger Identity

FingerGlass leverages both distinct finger gestures and finger identity to create a versatile interaction space.

3.1.1 Gesture Set. The design of FingerGlass's gesture set prioritizes both ergonomics and intuitiveness. Our aim was to create gestures that feel natural, comfortable, and easy to perform on the side-mounted fingerprint sensor, maximizing its interaction potential for smart glasses. The system recognizes the following gestures:

- **Sliding:** Horizontal and vertical sliding motions on the sensor. Sliding leverages users' familiarity with touch interfaces, providing an intuitive method for navigation and control. Vertical sliding can adjust volume, while horizontal sliding can switch tracks or navigate menu items.
- **Rolling:** Forward and backward rolling motions on the sensor. The natural tilting motion of the finger makes this gesture comfortable and easy to learn, suitable for fine-tuned adjustments, such as scrubbing through media playback.
- **Rotating:** Clockwise and counter-clockwise rotation on the sensor. Rotating is distinct and less prone to accidental activation, making it suitable for commands like switching modes or applications. Although requiring a slightly larger movement, it remains comfortable on smart glasses.
- **Tapping:** A single quick tap on the sensor. Tapping serves as a basic selection or activation gesture, akin to a button press, offering a simple and universally understood interaction.

Figure 2 visually depicts these gestures. Compared to alternative gesture sets, such as those involving large arm movements or complex multi-finger configurations, FingerGlass's gestures are specifically designed for the constrained interaction space of smart glasses. They prioritize subtle, ergonomic, and socially acceptable interactions. Compared to typical side-mounted touchpads (e.g., Meta's Ray-Ban glasses), which often support only horizontal sliding and tapping, FingerGlass offers a richer set: vertical and horizontal sliding, forward and backward rolling, clockwise and counter-clockwise rotating, and tapping. This expanded gesture vocabulary enables more intuitive and efficient device navigation and control by mapping gestures to user-defined commands or predefined system actions. For instance, during music playback, horizontal sliding can switch tracks, vertical sliding can control volume, and rolling can adjust playback progress. Results from the first task in our user study 4.2.1 indicate that grounded in principles of ergonomics and intuitiveness, our gesture set provides a natural and efficient interaction method for smart glasses. Several key gesture characteristics informed the design of the input mapping strategies discussed in the following section.

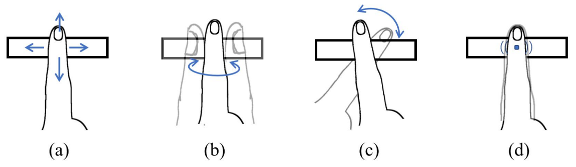
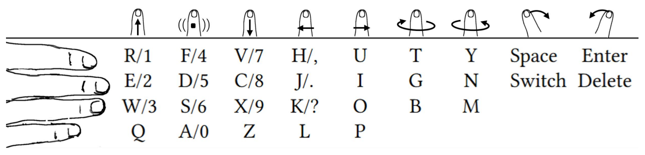


Figure 2: Illustration of recognized finger gestures: (a) sliding, (b) rolling, (c) rotating, and (d) tapping.

3.1.2 Finger Identity for Input Mapping. In addition to gesture and motion input, FingerGlass enables a broader spectrum of input mapping through the integration of finger identity. Figure 3 presents a text mapping rule used in our user study, which combines finger identification and gestures to facilitate eye-free single-handed text input on FingerGlass.

The design of this text mapping follows two fundamental principles. The first principle is to **leverage users' familiarity with the QWERTY keyboard layout**. For instance, in the case of sliding gestures, an upward sliding maps to keys in the upper row of the QWERTY keyboard layout, a downward sliding corresponds to the lower row, and tapping maps to the middle row. The numeral mapping adheres to a similar logic, where the top, middle, and bottom rows correspond to the positions of the numbers 1, 2, 3, 4, 5, 6, and 7, 8, 9, respectively. This mapping strategy simplifies the learning curve and facilitates user memory and adaptation. The second principle is to **consider the frequency of use of different fingers and gestures**. In the design of the key mappings, priority is given to the index and middle fingers, thereby minimizing reliance on the ring and little fingers, which enhances comfort and operational efficiency. The thumb finger is intentionally not used extensively for these interactions because its natural positioning and movement on the frame of the glasses is less consistent and ergonomic. Additionally, rotational gestures—requiring larger movement—are allocated to less frequently used special keys to mitigate wrist strain. This ensures a more comfortable and efficient interaction design.

By adhering to these mapping principles and finger usage strategies, we enable text typing conveniently across different modes. In particular, the "Switch" key design allows users to toggle between different modes (uppercase, lowercase, and numeric/symbol modes), thereby enhancing the overall usability. It is important to note that prior research has explored finger-based interaction in virtual reality and smart glasses, leveraging detailed fingerprint recognition for input [29]. Unlike its approach that defines numerous fingerprint regions, our method combines coarse finger identification with gestures. This combination offers a more intuitive and memorable mapping strategy compared to solely relying on touch location on the palm or finger. Furthermore, our approach simplifies the user enrollment process as it only requires identifying the finger, not specific regions within each fingerprint, making it more user-friendly compared to similar techniques utilizing fingerprint sensors.

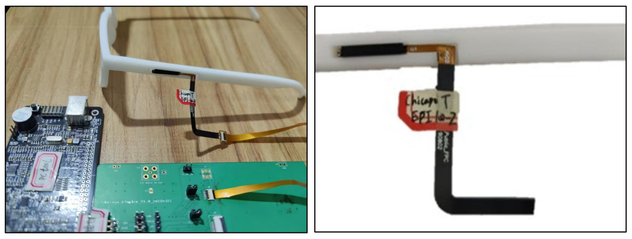


R/1	F/4	V/7	H/,	U	T	Y	Space	Enter
E/2	D/5	C/8	J/.	I	G	N	Switch	Delete
W/3	S/6	X/9	K/?	O	B	M		
Q	A/0	Z	L	P				

Figure 3: Mapping scheme for text entry using finger identity and gestures.

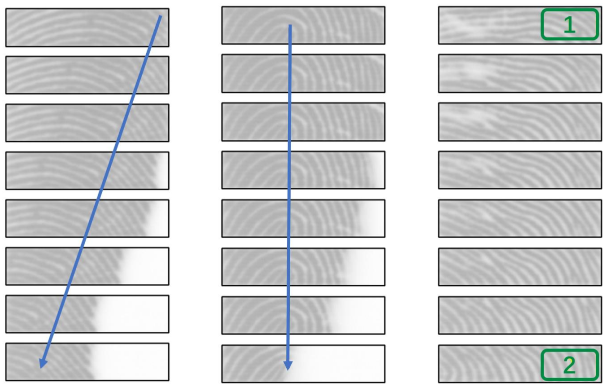
3.2 Device for Data Acquisition and Preprocessing

3.2.1 Device for Data Acquisition. A commercial side-mounted fingerprint sensor module (Goodix GF3626) has been integrated into the temple section of a prototype smart glasses frame fabricated through 3D printing (Figure 4). For ease of development, a smartphone development board was used in this prototype. However, it is important to note that the fingerprint sensor module can be made very small in actual production. The sensor captures capacitive image sequences at 40 frames per second during finger interaction. Each 160 × 36 pixel image represents capacitive changes due to finger contact and movement (Figure 5).



(a) prototype appearance (b) close-up of the sensor

Figure 4: Integration of the fingerprint sensor into the prototype smart glasses.



(a) slide (b) roll (c) rotate

Figure 5: Captured sequential fingerprint images. The blue lines with arrows denote the displacement of the fingerprint in the image sequence, while the green boxes highlight the changes in fingerprint orientation during rotational gestures.

3.2.2 Data Preprocessing. Raw capacitive image sequences undergo the following preprocessing steps:

Image Enhancement and Noise Removal: Histogram equalization enhances contrast to accentuate fingerprint features, followed by Gaussian filtering to reduce noise and improve the signal-to-noise ratio.

Gesture Segmentation: Consecutive frame differences exceeding a threshold ($\Delta C_t > \delta$) detect gesture start and end points, isolating relevant image segments.

$$\Delta C_t = \frac{1}{N} \sum_{i=1}^{N} |C_{t,i} - C_{t-1,i}| \tag{1}$$

where $C_{t,i}$ denotes the capacitance of the i-th pixel in the t-th frame, N is the number of all pixels.

Tap Detection and Frame Sampling: Tap gestures, characterized by short duration and significant contact area changes, are identified. For gestures not identified as taps, frame sampling is employed to normalize the length of the image sequence. Normalization is achieved by either upsampling short sequences or downsampling long sequences to the same length. Linear interpolation is utilized to extend the number of frames, and downsampling is achieved by randomly selecting a certain number of frames in the sequence. This step is crucial to ensure consistent feature extraction and gesture recognition across varying gesture speeds and durations.

3.3 Machine Learning Pipeline

FingerGlass implements and compares a dual recognition approach: a lightweight CNN-LSTM model and a rule-based engine for robust gesture recognition. Additionally, a dedicated CNN is used to identify fingerprints for distinguishing between fingers, enabling richer input interactions.

3.3.1 Fingerprint Identification. For identification, we select the frame with the largest contact area within each gesture sequence and build a lightweight CNN-based fingerprint recognition model. This model is based on the design principles of DeepPrint [7]. We adopt a more streamlined architecture by using MobileNetV2 [36], which is an improved backbone based on MobileNetV1 [22]. This change replaces the original ResNet18 [20], balancing recognition accuracy with reduced model complexity. Instead of using a softmax function for classification, we employ a fixed-length vector similarity-based identification approach.

Specifically, the trained CNN model extracts a 192-dimensional feature vector from each fingerprint image, which is then normalized to unit length. For an input fingerprint image, we first compute the cosine similarity between its feature vector and the feature vectors of the four registered fingers in the database. The cosine similarity is calculated as follows:

$$s(R_p, R_g) = R_p^T \cdot R_g \tag{2}$$

where R_p is the feature vector of the input fingerprint image, R_g is the feature vector of a registered fingerprint in the database, and $s(R_p, R_g)$ is the cosine similarity between R_p and R_g.

The system ranks the similarity scores and selects the fingerprint identity with the highest score as the recognition result. If the highest score exceeds a predefined threshold, the recognition is considered successful; otherwise, it is classified as an unknown finger.

3.3.2 Finger Gesture Recognition. For non-tap gestures, we implemented and compared two distinct methods for recognition: a rule-based engine and a lightweight CNN-LSTM deep learning model. A detailed comparison of their performance is provided.

Rule-Based Gesture Recognition: The rule-based gesture recognition method serves as a computationally lightweight and easily interpretable baseline for comparison against the more complex CNN-LSTM model. By leveraging readily observable features of fingerprint motion, the rule-based approach aims to achieve reliable gesture classification with minimal computational overhead.

The inherent diversity in captured fingerprint image sequences is highlighted in Figure 5, where each gesture type is visually distinct. These visual cues serve as the foundation for our gesture recognition approach. Sliding motions exhibit a smooth and consistent translation of the entire fingerprint pattern across successive frames. This movement is clearly depicted by the blue arrows in Figure 5, effectively illustrating the fingerprint's trajectory along the sensor's surface. The consistent shift of all fingerprint features is key to identifying this gesture. Unlike sliding, rolling gestures present a more nuanced pattern. While a slight overall translation of the fingerprint might be observed, the defining characteristic lies in the noticeable shift of the contact area's centroid within each frame. This subtle change in contact point, rather than a global shift, distinguishes rolling from sliding. Rotating gestures are visually apparent through the evolving orientation of the fingerprint throughout the sequence. Figure 5 uses green boxes to highlight the dynamic shift in fingerprint orientation within the contact area during the gesture. As shown, the fingerprint ridges gradually transition from a horizontal alignment to a near-vertical alignment as the finger rotates. This consistent change in orientation, even with minimal translation, is the hallmark of a rotating gesture.

Based on these distinct motion characteristics, we have formulated a set of rules for accurately classifying each predefined gesture. These rules are meticulously detailed in Table 1, providing a clear and concise guide to the specific conditions that trigger each gesture classification. For a deeper understanding of the complete rule-based gesture recognition process, including the algorithmic workflow and implementation details, please refer to Appendix A.2.1.

CNN-LSTM Based Gesture Recognition: Recent advancements in deep learning have led to significant progress in finger gesture estimation based on fingerprint [19] or touchscreen images [28, 30]. Compared to previous work, FingerGlass recognizes a broader range of fingerprint gestures, including rotation, sliding, rolling, and tapping by a network combining Convolutional Neural Network (CNN) feature extraction and Long Short-Term Memory (LSTM) temporal modeling.

The CNN component consists of three convolutional layers, max-pooling layers, and ReLU activation functions to extract spatial features from each fingerprint image. The LSTM component takes the sequence of feature vectors extracted by the CNN as input, learning the temporal dependencies to capture the dynamic changes in finger movements, ultimately outputting the gesture classification results. For the detailed architecture of the CNN and LSTM, please refer to the Appendix A.2.2.

3.4 Implementation and Performance Evaluation

We recruited 25 volunteers for training and validating our method. Each volunteer performed nine predefined gestures (slide up, slide down, slide left, slide right, roll left, roll right, rotate clockwise, rotate counter-clockwise, and tap) using eight fingers (excluding thumbs) on both hands. Each gesture was repeated 10 times. During the data collection for each gesture, a fingerprint sensor recorded

Table 1: Rule-Based Gesture Recognition Rules

Gesture	Feature	Rule
Sliding	Significant overall image translation	Calculate horizontal and vertical translation between consecutive frames D_x, D_y. If $\max\{|D_x|, |D_y|\} > D_{th}$, it is classified as sliding, with the direction corresponding to the dominant translation direction.
Rolling	Minor overall translation but significant centroid shift	Calculate horizontal and vertical translations D_x, D_y, and image centroid shifts V_x, V_y. If $\max\{|D_x|, |D_y|\} < D_{th}$ and $\max\{|V_x|, |V_y|\} > V_{th}$, it is classified as rolling, with the direction corresponding to the centroid shift direction.
Rotating	Change in image rotation angle	Calculate rotation angle change α between consecutive frames. If $|\alpha| > \alpha_{th}$, it is classified as rotating, with the direction corresponding to the angle change direction.
Tapping	Significant change in contact area with short duration	Contact area peak $A_{peak} > A_{th}$, duration $t < t_{th}$.

Note: D_{th}, V_{th}, α_{th}, A_{th}, **and** t_{th} **are predefined thresholds.**

Table 2: Accuracy Comparison Between Two Gesture Recognition Methods

Gesture	CNN-LSTM	Rule-Based
Sliding	98.5%	96.2%
Rolling	99.2%	95.5%
Rotating	98.2%	94.7%
Tapping	99.6%	99.6%
Average	98.8%	96.5%

Table 3: Accuracy of Fingerprint Identification and Gesture Recognition for Different Fingers

Finger	Identification	Gesture Recognition
Index Finger	99.6%	99.3%
Middle Finger	99.5%	99.0%
Ring Finger	99.0%	98.4%
Pinky Finger	98.8%	98.5%
Average	99.2%	98.8%

capacitive image sequences at a rate of 40 frames per second. This process yielded a total of 25 (volunteers) × 8 (fingers) × 10 (repetitions) × 9 (gestures) = 18,000 image sequences.

The dataset was randomly divided into five subsets for model training and evaluation using five-fold cross-validation. To rigorously evaluate the generalization capability, all images from the same individual were exclusively assigned to a single subset. During model training, a random resampling strategy was employed in each iteration to generate training samples. For model validation, we fixed 20 resampling seeds for each user to ensure that different algorithms were evaluated on the same validation sets. Each gesture had 5 (volunteers) × 10 (repetitions) × 20 (resamples) = 1,000 samples for performance evaluation in the validation stage. Here is the performance of our methods.

3.4.1 *Comparison of Recognition Accuracy.*

The comparison of recognition accuracy between the CNN-LSTM-based method and the rule-based method for the four predefined gestures—sliding, rolling, rotating, and tapping—is shown in Table 2. The CNN-LSTM-based method slightly outperforms the rule-based method in recognition accuracy, though both approaches effectively recognize the four predefined gestures. While the CNN-LSTM-based method demonstrated a marginal advantage in accuracy and was ultimately employed in our user study for practical use, the rule-based method's simplicity and interpretability proved invaluable for rapid prototyping, feature importance analysis, and establishing a strong baseline for performance evaluation.

3.4.2 *Finger-Specific Recognition Performance.*

The performance of both the fingerprint identification and gesture recognition models was evaluated for each finger separately to assess any potential variations in accuracy. Table 3 presents the results of this analysis.

As shown in Table 3, both fingerprint identification and gesture recognition achieve very high accuracy across all four fingers, with index and middle fingers performing the best. This indicates the robustness of our system in effectively distinguishing and recognizing gestures regardless of the finger used.

3.4.3 *Gesture Classification Confusion Matrix.*

To further investigate the performance of the CNN-LSTM gesture recognition model, we generated confusion matrices for each finger, as illustrated in Figure 6. These matrices provide insights into the types of misclassifications that occur and identify potential areas for improvement.

Overall, the gestures are generally well-distinguished, with most misclassifications occurring between sliding and rotating gestures, reflecting their similar motion dynamics. Sliding and rotating movements are often confused, particularly for the ring and pinky fingers,

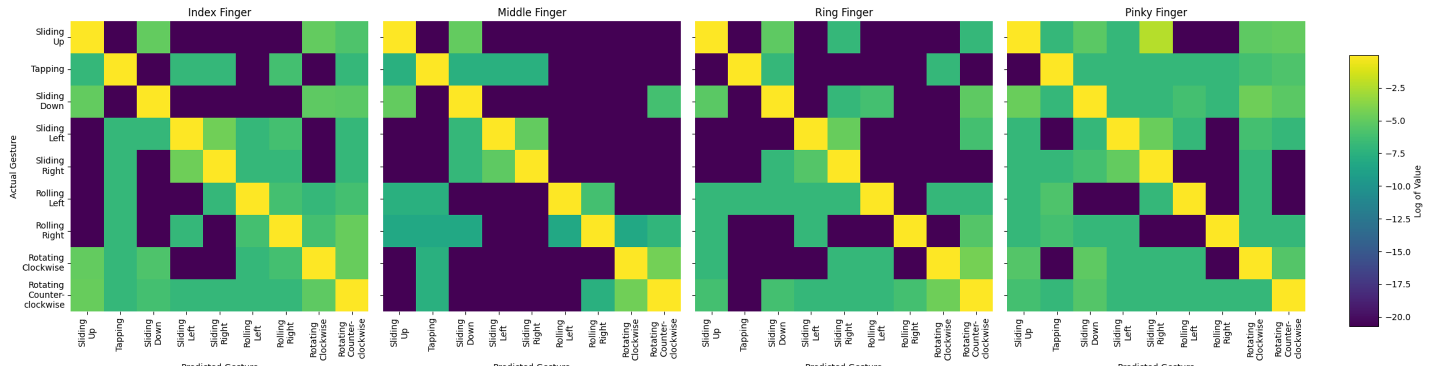


Figure 6: Confusion matrices for gesture recognition using CNN-LSTM for each finger. The small magnitudes of the observed error rates necessitate a logarithmic color mapping to resolve and emphasize the variations crucial for analysis.

due to the smaller contact area and subtle movement distinctions. Rolling gestures exhibit the highest accuracy, with minimal confusion with other gestures, suggesting these movements generate distinct patterns easily recognized by the CNN-LSTM model. Tapping gestures, characterized by unique temporal profiles, are accurately classified with very few errors.

3.4.4 Finger-Gesture Combination Performance. Finally, we analyzed the combined performance of fingerprint identification and gesture recognition for each finger-gesture pair. This provides a comprehensive overview of the system's accuracy in recognizing specific input commands. Table 4 presents the recognition accuracy for each combination.

As evident from Table 4, FingerGlass exhibits high and consistent recognition accuracy across all finger-gesture combinations, enabling robust and versatile interaction with smart glasses. It demonstrates that FingerGlass effectively utilizes fingerprint information for both identification and gesture recognition. Notably, the combination of tapping and rolling consistently achieves the highest accuracy, exceeding 99%. Conversely, sliding and rotating gestures, particularly when performed with the ring and pinky fingers, show slightly lower accuracy compared to other combinations. This highlights the potential for further improvement in distinguishing subtle movements associated with these gestures, particularly for fingers with smaller contact areas.

4 User Study

To comprehensively evaluate the performance and usability of FingerGlass on smart glasses, we conducted a user study that compared FingerGlass with a strong baseline, the One-Dimensional Handwriting method [48]. This method has demonstrated both high performance and significant influence to the field, providing a robust benchmark against which to assess FingerGlass's efficiency, accuracy, and user comfort. The comparative framework ensures a robust assessment of FingerGlass's potential in real-world applications while identifying unique strengths.

4.1 Participants and Apparatus

Twelve participants (six male, six female, aged 22-35) took part in the study. All participants had normal vision and finger dexterity.

The experiment was conducted using a custom-designed smart glasses prototype equipped with side-mounted fingerprint sensors. Data from the prototype was transmitted in real-time to an external PC (Intel Core i5 processor) where all data processing and algorithm execution occurred. Despite the data transfer overhead, both finger identity recognition and gesture classification, including data transmission time, were achieved in under 100 ms, average 73 ms, demonstrating the system's efficiency. While the current prototype utilizes an external PC, the computational demands of FingerGlass suggest that designing a highly integrated, dedicated processing chip is entirely feasible for future iterations, like the patent [8].

4.2 Procedure

Before using FingerGlass, participants were required to register their fingerprints. During the registration process, each user placed their finger perpendicularly to the temple arm, sliding it from the upper part of the first phalanx to the lower part. The sequence of fingerprint images captured served as the registration template for each finger. During the user study, participants were instructed that all initial placements should make the finger approximately perpendicular to the temple arm. This would allow for the capture of images similar to the registration template, thereby facilitating accurate fingerprint verification. Then, participants were asked to complete two basic tasks designed to assess the performance of FingerGlass in command input and text entry:

4.2.1 ***Command Input.*** Participants listened to a series of instructions, each corresponding to a specific FingerGlass gesture (9 gestures in total). The order of the gestures was randomized to prevent any biases or learning effects. Participants executed the corresponding gesture upon hearing the instruction. Each gesture was performed three times using four different fingers, resulting in a total of 12 attempts per gesture per participant. The system automatically recognized and recorded the recognition results. The

Table 4: Recognition Accuracy for Different Finger-Gesture Combinations

Gesture	Index Finger	Middle Finger	Ring Finger	Pinky Finger	Average
Sliding Up	99.3%	99.2%	98.3%	98.0%	98.7%
Tapping	99.7%	99.8%	99.4%	99.4%	99.6%
Sliding Down	99.1%	98.6%	98.2%	98.3%	98.6%
Sliding Left	99.2%	99.0%	98.3%	98.6%	98.8%
Sliding Right	99.4%	99.5%	98.6%	98.4%	99.0%
Rolling Left	99.6%	99.3%	98.7%	98.9%	99.1%
Rolling Right	99.8%	99.4%	98.6%	99.0%	99.2%
Rotating Clockwise	98.7%	98.4%	97.8%	97.8%	98.2%
Rotating Counter-clockwise	98.9%	98.2%	97.9%	98.1%	98.3%
Average	99.3%	99.0%	98.4%	98.5%	98.8%

accuracy of command recognition was recorded. Participants provided feedback on the gesture design through a questionnaire, including subjective ratings (1-5 Likert scale) for intuitiveness and ergonomics.

4.2.2 ***Text Entry.*** To assess the effectiveness of FingerGlass, we replicated the text input functionality of the 1D Handwriting system [48] using FingerGlass hardware. In 1D Handwriting, users input characters by performing a series of directional strokes that roughly resemble simplified letter shapes. For example, the letter 'L' is represented by a downward stroke followed by a rightward stroke. Different letters are formed by unique sequences of these directional strokes (up, down, left, right, and sometimes short/long variations). The system then recognizes these stroke sequences and translates them into characters. Leveraging the side-mounted fingerprint sensor, our system enabled users to perform directional gestures along the sensor's long edge, effectively mirroring the one-dimensional input paradigm of the 1D Handwriting technique. This allowed for a fair comparison under identical hardware constraints. In replicating the text mapping, we adhered to the original paper's recommendations, utilizing the "flip-up" gesture for toggling between uppercase and lowercase characters. However, as the original paper lacked specific gesture definitions for punctuation marks and numbers, we implemented a simplified approach for the 1D Handwriting method: users represented all punctuation and numeric input with a double-tap gesture. In contrast, FingerGlass users were required to input the correct punctuation marks and numbers directly, providing a more realistic and demanding test scenario for our system. This difference in handling special characters reflects a key distinction in the two input methods and highlights FingerGlass's capacity for more complex and accurate text entry. Participants engaged in three distinct phases using both input methods:

Phase I: Initial Learning. Participants underwent a 30-minute training session on FingerGlass usage, focusing on text entry. Following the training, participants used FingerGlass to input a standardized text passage containing English in upper and lowercase, punctuation marks, and numbers. The text passage consisted of three sentences selected from the Standardized Project Gutenberg Corpus [9] and was approximately 50 words in length. Throughout the input process, participants were allowed to correct errors and the system provided real-time audio feedback of the entered characters.

Phase II: One-Week Follow-up. One week later, participants performed the text entry task again following the same training pattern. Participants again inputted a standardized text passage containing English in upper and lowercase, punctuation marks, and numbers. The system continued to provide real-time audio feedback of the entered characters during this phase.

Phase III: Extensive Training. The goal of this phase was to simulate the speed and accuracy users could achieve with the input method after mastering it. The participants were randomly assigned a unique phrase and instructed to enter it twelve times. Performance metrics were recorded only for the final (twelfth) repetition.

4.3 Metrics and Results

We used distinct metrics for Command Input and Text Entry tasks. The primary metric for assessing command input was Command Recognition Accuracy, while several other metrics were used to evaluate Text Entry.

4.3.1 ***Command Recognition Accuracy.*** This metric measured the percentage of correctly recognized gestures out of the total gestures performed. As depicted in Figure 7, FingerGlass achieved high accuracy across all nine commands, ranging from 95.42% to 97.60%. This indicates the robustness and reliability of the gesture recognition system.

4.3.2 ***Gesture Subjective Rating.*** As shown in Table 5, all gestures scored highly on intuitiveness and ergonomics, though rotating had slightly lower ergonomics due to the larger movement. Participants reported that they find all gestures comfortable and easy to perform, noting that the glasses frame acted as a stable reference point. This frame of reference created proprioceptive-like awareness of the sensor location, making it incredibly easy to find and activate without any training, much like touching one's

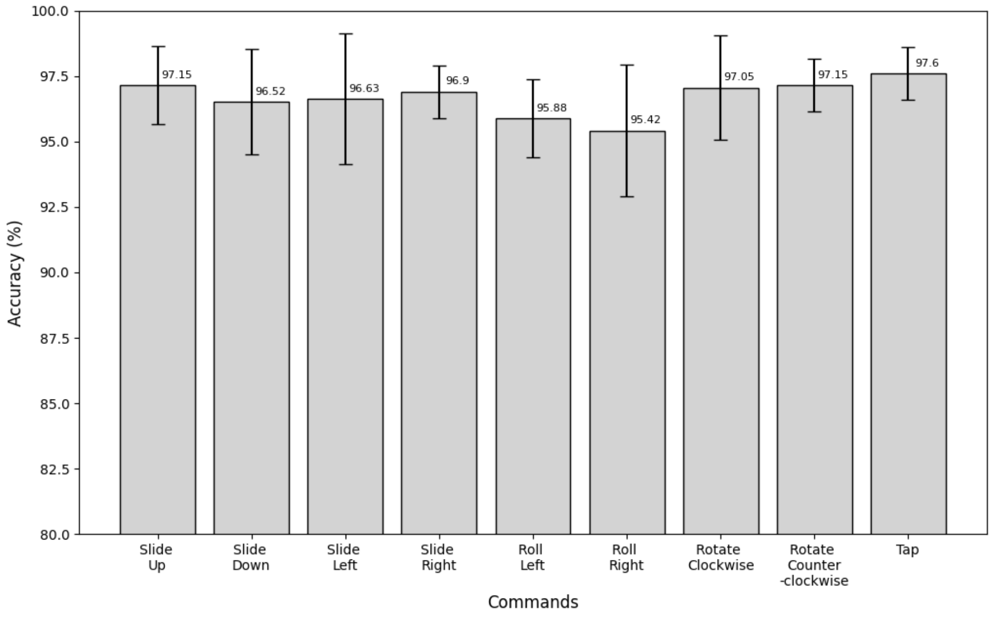


Figure 7: Command accuracy with standard deviation.

own nose or ear. This inherent ease of use contributed to the high intuitiveness and ergonomics scores.

Table 5: Gesture Subjective Rating

Gesture	Intuitiveness (1-5)	Ergonomics (1-5)
Sliding	4.8 ± 0.3	4.9 ± 0.2
Rolling	4.6 ± 0.4	4.7 ± 0.3
Rotating	4.5 ± 0.5	4.6 ± 0.4
Tapping	4.9 ± 0.3	4.9 ± 0.3

4.3.3 ***Words Per Minute (WPM).*** This metric quantified the text entry speed using FingerGlass, calculated as the number of words correctly entered per minute. Figure 8 illustrates WPM with two input methods in three phases. a notable improvement in WPM from Phase I (9.61 WPM) to Phase II (12.72 WPM), suggesting a positive learning curve associated with FingerGlass. The 12.72 WPM achieved in Phase II is comparable to other text entry techniques for wearable devices, such as TipText (11.9 WPM on average and 13.3 WPM in the last block), which requires an external peripheral for one-handed typing [45], and DigiTouch (13 WPM on average) which utilizes both hands [44]. The speed of 23.5 WPM achieved in Stage 3, a more restricted and simplified text entry evaluation, demonstrates the efficiency that can be achieved when the user is proficient enough to meet the most common use cases in smart glasses, such as instant messaging, short notes, and keyword searches.

The 1D handwriting method had significantly lower WPM values: stage 1 (8.1 WPM), stage 2 (9.5 WPM) and stage 3 (16.1 WPM). Although it has improved over time, the increase has been limited and consistently lower than FingerGlass. The WPM values of the two methods in the first stage were similar, because the letter imitation gestures of the 1D handwriting method were easy to learn. However, as the training progressed, the advantages of FingerGlass gradually became apparent. This is mainly because FingerGlass, combined with fingerprint recognition, can be input with simple gestures, while 1D handwriting requires multiple consecutive gestures, which prolongs the input time and reduces the input efficiency. It should be noted that FingerGlass not only achieves faster typing speed, but also supports a more complete character set, including punctuation and numbers.

Although the WPM values during the typing task in phase I and II may not appear particularly high, several factors contributed to these.

Input Complexity: The input text space is quite extensive, including uppercase letters, lowercase letters, numbers, and punctuation. Users needed to switch between various modes, closely simulating the demands of real-world typing scenarios. Using a more restricted range, such as only letters and spaces, or employing variant T9 mapping methods like [3] could also simplify the input process significantly, thereby increasing typing speed.

New Typing Mapping: Both FingerGlass and 1D handwriting employ new text typing mapping. Despite the optimized mapping based on the QWERTY layout, FingerGlass posed some challenges for users to memorize within the limited learning time of just 30 min. This likely resulted in users not being completely familiar or proficient during the test phase.

Lack of Visual Feedback: The current prototype was designed for smart glasses without display and thus users had to rely solely on auditory feedback, which is less intuitive and could delay input speeds.

Despite these challenges, it is noteworthy that all users were able to achieve more than 20 WPM during phase III using FingerGlass, demonstrating the system's potential with increased familiarity and practice.

4.3.4 *Error Rate (ER).*

This metric represented the percentage of incorrectly entered characters relative to the total characters inputted. As shown in Figure 8, the error rate in FingerGlass decreased significantly from stage 1 (8.55%) to stage 2 (5.47%) and further to stage 3 (3.12%), indicating an increase in proficiency in text entry with continued use. For 1D handwriting methods, error rates were higher in all stages: stage 1 (10.34%), stage 2 (9.87%), and stage 3 (7.65%). This is due to the fact that the method requires performing continuous gestures along a single axis with direction changes and precisely controlling the distance of the slide to distinguish between short and long slides, thus increasing the difficulty of the input. The probability of errors in a single direction accumulates many times, further increasing the likelihood of errors. It is worth noting that the comparative method may require longer sliding gestures, and the limited length of the fingerprint sensor used in our implementation could have contributed to confusion between short and long slides, potentially affecting its observed performance.

The low error rate of FingerGlass's implementation, especially in the third phase, suggests that users can attain a high degree of accuracy in text entry on smart glasses after mastering the finger identification-based gesture input method.

4.3.5 *NASA-TLX.*

This subjective workload assessment tool was employed to evaluate participants' perceived workload after completing the text entry task using FingerGlass. The NASA-TLX encompasses six dimensions: mental demand (MD), physical demand (PD), temporal demand (TD), performance (P), effort (E), and frustration (F). Each dimension was rated on a scale of 1 to 100, with higher scores indicating higher workload. Figure 9 presents the NASA-TLX scores after phase III.

FingerGlass and 1D handwriting exhibited low physical demands, with FingerGlass reporting a slightly higher score (PD: 48.59) compared to 1D handwriting (PD: 39.24). This difference may be attributed to the finger-switching required for various gestures in FingerGlass. Both methods necessitate learning new input mappings, resulting in relatively similar mental demands (MD: 66.62 for FingerGlass vs. 67.09 for 1D handwriting). The gesture design of 1D handwriting, mimicking the trajectory of letter writing, may contribute to its slightly low perceived mental workload. Despite this, 1D handwriting consistently yielded higher scores across all other NASA-TLX dimensions. 1D handwriting presented significantly higher levels of temporal demand (TD: 75.27), performance pressure (P: 78.25), effort (E: 81.21), and frustration (F: 40.22) compared to FingerGlass (TD: 60.55, P: 67.44, E: 74.09, and F: 32.98, respectively). Prolonged use of 1D handwriting for text entry appears to induce greater fatigue and frustration, likely due to its more complex gestures, reduced accuracy, and the consequent increase in time and effort required. Conversely, FingerGlass, with its more concise gestures and higher accuracy, effectively mitigates user burden across these dimensions, affording a more comfortable and efficient input experience.

4.4 Discussion

The results of the user study indicate that FingerGlass, as an interaction technique leveraging fingerprint sensors for smart glasses, holds great promise for enhancing user experience. Participants widely reported positive feedback regarding FingerGlass's ergonomics and social acceptability. The side-mounted sensor aligns perfectly with where the user's finger naturally falls during interaction, incurring no extra movement and unnecessary social awkwardness, making it more suitable for use in social settings compared to interaction modalities requiring overt hand gestures or voice interactions.

FingerGlass offers a richer interaction space and a more diverse command set, demonstrating superior performance in NASA-TLX metrics compared to the 1D handwriting method. It is important to note that, while subjective feedback on ergonomics was positive, our NASA-TLX assessment showed a slightly higher physical demand score for FingerGlass compared to 1D Handwriting. This difference may be attributed to the finger-switching required for different gestures in FingerGlass. A key advantage of FingerGlass is its stability. Users frequently reported significant device wobble with 1D handwriting due to the continuous directional changes required, particularly problematic on lightweight smart glasses, a burgeoning technological trend. This wobble can lead to visual discomfort and impaired interaction. Conversely, FingerGlass exhibited minimal movement; only three users noted a slight, transient wobble during tapping input. These findings suggest that FingerGlass offers a more stable and comfortable user experience. Further research will investigate its long-term usability and user acceptance in real-world scenarios.

The prototype of FingerGlass that we implemented for our user study lacks visual display entirely. This absence of visual feedback, while the system relies solely on discrete gestures and character input, have caused inconvenience for some users and contributed to the slightly higher learning costs and cognitive load.

5 Applications

To demonstrate the interaction potential and usefulness of FingerGlass in practice, we have designed a hypothetical use case. Based on this design, users can utilize FingerGlass to perform functions like music playback, calls, photography/videography, and navigation, and switch between these functions seamlessly.

As demonstrated in Figure 10, all applications are seamlessly integrated into one operating system. The FingerGlass fingerprint module serves as an input device, reporting a variety of input events. The specific mapping and response to these events are determined by each individual application. Management and scheduling of applications based on user input fall under the responsibility of the operating system. Application invocation by the system can be performed through a linear menu (navigating with directional gestures) or through shortcut commands (app names or specific gestures). Within any application, the user can navigate either using a linear menu (directional gestures) or shortcuts (app names or specific gestures), and also use text input where necessary.

For example, the system is in a ready state at start, providing the current time and battery level via voice broadcast upon any finger tap. A clockwise index finger rotation transitions to the

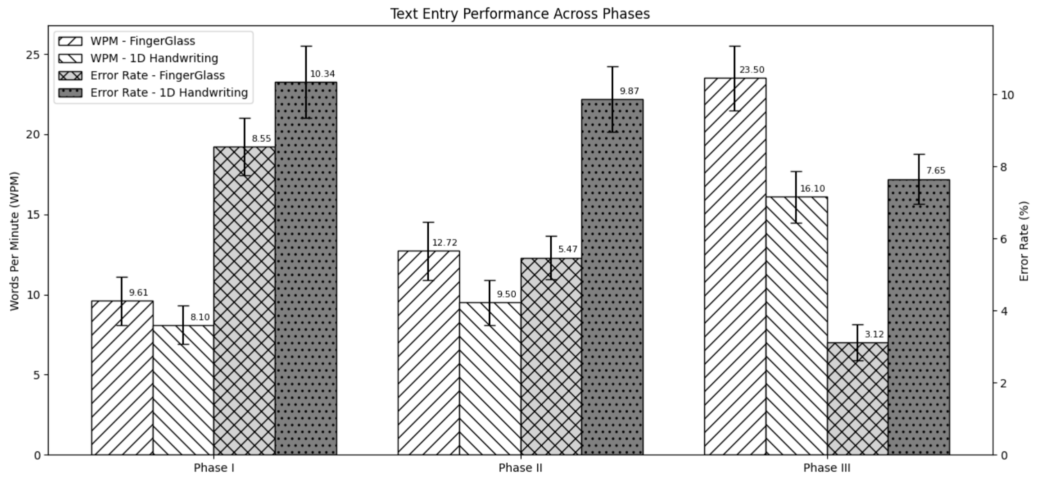


Figure 8: Performance metrics across the three phases in the text typing task.

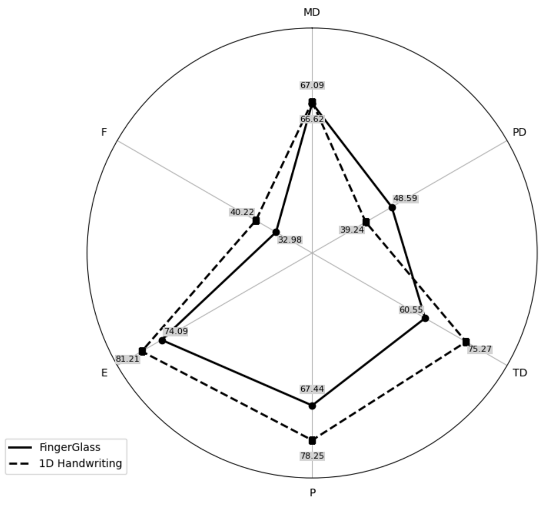


Figure 9: NASA-TLX workload assessment of the 1D handwriting and the FingerGlass.

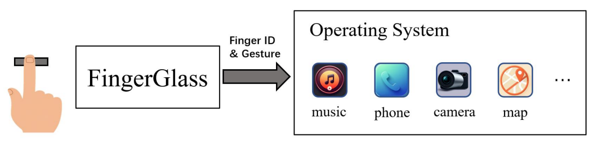


Figure 10: Diagram of the FingerGlass application structure.

first application. Within any application, a clockwise index finger rotation switches to the next application, while a counterclockwise rotation returns to the ready state. The specific gesture mappings for each application are presented in Table 6, Table 7, Table 8, and Table 9. The overall design emphasizes the use of the index finger for operations, as it is more natural and accurate. Additionally, all operations provide voice feedback, making it suitable for smart glasses without display feedback. The text input interface acts as a system-level application that can be activated or deactivated by a clockwise rotation of the ring finger. This design choice utilizes a free mapping key from the typing mapping in Table 3, ensuring there is no overlap with other operations, thus maintaining user-friendliness and operational efficiency. It can be invoked anytime when text input is required, supporting various text entry needs across different applications.

Table 6: Gesture Mapping for Music Playback

Action	Gesture
Volume Increase	Index finger slide up
Volume Decrease	Index finger slide down
Next Track	Index finger slide right
Previous Track	Index finger slide left
Fast Forward	Index finger roll left
Rewind	Index finger roll right
Play/Pause	Index finger tap
Text Input for Searching Songs	Ring finger rotate

Table 7: Gesture Mapping for Call Function

Action	Gesture
Answer/Make Call	Index finger slide left
Reject/End Call	Index finger slide right
Record Audio	Index finger tap
Enter Text Input Interface	Ring finger rotate

Table 8: Gesture Mapping for Photography/Videography

Action	Gesture
Take Photo	Index finger tap
Start/Stop Recording	Middle finger tap

Table 9: Gesture Mapping for Navigation

Action	Gesture
Start/End Navigation	Index finger tap
Status Broadcast	Middle finger tap
Text Input for Destination Search	Ring finger rotate

6 Limitations and Future Work

This study represents an initial exploration of FingerGlass and acknowledges certain limitations that provide avenues for future research.

Size of the Prototype The current FingerGlass prototype, constrained by hardware limitations, has not yet achieved full integration with smart glasses and is not convenient for users to wear for extended periods in daily life. Future work will focus on developing low-power, miniaturized dedicated chips to fully integrate FingerGlass into smart glasses, creating a more practical and user-friendly product.

Visual Feedback The prototype of FingerGlass is designed specifically for smart glasses without a display like the Ray-Ban Meta Smart Glasses, hence we have not explored the impact of display feedback on text input performance. Visual feedback can decrease the cognitive load for users and enhance text input efficiency. Future work will also explore leveraging language models to further improve text input efficiency with or without visual feedback.

Input Technique The current mapping between finger gestures and commands may not be optimal for all users and applications. Further research is needed to explore alternative mapping methods, potentially employing machine learning techniques to personalize gesture recognition and command assignment. Additionally, this study did not utilize language models to predict and correct text input. Future work could leverage language models, similar to those used in smartphone keyboards [50], to further enhance text input speed and accuracy with FingerGlass. Furthermore, considering the capability of fingerprint sensors to track finger movements with high fidelity, incorporating word gesture input methods presents a compelling direction for future research. The seminal work by Kristensson and Zhai on command strokes [26], which explored using pen gestures on a keyboard for efficient command selection, could provide valuable insights for adapting such techniques to the FingerGlass paradigm. This could allow for fluid and rapid text entry, leveraging the natural dexterity of finger movements on a small surface.

Other Finger Attributes This paper primarily considered the movement, rotation, and identity of fingers. However, additional attributes can be extracted from the fingerprint images, such as finger shear force [49] and finger angle [19], which would further increase the input dimensions of the fingerprint sensor. Measuring these attributes from a small area sensor poses a challenge, but future work could explore these possibilities to enrich the FingerGlass input capabilities.

By addressing these limitations, future research can enhance the design and functionality of FingerGlass, making it a more robust and practical tool for interacting with smart devices.

7 Conclusion

FingerGlass presents a promising interaction technique for smart glasses, leveraging fingerprint sensing. By accurately identifying fingers and recognizing diverse gestures, it offers a more ergonomic, discreet, and feature-rich interaction space. While further development is needed to miniaturize the technology and explore additional finger attributes, our user study demonstrates promising results in command input and text entry tasks, highlighting the potential of FingerGlass to enhance smart glasses usability and drive wider adoption.

Acknowledgments

We would like to thank Goodix Technology for providing the fingerprint sensors used in this project. This work was supported in part by the National Natural Science Foundation of China under Grants 62376132 and 62321005.

References

[1] Sunggeun Ahn, Seongkook Heo, and Geehyuk Lee. 2017. Typing on a Smartwatch for Smart Glasses. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*. 201–209.

[2] Sunggeun Ahn and Geehyuk Lee. 2019. Gaze-Assisted Typing for Smart Glasses. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 857–869.

[3] Kai Akamine, Ryotaro Tsuchida, Tsuneo Kato, and Akihiro Tamura. 2024. PonDeFlick: A Japanese Text Entry on Smartwatch Commonalizing Flick Operation with Smartphone Interface. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–11.

[4] Daniel Ashbrook, Patrick Baudisch, and Sean White. 2011. Nenya: Subtle and Eyes-Free Mobile Input with a Magnetically-Tracked Finger Ring. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2043–2046.

[5] Liwei Chan, Yi-Ling Chen, Chi-Hao Hsieh, Rong-Hao Liang, and Bing-Yu Chen. 2015. CyclopsRing: Enabling Whole-Hand and Context-Aware Interactions Through a Fisheye Ring. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 549–556.

[6] John J. Dudley, Jingyao Zheng, Aakar Gupta, Hrvoje Benko, Matt Longest, Robert Wang, and Per Ola Kristensson. 2023. Evaluating the Performance of Hand-Based Probabilistic Text Input Methods on a Mid-Air Virtual Qwerty Keyboard. *IEEE Transactions on Visualization and Computer Graphics* 29, 11 (2023), 4567–4577. https://doi.org/10.1109/TVCG.2023.3320238

[7] Joshua J Engelsma, Kai Cao, and Anil K Jain. 2019. Learning a Fixed-Length Fingerprint Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43, 6 (2019), 1981–1997.

[8] Alberto Ferrari and Marco Tartagni. 2002. Touchpad providing screen cursor/pointer movement control. US Patent 6,392,636.

[9] Martin Gerlach and Francesc Font-Clos. 2020. A Standardized Project Gutenberg Corpus for Statistical Analysis of Natural Language and Quantitative Linguistics. *Entropy* 22, 1 (2020), 126.

[10] François Guimbretière and Chau Nguyen. 2012. Bimanual Marking Menu for Near Surface Interactions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 825–828.

[11] Jiajing Guo, Andrew Benton, Nan Tian, William Ma, Nicholas Feffer, Zhengyu Zhou, and Liu Ren. 2023. EyeClick: A Robust Two-Step Eye-Hand Interaction for Text Entry in Augmented Reality Glasses. In *Adjunct Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. 1–4.

[12] Lawrence Alan Gust. 2006. Compact optical pointing apparatus and method. US Patent 7,102,617.

[13] Sean Gustafson, Daniel Bierwirth, and Patrick Baudisch. 2010. Imaginary Interfaces: Spatial Interaction with Empty Hands and Without Visual Feedback. In

Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology. 3–12.

[14] Sean G Gustafson, Bernhard Rabe, and Patrick M Baudisch. 2013. Understanding Palm-Based Imaginary Interfaces: The Role of Visual and Tactile Cues When Browsing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* 889–898.

[15] Taejin Ha, Steven Feiner, and Woontack Woo. 2014. WeARHand: Head-Worn, RGB-D Camera-Based, Bare-Hand User Interface with Visually Enhanced Depth Perception. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR).* IEEE, 219–228.

[16] Jooyeun Ham, Jonggi Hong, Youngkyoon Jang, Seung Hwan Ko, and Woontack Woo. 2014. Smart Wristband: Touch-And-Motion–Tracking Wearable 3D Input Device for Smart Glasses. In *Distributed, Ambient, and Pervasive Interactions: Second International Conference, DAPI 2014, Held as Part of HCI International 2014, Heraklion, Crete, Greece, June 22-27, 2014. Proceedings 2.* Springer, 109–118.

[17] Chris Harrison, Hrvoje Benko, and Andrew D Wilson. 2011. OmniTouch: Wearable Multitouch Interaction Everywhere. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology.* 441–450.

[18] Chris Harrison, Desney Tan, and Dan Morris. 2011. Skinput: Appropriating the Skin as an Interactive Canvas. *Commun. ACM* 54, 8 (2011), 111–118.

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

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity Mappings in Deep Residual Networks. In *Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV 14.* Springer, 630–645.

[21] Christian Holz and Patrick Baudisch. 2013. Fiberio: a touchscreen that senses fingerprints. In *Proceedings of the 26th annual ACM symposium on User interface software and technology.* 41–50.

[22] Andrew G Howard. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861* (2017).

[23] Yi-Ta Hsieh, Antti Jylhä, Valeria Orso, Luciano Gamberini, and Giulio Jacucci. 2016. Designing a Willing-to-Use-in-Public Hand Gestural Interaction Technique for Smart Glasses. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems.* 4203–4215.

[24] MD Rasel Islam, Doyoung Lee, Liza Suraiya Jahan, and Ian Oakley. 2018. GlassPass: Tapping Gestures to Unlock Smart Glasses. In *Proceedings of the 9th Augmented Human International Conference.* 1–8.

[25] Wolf Kienzle and Ken Hinckley. 2014. LightRing: Always-Available 2D Input on Any Surface. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology.* 157–160.

[26] Per Ola Kristensson and Shumin Zhai. 2007. Command Strokes With and Without Preview: Using Pen Gestures on Keyboard for Command Selection. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* 1137–1146.

[27] Lik-Hang Lee and Pan Hui. 2018. Interaction Methods for Smart Glasses: A Survey. *IEEE access* 6 (2018), 28712–28732.

[28] Chentao Li, Jinyang Yu, Ke He, Jianjiang Feng, and Jie Zhou. 2024. SwivelTouch: Boosting Touchscreen Input with 3D Finger Rotation Gesture. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 8, 2, Article 53 (may 2024), 30 pages. https://doi.org/10.1145/3659584

[29] Zongjian Liu, Jieling He, Jianjiang Feng, and Jie Zhou. 2023. PrinType: Text Entry via Fingerprint Recognition. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 4 (2023), 1–31.

[30] 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.* 220–229.

[31] Roderick McCall, Benoît Martin, Andrei Popleteev, Nicolas Louveton, and Thomas Engel. 2015. Text Entry on Smart Glasses. In *2015 8th International Conference on Human System Interaction (HSI).* IEEE, 195–200.

[32] Masa Ogata, Yuta Sugiura, Hirotaka Osawa, and Michita Imai. 2012. IRing: Intelligent Ring Using Infrared Reflection. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology.* 131–136.

[33] Anna Ostberg, Mohamed Sheik-Nainar, and Nada Matic. 2016. Using a Mobile Device Fingerprint Sensor as a Gestural Input Device. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems* (San Jose, California, USA) *(CHI EA '16).* Association for Computing Machinery, New York, NY, USA, 2625–2631. https://doi.org/10.1145/2851581.2892419

[34] Laxmi Pandey, Khalad Hasan, and Ahmed Sabbir Arif. 2021. Acceptability of Speech and Silent Speech Input Methods in Private and Public. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems.* 1–13.

[35] Jun Rekimoto. 2001. GestureWrist and GesturePad: Unobtrusive Wearable Interaction Devices. In *Proceedings Fifth International Symposium on Wearable Computers.* IEEE, 21–27.

[36] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 4510–4520.

[37] Dana Slambekova, Reynold Bailey, and Joe Geigel. 2012. Gaze and Gesture Based Object Manipulation in Virtual Worlds. In *Proceedings of the 18th ACM Symposium on Virtual Reality Software and Technology.* 203–204.

[38] Paul Streli, Jiaxi Jiang, Andreas Rene Fender, Manuel Meier, Hugo Romat, and Christian Holz. 2022. TapType: Ten-finger Text Entry on Everyday Surfaces via Bayesian Inference. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems.* 1–16.

[39] Atsushi Sugiura and Yoshiyuki Koseki. 1998. A user interface using fingerprint recognition: holding commands and data objects on fingers. In *Proceedings of the 11th annual ACM symposium on User interface software and technology.* 71–79.

[40] Takumi Toyama, Daniel Sonntag, Andreas Dengel, Takahiro Matsuda, Masakazu Iwamura, and Koichi Kise. 2014. A Mixed Reality Head-Mounted Text Translation System Using Eye Gaze Input. In *Proceedings of the 19th International Conference on Intelligent User Interfaces.* 329–334.

[41] Cheng-Yao Wang, Wei-Chen Chu, Po-Tsung Chiu, Min-Chieh Hsiu, Yih-Harn Chiang, and Mike Y Chen. 2015. PalmType: Using Palms as Keyboards for Smart Glasses. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services.* 153–160.

[42] Cheng-Yao Wang, Min-Chieh Hsiu, Po-Tsung Chiu, Chiao-Hui Chang, Liwei Chan, Bing-Yu Chen, and Mike Y Chen. 2015. PalmGesture: Using Palms as Gesture Interfaces for Eyes-Free Input. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services.* 217–226.

[43] David J Ward, Alan F Blackwell, and David JC MacKay. 2000. Dasher—A Data Entry Interface Using Continuous Gestures and Language Models. In *Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology.* 129–137.

[44] Eric Whitmire, Mohit Jain, Divye Jain, Greg Nelson, Ravi Karkar, Shwetak Patel, and Mayank Goel. 2017. DigiTouch: Reconfigurable Thumb-to-Finger Input and Text Entry on Head-Mounted Displays. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 3 (2017), 1–21.

[45] Zheer Xu, Pui Chung Wong, Jun Gong, Te-Yen Wu, Aditya Shekhar Nittala, Xiaojun Bi, Jürgen Steimle, Hongbo Fu, Kening Zhu, and Xing-Dong Yang. 2019. TipText: Eyes-Free Text Entry on a Fingertip Keyboard. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology.* 883–899.

[46] Xing-Dong Yang, Tovi Grossman, Daniel Wigdor, and George Fitzmaurice. 2012. Magic Finger: Always-Available Input Through Finger Instrumentation. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology.* 147–156.

[47] Shanhe Yi, Zhengrui Qin, Ed Novak, Yafeng Yin, and Qun Li. 2016. GlassGesture: Exploring Head Gesture Interface of Smart Glasses. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications.* IEEE, 1–9.

[48] Chun Yu, Ke Sun, Mingyuan Zhong, Xincheng Li, Peijun Zhao, and Yuanchun Shi. 2016. One-Dimensional Handwriting: Inputting Letters and Words on Smart Glasses. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems.* 71–82.

[49] Jinyang Yu, Jianjiang Feng, and Jie Zhou. 2023. PrintShear: Shear Input Based on Fingerprint Deformation. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 7, 2 (2023), 1–22.

[50] Mingrui Ray Zhang, He Wen, and Jacob O Wobbrock. 2019. Type, Then Correct: Intelligent Text Correction Techniques for Mobile Text Entry Using Neural Networks. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology.* 843–855.

[51] 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.* 1–16.

A Methodology

A.1 Detailed Fingerprint Identification Model

A.1.1 Model Architecture. To effectively extract features from fingerprint images, we employed a modified MobileNetV2 architecture. Originally designed for efficient image classification on mobile devices, MobileNetV2 uses depthwise separable convolutions and inverted residual blocks to achieve a balance between accuracy and computational cost. Given that the input fingerprint images are of size160 $\times$ 36, several adjustments were made to the standard MobileNetV2 architecture to better accommodate the specific characteristics of these images:

Adjustment of the initial convolutional layer: The stride of the first convolutional layer was reduced from 2 to 1. This modification helps preserve the spatial resolution of the feature maps in the early layers, which is crucial for capturing fine-grained fingerprint features.

Reduction of network depth: To match the smaller input size and control the model's complexity, we reduced the number of repetitions in certain stages of the network. Specifically, the number of repetitions in Block 2 was reduced from 2 to 1, and in Block 3, it was reduced from 3 to 2. The final two stages (stage 6 and stage 7) of the standard MobileNetV2 architecture were removed to further reduce the network's depth and computational cost.

Adjustment of output feature dimensions: The number of output channels in the penultimate convolutional layer was reduced from 1280 to 512, which decreases the model's parameter count and computational burden while maintaining its expressive power. The resulting network architecture is detailed in Table 10. The final output of this modified structure is a 192-dimensional feature vector, which is subsequently used for fingerprint matching.

A.1.2 *Training Procedure.*
We train the model using a combined loss function comprising two components: a classification loss and a triplet loss.

Classification Loss: The classification loss employs the standard cross-entropy loss function used in multi-class classification. For each input fingerprint image, the network predicts a probability distribution over all registered identities. The cross-entropy loss encourages the network to assign a high probability to the correct identity and low probabilities to the remaining identities.

Triplet Loss: For robust finger identification, we aim to learn feature representations where fingerprints from the same finger are highly similar and fingerprints from different fingers are dissimilar. Triplet loss is particularly well-suited for this objective. It utilizes triplets of fingerprint images: an anchor image, a positive image (belonging to the same identity as the anchor), and a negative image (belonging to a different identity). The goal is to minimize the distance between the feature vectors of the anchor and positive images while maximizing the distance between the anchor and negative images. This directly encourages the network to learn embeddings where similarity in feature space reflects fingerprint identity. While cosine similarity is a common metric for comparing fingerprint features, triplet loss with Euclidean distance implicitly optimizes for a feature space where Euclidean distance also effectively reflects similarity for identification.

The triplet loss function can be defined as follows:

$$L_{\text{triplet}} = \max\left(0, \|f(a) - f(p)\|_2^2 - \|f(a) - f(n)\|_2^2 + \alpha\right) \quad (3)$$

where $f(a)$ represents the feature vector of the anchor image. $f(p)$ represents the feature vector of the positive image. $f(n)$ represents the feature vector of the negative image. $\|.\|_2^2$ represents the squared Euclidean distance. α is a margin parameter that enforces a minimum distance between positive and negative pairs.

By minimizing the triplet loss, the network learns to generate compact feature representations for fingerprints belonging to the same identity while pushing apart those belonging to different identities.

A.1.3 *Combined Loss.*
The overall loss function is a weighted sum of the classification loss and the triplet loss:

$$L_{\text{total}} = \lambda_1 * L_{\text{classification}} + \lambda_2 * L_{\text{triplet}} \quad (4)$$

where λ_1 and λ_2 are weighting factors that control the relative importance of each loss component.

During training, we optimize the model parameters to minimize this combined loss function using stochastic gradient descent with momentum. This process encourages the network to learn both discriminative features for classification and robust representations suitable for fingerprint matching based on feature vector similarity.

A.2 Finger Gesture Recognition

A.2.1 ***Rule-Based Gesture Recognition.***
This method employs image processing techniques and a direction determination algorithm to analyze a sequence of images, identifying gestures based on motion patterns. The process involves calculating frame-by-frame displacement, centroid shifts, and angle changes to determine the direction of image motion. The following steps outline the process:

Image Preprocessing. Convert the input image to grayscale and Apply Gaussian blur to reduce noise and smooth the image:

$$\text{blurred} = \text{GaussianBlur}(\text{image}, (5,5), 0) \quad (1)$$

Gradient Calculation. Compute the horizontal (Gx) and vertical (Gy) gradients of the blurred image using the Sobel operator:

$$G_x = \text{Sobel}(\text{blurred}, CV\_64F, 1, 0, \text{ksize} = 3) \quad (2)$$

$$G_y = \text{Sobel}(\text{blurred}, CV\_64F, 0, 1, \text{ksize} = 3) \quad (3)$$

Displacement Calculation. Translation: Calculate the translation by shifting the image and comparing the difference with the original:

$$\begin{aligned}\text{shifted\_img} = \text{warpAffine}(&\text{img}, M, (\text{width}, \text{height}),\\ &\text{borderMode} = \text{BORDER\_WRAP})\end{aligned} \quad (4)$$

Compute the sum of absolute differences for the difference image in the left, middle, and right regions:

$$\text{diff\_sum} = \sum \text{absdiff}(\text{shifted\_img}, \text{img\_2}) \quad (5)$$

Centroid Shift: Determine the difference in centroid coordinates between consecutive frames:

$$\Delta y_{\text{center}} = y_{\text{centroid2}} - y_{\text{centroid1}} \quad (6)$$

$$\Delta x_{\text{center}} = x_{\text{centroid2}} - x_{\text{centroid1}} \quad (7)$$

Angle Calculation. Calculate the average angle for each region (left, middle, right) using the arctangent function on the gradients:

$$\theta_{\text{left}} = 0.5\,\text{atan2}\left(2\sum G_x \cdot G_y, \sum G_x^2 - \sum G_y^2\right) \quad (8)$$

$$\theta_{\text{middle}} = 0.5\,\text{atan2}\left(2\sum G_x \cdot G_y, \sum G_x^2 - \sum G_y^2\right) \quad (9)$$

$$\theta_{\text{right}} = 0.5\,\text{atan2}\left(2\sum G_x \cdot G_y, \sum G_x^2 - \sum G_y^2\right) \quad (10)$$

Table 10: Modified MobileNetV2 Architecture for Fingerprint Feature Extraction

Layer	Input Size	Operator	Stride	Expansion Ratio	Output Channels
Input	160×36×3	Conv2D	1	-	16
Inverted Residual Block 1 (Repeat × 1)					
	-	Expansion (1×1 Conv2D)	-	1	-
	-	Depthwise Convolution (3×3)	1	-	-
	-	Projection (1×1 Conv2D)	-	-	16
Inverted Residual Block 2 (Repeat × 1)					
	-	Expansion (1×1 Conv2D)	-	6	-
	-	Depthwise Convolution (3×3)	2	-	-
	-	Projection (1×1 Conv2D)	-	-	24
Inverted Residual Block 3 (Repeat × 2)					
	-	Expansion (1×1 Conv2D)	-	6	-
	-	Depthwise Convolution (3×3)	2	-	-
	-	Projection (1×1 Conv2D)	-	-	32
Inverted Residual Block 4 (Repeat × 4)					
	-	Expansion (1×1 Conv2D)	-	6	-
	-	Depthwise Convolution (3×3)	1	-	-
	-	Projection (1×1 Conv2D)	-	-	64
Inverted Residual Block 5 (Repeat × 3)					
	-	Expansion (1×1 Conv2D)	-	6	-
	-	Depthwise Convolution (3×3)	2	-	-
	-	Projection (1×1 Conv2D)	-	-	96
	-	Expansion (1×1 Conv2D)	-	6	-
	-	Depthwise Convolution (3×3)	1	-	-
	-	Projection (1×1 Conv2D)	-	-	512
	-	Global Average Pooling	-	-	-
	-	Fully Connected	-	-	192

where

$$\operatorname{atan2}(y,x)=\begin{cases}\arctan\left(\frac{y}{x}\right), & \text{if } x>0\\ \arctan\left(\frac{y}{x}\right)+\pi, & \text{if } x<0 \text{ and } y\geq 0\\ \arctan\left(\frac{y}{x}\right)-\pi, & \text{if } x<0 \text{ and } y<0\\ \frac{\pi}{2}, & \text{if } x=0 \text{ and } y>0\\ -\frac{\pi}{2}, & \text{if } x=0 \text{ and } y<0\\ \text{undefined}, & \text{if } x=0 \text{ and } y=0\end{cases}$$

Motion Type Determination

- Up, Down, Left, Right Sliding: Determine sliding direction based on the magnitude and direction of the total translation:

$$\text{if } |total\_y\_shift| > |total\_x\_shift| :$$

$$\begin{cases}\text{if } total\_y\_shift > 0, \text{direction = "Down"}\\ \text{else, direction = "Up"}\end{cases} \tag{11}$$

$$\text{else :}$$

$$\begin{cases}\text{if } total\_x\_shift > 0, \text{direction = "Left"}\\ \text{else, direction = "Right"}\end{cases} \tag{12}$$

- Left, Right Rolling: Determine rolling direction through the direction of centroid shift if translation is minimal:

$$|total\_y\_shift| < 5 \text{ and } |total\_x\_shift| < 5 \tag{13}$$

$$\begin{cases}\text{if } total\_x\_center\_shift < 0, \text{ direction = "Left Roll"}\\ \text{else, direction = "Right Roll"}\end{cases} \tag{16}$$

- Clockwise, Counter-clockwise Rotation: Determine the rotation direction by computing the angular change in multiple regions and consolidating the results:

$$\Delta\theta_i = \theta_{i+1} - \theta_i \tag{17}$$

$$\text{bias} = \begin{cases}\text{if } \Delta\theta_i > 2.5, \text{ bias} -= \pi\\ \text{if } \Delta\theta_i < -2.5, \text{ bias} += \pi\end{cases} \tag{18}$$

$$\theta_{\text{end}} = \theta_{\text{end}} + \text{bias} \tag{19}$$

$$\text{if } \theta_{\text{end}} - \theta_{\text{start}} > 0 : \text{ direction = "Counter-clockwise Rotate"} \tag{20}$$

$$\text{else : direction = "Clockwise Rotate"} \tag{21}$$

$$\text{direct\_l} = \text{get\_direction}(\theta_{\text{left}}) \tag{22}$$

$$\text{direct\_m} = \text{get\_direction}(\theta_{\text{middle}}) \tag{23}$$

$$\text{direct\_r} = \text{get\_direction}(\theta_{\text{right}}) \tag{24}$$

$$\text{direction} = \text{most\_common}(\text{direct\_l}, \text{direct\_m}, \text{direct\_r}) \tag{25}$$

A.2.2 ***Detailed CNN and LSTM Architectures.*** To achieve efficient and effective gesture recognition, we enhance the CNN architecture by incorporating concepts from MobileNet, such as depthwise separable convolutions, which help reduce the number of parameters while maintaining performance. The LSTM architecture is designed to handle the temporal aspects of gesture recognition, outputting probabilities for each gesture class.

Table 11: LSTM Architecture

Parameter	Value
Number of LSTM layers	2
Hidden state dimension	256
Input dimension	128 (from CNN feature vector)
Activation function	tanh
Output dimension	8

CNN: The CNN architecture starts with an initial standard convolution layer (Conv1) for basic feature extraction. This is followed by two depthwise separable convolutional blocks (DepthwiseConv + PointwiseConv) to efficiently learn spatial hierarchies. Each depthwise separable convolution block is followed by a max-pooling layer to reduce the spatial dimensions of the feature maps. The output is then flattened and passed through a dense layer, resulting in a feature vector of size 128 that is suitable as input for the LSTM.

LSTM: The LSTM network is designed with two layers, each with a hidden state dimension of 256, to capture the temporal dependencies in the gesture sequences. The input to the LSTM at each time step is the 128-dimensional feature vector generated by the CNN. The final output layer has 8 units, corresponding to the 8 gesture classes.

Rotation: Left, Right ,

Sliding: Up, Down, Left, Right ,

Rolling: Left, Right.

The LSTM outputs a probability distribution over these 8 classes for each input sequence, enabling accurate classification of the finger gestures.

Table 12: CNN Architecture

Layer	Type	Filter Size	Stride	Activation	Output Channels
Conv1	Convolutional	3×3	1	ReLU	32
DepthwiseConv1	Depthwise Separable Convolution	3×3	1	ReLU	32
PointwiseConv1	1×1 Convolution	-	1	ReLU	64
MaxPool1	Max Pooling	2×2	2	-	-
DepthwiseConv2	Depthwise Separable Convolution	3×3	1	ReLU	64
PointwiseConv2	1×1 Convolution	-	1	ReLU	128
MaxPool2	Max Pooling	2×2	2	-	-
Flatten	-	-	-	-	-
Dense	Fully Connected	-	-	ReLU	128