MagicInput: Training-free Multi-lingual Finger Input System using Data Augmentation based on MNISTs

Hao Pan*
Shanghai Jiao Tong University
panh09@sjtu.edu.cn

Yi-Chao Chen*
Shanghai Jiao Tong University
yichao@sjtu.edu.cn

Qi Ye
Shanghai Jiao Tong University
yeqi3614@sjtu.edu.cn

Guangtao Xue†
Shanghai Jiao Tong University
gt_xue@sjtu.edu.cn

ABSTRACT

Text input systems based on device-free finger tracking technologies have attracted considerable attention in the use scenarios of mobile and the Internet-of-Things (IoT) devices. Issues pertaining to 2D tracking have prompted interest in using 1D finger trajectories for the recognition of handwritten letters. Nonetheless, 1D tracking imposes two major challenges: (i) Trajectory information loss from 2D to 1D; and (ii) Inter-user diversity in writing traits. These challenges could possibly be overcome by collecting a large training dataset for every user; however, this would impose an unacceptable burden on users. This paper presents a text input system with multi-language support without training using acoustic-based 1D finger tracking technology. We developed a novel data augmentation scheme, in which the handwritten image dataset MNISTs are used to create artificial datasets (called *Track*MNISTs). We compensate for the trajectory information loss of 1D by creating personal dataset (from *Track*MNIST) to match the writing habits of individual users. The proposed data augmentation mechanism is also applicable to multilingual letter recognition. In experiments, *MagicInput* achieved outstanding classification accuracy on unseen users: 10 digits (98.3%), 26 uppercase/lowercase English letters (97.8%/95.3%), 49 Japanese characters (91.4%), and the 30 commonly used Chinese characters (93.8%).

*Both authors contributed equally to this research.
†Guangtao Xue is the corresponding author.

IPSN '21, May 18–21, 2021, Nashville, TN, USA

ACM ISBN 978-1-4503-8098-0/21/05…$15.00
https://doi.org/10.1145/3412382.3458261

CCS CONCEPTS

• **Human-centered computing** → **Text input**; *Interaction design process and methods*.

KEYWORDS

Text input system; training-free; multi-language support

ACM Reference Format:
Hao Pan, Yi-Chao Chen, Qi Ye, and Guangtao Xue. 2021. *MagicInput*: Training-free Multi-lingual Finger Input System using Data Augmentation based on MNISTs. In *The 20th International Conference on Information Processing in Sensor Networks (co-located with CPS-IoT Week 2021) (IPSN '21), May 18–21, 2021, Nashville, TN, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3412382.3458261

1 INTRODUCTION

Motivation: Text input systems based on device-free finger tracking technologies have attracted considerable attention for mobile devices. Finger tracking technology based on acoustic signals has been shown to achieve high precision using the built-in speaker and microphone. The known audio signals (at inaudible frequencies) are transmitted via a mobile, reflected signal by the moving finger will be received and be analyzed to track the finger location over time.

Currently, to obtain the distance between the finger and the mobile, mainstream acoustic-based tracking technologies calculate the initial absolute distance and update the value by tracking the distance changes associated with the movement of the finger [23, 32]. And the state-of-the-art work (*e.g.*, Strata) can limit distance estimation error to less than 3*mm* in 1D tracking, and coordinate positioning error of 0.6*cm* in the 2D space during the preliminary duration [32]. Note that this approach imposes cumulative errors pertaining to distance change, such that the absolute distance error increases over time. Even a tiny error in estimating the absolute distance can lead to grievous errors in the 2D positioning coordinates. When acoustic-based tracking system is used for extended durations, the absolute distance estimation error increases to

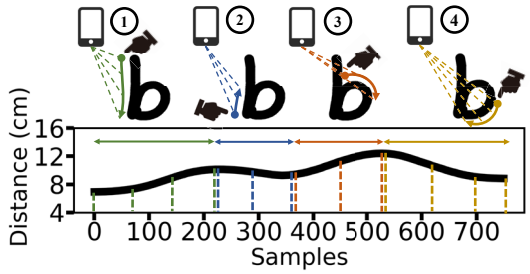


Figure 1: Exemplar 1D tracking trajectory while the user writes a letter 'b' in the air.

1.3cm in 1D tracking, and 7.9cm in 2D coordinate positioning. Calibration mechanisms can be used for the cancellation of accumulated error at regular intervals; however, this requires that the user's finger remain stationary for an extended period, thereby undermining the user experience. Therefore, 1D acoustic-based finger tracking is the better for high precision and stability over time.

Fig. 1 is an example of 1D tracking trajectory as the user writes the letter 'b' in the air. The finger's 1D position is continuously tracked and recorded as a specific time-series pattern determined only by variations in the relative distance. Overall, 1D tracking provides three notable advantages for the recognition of handwriting: (1) Only one microphone and one speaker are required; (2) Errors in distance change estimation are negligible (1.3mm), making it ideal for tracking process over an extended duration; (3) Estimates of distance change based on acoustic signal phase variations impose low computational overhead, thereby lowering energy consumption.

Challenges: The use of 1D tracking for handwriting recognition also imposes two major challenges: (i) Different users have different writing habits and characteristics. Unless the system has access to a large-scale training dataset related to the writing traits of an enormous number of users, the trained classifiers will lack robustness when dealing with unknown users. Furthermore, collecting training data from every target user would be exceedingly difficult and negatively impact the user experience. (ii) Unlike 2D tracking, 1D tracking discards all characteristics of handwritten letters except for the distance change. This makes it difficult to develop an algorithm for the classification of letter traces with high precision for the majority of users.

Proposed Methods: In this paper, we sought to overcome the lack of large-scale training datasets by developing a data augmentation scheme based on the MNIST-series datasets (hereafter referred to as MNISTs). The proposed scheme involves generating artificial datasets (referred to as *Track*MNISTs), comprising 1D finger tracking patterns converted from the handwritten images in MNISTs. We designed data augmentation scheme into three steps: (i) We developed a stroke-sequence restoration algorithm to reverse engineer the actual handwriting process from which the images in MNIST were derived. (ii) We simulated the trajectory of a finger jumping to indicate a disconnection between strokes when writing letters in the air. (iii) We augmented the dataset with data simulating other personal writing characteristics, such as writing speed and writing position. Note that the MNIST-series includes many datasets of digital images covering a wide variety of language types and rich writing styles. Thus, the proposed data enhancement scheme is applicable to MNISTs of multiple languages. We also sought to overcome the difficulties involved in designing a high-precision and robust classifier for the unseen users. This was achieved by tailoring a dataset specifically for each target user (*i.e.*, one that best matches their writing characteristics) in conjunction with a simple classifier to enhance letter trace classification performance.

A prototype of our proposed text input system (called *MagicInput*) was implemented on the Android phones equipped with the Strata system [32] for 1D finger tracking. We then conducted extensive experiments aimed at evaluating the effectiveness and generalizability of the proposed text input system to multiple languages and a diversity of unseen users. In experiments, *MagicInput* achieved outstanding classification accuracy: 10 digits (98.6%) 26 lowercase English letters (97.8%), 26 uppercase English letters (95.3%), 49 Japanese characters (91.4%), and the 30 most commonly used Chinese characters (93.8%).

Contribution: The contributions of this work are summarized as follows:

- We developed a finger text input system with multi-language support without training, based on an innovative data enhancement scheme involving the generation of an artificial dataset (*Track*MNIST) from MNIST-series datasets.
- We developed a stroke-sequence restoration algorithm aimed at simulating the handwriting process underlying the images in the MNISTs. We also simulated the specific trajectory of finger jumping at disconnection between strokes, as well as the various writing characteristics (*e.g.*, font size, writing speed and position).
- To overcome interference associated with the characteristics of multiple users, we developed an online matching mechanism to compile from *Track*MNISTs a personalized dataset with a high similarity to the writing characteristics of the target user. Combining a personalized dataset with a simple classifier model was shown to greatly improve classification accuracy.
- A prototype of the *MagicInput* implemented on Android phones demonstrated the effectiveness and generalizability of the proposed system in recognizing letters and characters written by unseen users in a variety of languages.

2 RELATED WORK

2.1 Vision-based Text Input Methods

The standard approach to vision-based text recognition involves analyzing a image of handwritten text in real time by extracting character-related features for classification via machine learning models or neural network-based deep learning methods, and can achieve character-level accurate rate (95.88% $\sim$ 99.96%) on different languages after model training through enormous labeled datasets [20, 21, 29]. Note however that vision-based methods are highly susceptible to ambient, they raise issues pertaining to privacy, and impose heavy computational burden.

2.2 RF-based Text Input Methods

RF-based device-free tracking schemes have also attracted considerable attention [1, 11, 14, 22, 24]. Systems based on Channel State Information (CSI) have proven highly effective in guiding human-computer interactions; however, the use of RF technology limits the detection resolution to the centimeter scale, which makes it difficult to differentiate among fine finger movements. WriFi uses WiFi signals for continuous wireless input upper English letter recognition (with 86% accuracy) based on PCA and FFT [8]. WiReaderr is an adaptive system that uses the CSI of commercial WiFi systems for handwriting recognition on 26 uppercase English letters (with 90.64% accuracy) [9]. However, this requires that users produce obvious body movements (*e.g.*, raise arms) to input handwriting and cannot support lower English letters, which can be awkward in many social situations.

2.3 Acoustic-based Text Input Methods

Many researches have used low-cost acoustic signals for the input of handwriting. Acoustic-based methods can be divided into those that rely on tracking and those that rely on audible sounds. A microphone is used to receive acoustic signals reflected from the hand/finger as it traces symbols in the air [17, 18, 23, 32]. Examples include LLAP [23] using phase-based tracking and FingerIO [18] using OFDM symbol based movement detection. By contrast, the Strata [32] system used in the current work is the state-of-the-art acoustic tracking scheme with the minimum absolute distance estimation error, which combines the absolute initial and relative distance estimates to accurately track the moving target.

A number of methods have been developed to identify handwritten letters by analyzing the sound produced as a finger or stylus is brushed across a hard surface [7, 13, 31, 33]. Note however that these methods work only with specific surfaces and cannot handle with multiple users. Writinghacker [31] and WordRecoder [7] relies on training data collected from multiple users; however, recognition accuracies on unseen users are not satisfied. The handwriting system in [13] must be retrained for every user. And the system in [33] uses three template matching schemes to identify cursive letters; however, it is limited to a single user.

2.4 Sensor-based Text Input Methods

Sensor-based methods are meant to capture and characterize motions associated with the task of handwriting [3, 6, 27, 30]. For example, UbiTouch identifies handwritten characters by sensing the movement of the user's hand in the proximity of a light sensor (with character recognition rate of 79%) [27]. iRing senses the rotation and bend angle of the user's finger using an infrared sensor embedded in a ring [19]. Some methods apply bioelectrical sensors to the skin of human body as an input interface [12, 25, 26]; other systems use the accelerometers or gyroscopes built into mobile devices to recognize hand movement; however, waving the device through the air tends to be awkward.

3 PRELIMINARY STUDY

In this section, we first compare the feasibility of 2D and 1D tracking for letter classification. We then discuss the impact of training datasets on the performance of the letter trace classification models.

3.1 2D Tracking vs. 1D Tracking

We implemented the acoustic-based finger tracking algorithm, Strata, on a Huawei Honor 9, to enable the tracking of finger movements. The performance of the Strata system in 2D tracking was evaluated using the experimental testbed in Figs. 2. A volunteer was tasked with writing the letter 'a' 20 times repeatedly with a finger on a Surface screen, wherein the images recorded on the touchscreen were used as the ground truth. Meanwhile, the 2D tracking system recorded the finger movement in real time. A portion of the tracking results are presented in Fig. 3, we found that the positioning error across the entire 2D plane eventually reached $7.9cm$, by which point the trajectory was no longer representative of the letter 'a'.

The next step involved using a microphone and speaker to perform 1D finger tracking via the Strata. Using the experiment testbed in Fig. 4, we displayed standard letters ('a'-'d') on a Surface and tasked the volunteer with writing each letter 50 times in accordance with the standard shape in the standard stroke order. Meanwhile, the Strata 1D tracking system recorded the trajectory patterns in real time. The normalized results are presented in Fig. 5. The black bold line (the ground truth) in each sub-graph indicates the simulated normalized change in distance between the point of origin and each node in the standard letter image. Despite the errors in 1D tracking, the tracking pattern of each letter did not vary. Moreover, the tracking patterns for any given letter did not vary over time, thereby demonstrating the stability of the system.

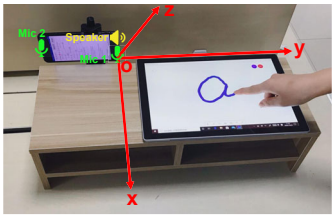
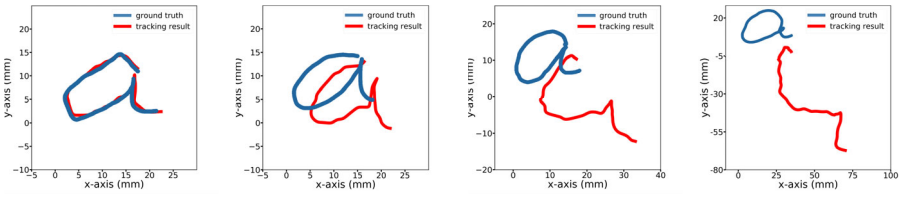


Figure 2: Testbed setup for the Strata 2D finger tracking.



(a) The 1st time. (b) The 3rd time. (c) The 10th time. (d) The 20th time.

Figure 3: Examples of tracking the 2D trajectories involved in the repetitive movement of fingers writing the letter 'a'.

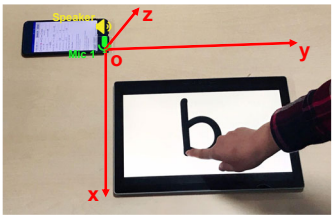
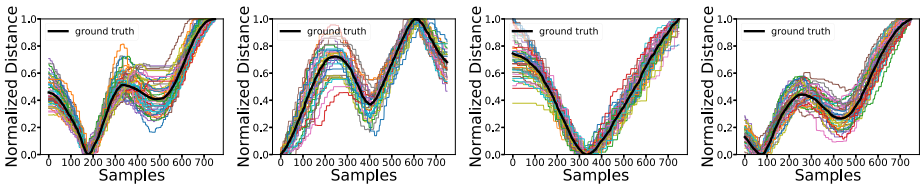


Figure 4: Testbed setup for the Strata 1D finger tracking.



(a) Writing "a". (b) Writing "b". (c) Writing "c". (d) Writing "d".

Figure 5: Comparison of ground truth finger trajectories and actual finger trajectories in terms of tracking errors: a) letter 'a'; b) letter 'b'; c) letter 'c'; d) letter 'd'.

3.2 1D Trajectory Classification

Letter trace data collection: Three volunteers were tasked with inputting examples of their handwriting in their own writing styles into a Surface. This task involved writing 26 lowercase letters a total of 50 times each, during which the Strata collected 1D distance information of the associated finger movements. To compensate for variations in writing speed among the volunteers, the resulting data was normalized to cover the same length, thereby ensuring that the classifier obtained inputs of equal dimensions.

Performance of classification algorithms: The letter trace datasets obtained from the three volunteers were normalized and then fed into various time-series classification algorithms: k-nearest neighbor (KNN), logistic regression (LR), support vector machines (SVM), naive bayes (NB), random forest (RF), adaBoost (AB) and long short-term memory network (LSTM). The corresponding classification accuracy results (after cross-validation) are listed in Tab. 1. All of these mainstream classifiers performed well; however, RF and AB achieved the highest accuracy of roughly 90%.

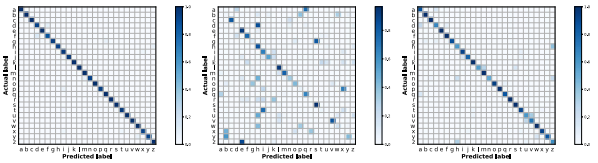
Classifier	KNN	LR	SVM	NB	RF	AB	LSTM
ACC(%)	88.5	88.9	87.2	88.3	91.3	91.5	87.2

Table 1: Results of canonical classifiers in recognizing hand-written letters based on finger trajectories.

Impact of training dataset: In most application scenarios of HCI systems, the trained classifier cannot achieve satisfied classification results on untrained/unseen users. A number of researchers have developed the feature-engineering or transfer learning methods to resolve the impact of cross-user on the classification tasks. The feature-engineering method is inapplicable to the current study, due to the fact that the dimensionality reduction associated with the use of 1D tracking leads to information loss. Transfer learning methods (1) need a large amount of data in both source field and target field, or (2) involve the inclusion of expert rules to alter the direction of the prediction model (zero-shot scenarios), and cannot be applied in the current scenario. In this paper, we try to solve this problem with the model selection mechanism by generating a personalized dataset for each untrained user.

Here, we discuss the impact of various training datasets on the classification of handwritten letter traces in terms of accuracy. We first trained a RF classifier (as RF1) with the volunteer 1's dataset. Volunteer 1 then wrote the same letters again (testing data) to verify the performance. The classification accuracy of model RF1 reached 97.5%, the corresponding confusion matrix is shown in Fig. 6(a). Similarly, we trained model RF2 with the volunteer 2's dataset. When using testing data from volunteer 1, the classification accuracy of RF2 reached 34.8%, the confusion matrix is shown in Fig. 6(b). Finally, we trained model RF3 with both volunteer 1 and 2's dataset, and again used testing data from volunteer 1 for analysis. The classification accuracy of model RF3 reached 79.3%, the confusion matrix is in Fig. 6(c).

In classifying the letter traces of a specific user (*e.g.*, volunteer 1), the performance of RF1 far exceeded that of RF2 and RF3. This can be attributed to the fact that the training of RF1 is perfectly suited to the specific writing traits of that user. By contrast, the training of RF2 was entirely unsuited to the writing traits of volunteer 1, whereas RF3 had to account for



(a) RF1 (97.5%). (b) RF2 (34.8%). (c) RF3 (79.3%).

Figure 6: Confusion matrices (with average accuracy) of three RF models trained from different datasets.

the writing traits of all three users, resulting in a more general model. Overall, these results indicate that as long as we have a training dataset that matches the target user's writing traits, then a classifier can be tailored specifically to that user to optimize classification performance.

4 DATA AUGMENTATION

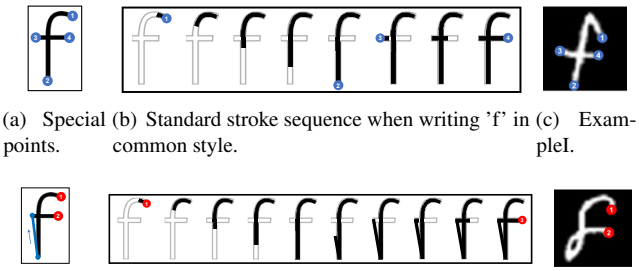
In this section, we detail the proposed data augmentation scheme involving the conversion of the MNISTs into an artificial letter tracking trace datasets (*Track*MNISTs).

4.1 Simulating Traces from MNIST Images

Creating *Track*MNIST for letter classification required the transformation of every still image in the MNIST into trajectories mimicking the movement of user's finger through the air while writing the letter represented in the image. The data transformation process was divided into two steps: 1) Determine the writing direction of each stroke as well as the stroke sequence; and 2) Simulate the trajectory of the jumping motion performed by the finger when the user encounters disconnected strokes. Note that the default stroke sequences simulated in this paper matched the standard stroke sequence taught in most schools, regardless of language.

4.1.1 Restoring Stroke Order Sequences. In the following, lowercase English letters were used as examples to illustrate the pipeline of restoring stroke order sequences. We downloaded a video for the standardized instruction of writing lowercase English letters from the Teach Handwriting website [10]. By carefully analyzing each frame in the video, we were able to extract the special points (Fig. 7(a)) as well as the specific direction of each stroke (Fig. 7(b)).

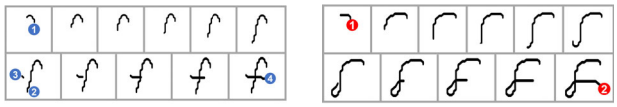
Fig. 7(c) presents an example written by a volunteer in accordance with the standard writing protocol for the common style of the letter 'f' in Figs. 7(b) and 7(a). It was observed however that some of the volunteers employed a continuous-stroke writing habits instead of the common style. Fig. 7(f) presents an example written by a volunteer in accordance with the standard writing protocol for the continuous-stroke style in Figs. 7(e) and 7(d). For each image in each writing style, we firstly involved identifying the special points (start and



(a) Special points. (b) Standard stroke sequence when writing 'f' in common style. (c) ExampleI.

(d) Special points. (e) Standard stroke sequence when writing 'f' in continous-stroke style. (f) ExampleII.

Figure 7: Start and end points and stroke sequence involved in writing a standard lower case English 'f'.



(a) Simulated strok sequence of handwritten 'f' in common style. (b) Simulated strok sequence of 'f' in continous-stroke style.

Figure 8: Examples of stroke sequences simulated from still handwritten images in the EMNIST.

end points) based on a common characteristic. Note that the degree value (*i.e.*, the number of edges incident to the vertex) was odd. Thus, we skeletonized the handwritten image, traversed each node, calculated its degree value, and then filtered out the nodes whose degree value was odd, the coordinates of which were then recorded in the ***Start point set*** S_n. If the S_n is empty, we sampled some points uniformly and recorded them in S_n. We also defined the corresponding ***Direction set*** D_n to collect the standardized direction in each timestamp.

We developed a stroke sequence algorithm to apply stroke sequence information to still images in the MNISTs, and Fig. 8 shows the results of stroke order sequences of the two examples 'f'. The proposed algorithm was implemented using particle swarm optimization, whose core idea is to use a group of particles to traverse the binarized images. We described the details as following:

- Step 1: We select one start point from the S_n.
- Step 2: We initialize a certain number of particles at the start point, and initialize the movement direction with Gaussian distribution ($N(\mu, \sigma^2)$) for each particle, where the expected value (μ) is set as the first direction value in *Direction set* D_n and σ is set as 10°.
- Step 3: The particle swarm start to swim with the defined direction. If the particle moves outside the letter, it is eliminated; otherwise, it is lived until the next timestamp. If there exists surviving particles in the first timestamp, we go to Step 4. Otherwise, we return back to Step 1 and select another start point.

- Step 4: At the following timestamp, we defined the corresponding wandering direction for the surviving particle swarm according to the D_n. If we traversed the whole direction set, and there existed surviving particles, we record their moving trajectories and collect them in the ***Simulated stroke sequence set*** T and go to Step 6. If the direction set has not been traversed and the whole particles were eliminated, it means there exist discontinuous strokes and we go to Step 5.
- Step 5: We select a new point from *Start point set* as the start point of the next stroke, and migrated the living particles to the new start point and repeat Step 3.
- Step 6: We execute the average processing on all traces in the T to obtain the simulated writing movement behavior of the current handwritten letter picture.

4.1.2 Simulating Finger Jumping Traces Between Disconnected Strokes. Whenever a letter is written using a discontinuous stroke, the finger will tend to jump when the user encounters a disconnection between strokes. This jumping motion cannot be identified in the handwritten images; however, an acoustic-based tracking system should be able to detect this type of characteristic motion. Take for example an 'f' written in the common style (left side of Fig. 9), where the user's finger departs from the plane on which the letter was written (at the end point of one stroke) before returning to the same plane (at the start of the next stroke).

The trace left by the movement of the user's finger presents a continuous curve as it departs from and returns to the writing plane (see Fig. 9). Note that users differ in their jump trajectories and even the same user will vary in their jump trajectories according to the preceding and/or following stroke. To overcome this variability, we employed a second-order Bezier curve to simulate the trace left by a user's finger when encountering a jump from one stroke to the next.

The principle of the second-order Bezier curve is illustrated in Fig. 10. The second-order curve describes the curve using two data points $A(x_1, y_1)$ and $C(x_3, y_3)$ and one control point $B(x_2, y_2)$. We try to identify points D, E and F respectively on lines AB, BC and DE, $\frac{AD}{AB} = \frac{BE}{BC} = \frac{DF}{DE}$. Thus, all of the trajectories of F form a Bezier curve, and the coordinate relationship of point $F(x_f, y_f)$ can be described as follows:

$$\begin{cases} x_f = (1-t^2)x_1 + 2t(1-t)x_2 + t^2 x_3, \\ y_f = (1-t^2)y_1 + 2t(1-t)y_2 + t^2 y_3 \end{cases} \quad (1)$$

where $t \in [0, 1]$. When using a second-order Bezier curve to simulate a trajectory, the landing point of the previous stroke is designated A in the jump plane, and the start point of the next stroke is designated B. We then set the coordinate value of C randomly to generate multiple curves passing through A and B to simulate the trajectory of the finger movements of multiple users during the jump between strokes.

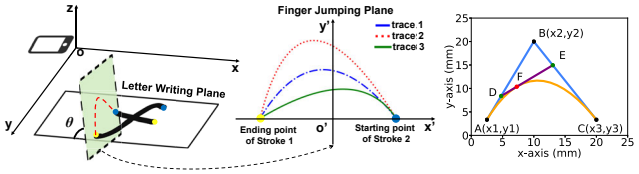


Figure 9: Example of stroke discontinuity when writing the letter 'f'.

Figure 10: 2nd Bezier curve.

Note that in addition to differences in the shape of traces related to the jump plane, there are also differences in the angle (θ in Fig. 9) between the finger jumping plane and letter writing plane. We simulate the jump characteristics specific to particular users by varying the value of θ.

4.2 Simulating Other Writing Characteristics

As shown on the right side of Fig. 9, we assume that the tracking device begins at the origin of the three-dimensional (3D) spatial coordinate system, and XOY plane is parallel to the writing plane. Using the known 3D coordinates of each point in the letter and the trajectory of finger jumps, it is relatively easy to calculate the distance between the origin and each point along the trajectory of the moving finger in accordance with the corresponding stroke sequence. We can generate an artificial dataset of traces collected by the tracking system when the user writes letters with his finger.

The above-mentioned method can only be used to represent inter-user differences in writing style. Nonetheless, there are three other characteristics that can affect the tracking of finger trajectories: letter size, writing speed, and the position of the finger relative to the mobile device.

Size of finger-written letters: The size of a letter written in the air can affect the tracking, unless the finger moves at different speed, such that the size affects only the time to completion. In generating the artificial dataset, we first sought to normalize the length of the traces via up-/down-sampling. In so doing, we were able to eliminate the effect of letter size on tracking traces, and thereby disregard the issue of letter size in augmenting the personal datasets.

Speed of finger writing: As shown in Fig. 1, the difference in writing speed can affect the slope (or pattern) of the tracking traces. Any variation in the speed of the finger while drawing a single stroke can profoundly affect the resulting trace pattern. To make the artificial dataset robust to variations in writing speed, we changed the number of points associated with each stroke trajectory and each jump trajectory. Increasing the number of points in a given trajectory simulates faster writing, and vice versa.

Position of finger relative to mobile device: Altering the position of the user's finger relative to the mobile device could alter the resulting tracking trace. As shown in the 3D coordinate system in Fig. 9, we simulated the two finger positions

Figure 11: Standard stroke sequence of 'lun'.

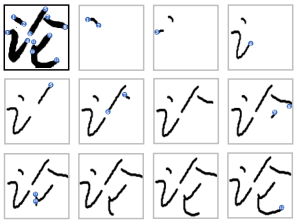


Figure 12: Simulated stroke sequence of 'lun'.

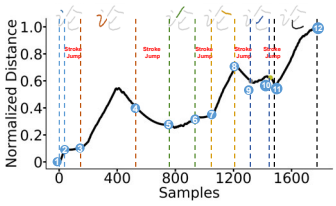


Figure 13: Example of artificial 1D tracking pattern when writing 'lun'.

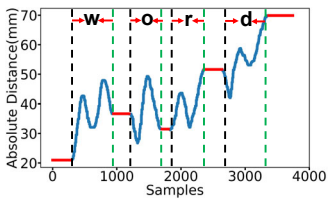


Figure 14: Example of letter segmentation when input a word: 'word'.

relative to the mobile device: XOY plane and $(-X)OY$ plane, instead of the four positions considering spatial symmetry. Thanks to the normalization operation, there was no need to consider the initial distance between the finger and the device.

4.3 Multi-language Support

The simulated tracking trace data corresponding to images in the EMNIST dataset is applicable to writing of almost any kind, such as the 10 digits, the 26 uppercase and lowercase English letters, and even written languages of high complexity, such as Chinese and Japanese. In this section, we use Chinese characters as an example to verify the universality of the proposed tracking trace simulation scheme.

We first downloaded a video for the standardized instruction of writing Chinese characters from the website [2]. By carefully analyzing each frame, we were able to extract the start and end points of each stroke as well as the specific direction of each stroke (Fig. 11). We selected one corresponding handwritten image from the CASIA-HWDB dataset [15], preprocessed it via skeletonization and identified start points. We then compare the number of potential start points extracted from the handwritten picture with those in the video to determine whether the continuous-stroke style were evident in the handwritten image for that character. In the event that continuous-stroke style were observed, we would regenerate the corresponding stroke sequence. The restoring stroke order sequence algorithm was used to identify the specific stroke sequence used by the volunteer when writing this character (as shown in Fig. 12). At the disconnection of strokes, we again used second-order Bezier curves to simulate the jump trajectory of the volunteer's finger. Finally, we calculated the distance between the device and each node encountered along the trajectory of the user's finger, the corresponding results of which are shown in Fig. 13.

5 SYSTEM DESIGN

Fig. 15 illustrates the system architecture comprising a server side and a mobile device side. The main task on the server side is to convert still images in MNISTs into tracking traces. The resulting *Track*MNIST dataset is then used to train a general-purpose Random Forest model for the letter classification on the warm-up stage. On the mobile device side, Strata is used to track the input patterns traced by the user's finger using the built-in microphones and speakers. The input system can be divided into three components based on their functions and are described in the following sub-sections.

5.1 Letter Trace Segmentation

In the current system, we impose the restriction on users that they need to pause briefly after each letter. This pause can be reflected in the 1D tracking system, wherein the slope of the relative change in distance is 0. As shown in Fig. 14, if the slope changes from non-zero to zero, then the zero boundary point is identified as the ending timestamp of the previous letter. If the slope changes from zero to non-zero, then the zero boundary point is identified as the start timestamp of the next word. After identifying the start and end timestamps of a letter, the trajectory between the two timestamps is used as the trace related to that letter.

5.2 Letter Classification Mechanism

This part addresses the problem of similarities among trajectories and information loss resulting from the use of a 1D (as opposed to 2D) finger tracking system. In the *MagicInput*, we developed a novel letter classification mechanism based on a personalized dataset.

5.2.1 Warm-up. In this stage, the *MagicInput* provided users with the output Top-K results and let them select the correct one. The "Word Correction" module can also check and correct the misclassified letters. Therefore, the *MagicInput* can satisfy the requirement of the multi-language text input recognition in the warm-up stage.

Meanwhile, the mobile device also uploads the trace for the letter fragment of interest and the corresponding label that user selected to the server. Personalized dataset will be used to improve the user experience and obtain the high performance on Top-1 classification.

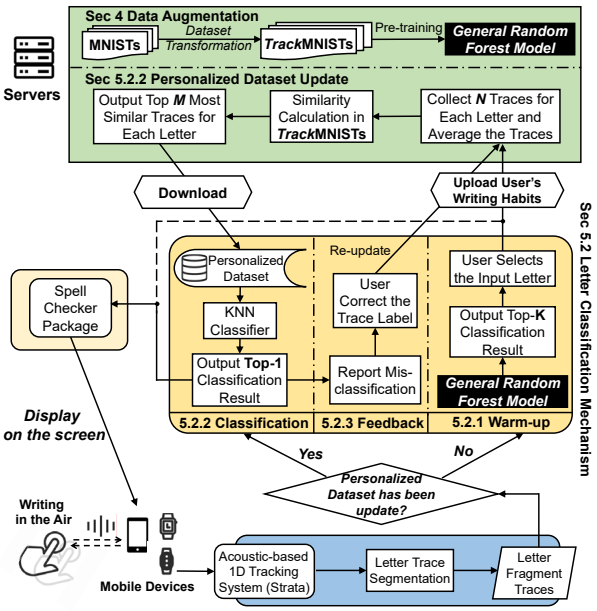


Figure 15: System architecture of the *MagicInput*.

5.2.2 Letter classification. Confronting the lack of labeled training data applicable to unseen users, we sought to develop the means by which to match training datasets to the user's writing characteristics and recognize letters using a simple machine learning scheme.

Personalized dataset update: The mobile device repeated N times of upload process for each letter. The server then finds in the *Track*MNISTs the top M traces that most closely resemble the average N uploaded letter traces, and sends them back to the mobile terminal. Until all the letters (*e.g.*, the whole 26 English lowercase letters) are updated, the mobile terminal obtain the complete dataset which included the testing user's writing habits.

K-nearest neighbors: With the updated personalized dataset, the KNN method is used to classify the traces of letter fragments created by the user. Note that low computational complexity of the KNN method makes it ideal for mobile devices. Note also that tailoring the dataset to the characteristics of the user allows KNN to achieve Top-1 classification results.

5.2.3 Failure Feedback. In the event that a user's writing habits change over time, the local dataset (*i.e.*, on the mobile device) must be changed accordingly. In the event of a classification error, the user only needs to make a note of the erroneous classification result and input the correct label. The mobile device then re-uploads the erroneous trace of letter fragment with the correct label to the server, which then examines the top M traces that most closely match the trace and sends them back to the mobile device. This makes it possible for the mobile to update its personalized dataset to bring it in line with the current writing habits of the user.

5.3 Word Correction

To deal with misclassifications due to similarities among letters (*e.g.*, 'a' and 'd') and imperfection of the classifier, we included a word correction mechanism (*e.g.*, Spell Check Package in Android) aimed at recognizing words rather than individual letters. Finally, the corrected words were displayed on the mobile device's screen.

6 EVALUATION

6.1 Experiment Setup

Hardware platform: We implement the proposed *MagicInput* as a prototype application on the Android phone. In the following experiments, the *MagicInput* app was installed on a Huawei Honor 9 running Android 8.0. The *MagicInput* cloud server was installed in a PC with an Intel Xeon Gold 5118 CPU @ 2.30GHz 12 Core and 48GB of RAM. Note that the server is only needed in the warm-up stage.

Multiple language MNIST datasets: We utilized the large-scale handwritten image datasets on the Internet and the data augmentation scheme (see Sec. 4) to create multiple-language artificial dataset as *Track*MNISTs. We used EMNIST dataset [5] to create artificial datasets for 10-class digits and 26-class uppercase/lowercase English letters, and used Kuzishiji-49 [4] dataset for 49-class Hiragana characters. For Chinese characters, we selected 30 commonly used and relatively complex Chinese characters from the Chinese MNIST dataset – CASIA-HWDB [16], and created corresponding artificial dataset for these 30-class Chinese characters.

Letter trace testing data collection: We recruited 15 volunteers as target users. During data collection, the users wrote letters with their fingers in the air along the XOY plane using the 3D coordinate system established by the mobile device. We made it clear to the volunteers should strive to keep their finger stationary for a period of time after each letter was written. To facilitate the labeling of collected data, we had the users input specific text content (including 10 digits, 26 uppercase and lowercase English letters as well as 49 Japanese and 30 Chinese characters) for 10 times, during which the mobile device recorded changes in the 1D position of the user's finger as it was moved. We then collected these traces into the testing dataset for each user.

6.2 Micro-benchmark

We performed micro-benchmark evaluation to select the key parameters in the system design, and conducted the following experiments in a user independent manner considering that these system parameters do not change with users.

6.2.1 1D Tracking Error Distribution. During data augmentation (Sec.4), we need to add it is necessary to consider the distribution of errors in the Strata (1D tracking) data while

Dataset	MNIST	EMNIST	EMNIST	Kuzushiji -49	CASIA -HWDB
Language Type	Digit	Uppercase English	Lowercase English	Japanese	Chinese
Rate(%)	96.3	91.1	86.4	64.5	71.8

Table 2: The success rate of the stroke order restoration algorithm on different MNISTs.

(a) Lowercase English letter 'f'. (b) Chinese character 'lun'.

Figure 16: Failure cases in restoring the stroke sequence.

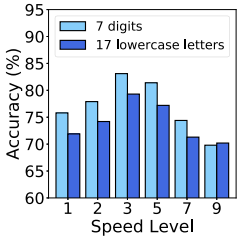


Figure 17: Parameter selection for finger moving speed levels of writing strokes.

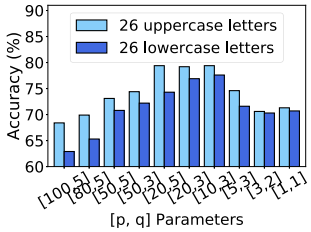


Figure 18: Parameter selection for simulating the traces left by a finger jumping movement.

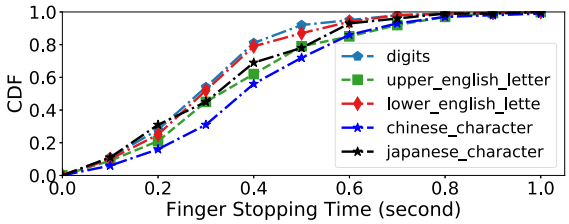


Figure 19: CDF of user's finger stopping time for letter segmentation.

estimating the relative changes in distance in real-world environments. Thus, we employed the testbed in Fig. 4, in which the volunteers were tasked with writing each lowercase English letter 50 times with his/her finger in accordance with the standard image of the letter using the standard stroke sequence. Throughout this exercise, Strata calculated the relative changes in distance between the user's finger and the mobile device. We also calculated the true changes in distance between the mobile device and the standard letter stroke sequence in the image as the ground truth. From this, we derived that the error distribution is a normal distribution (with mean $0.23mm$ and variance $0.16mm$).

6.2.2 Data Augmentation. **Performance of Restoring Stroke Sequence Algorithm.** We also evaluated the effectiveness of the proposed stroke sequence restoration algorithm (Sec. 4.1.1). Note that the algorithm is based on the concept of particle swarms; therefore, if $Num(T)$ (T is the *Simulated stroke sequence set*) is greater than or equal to 1, then we know that the particle swarm has traversed all of the standard strokes outlined in the instructional video. Thus, the success rate in assigning stroke sequences to images in the MNISTs was based on whether $Num(T) \geq 1$.

The stroke sequence restoration algorithm was applied to five MNIST datasets, the conversion success rates of which are listed in Tab. 2. The success rates when applied to MNIST ($0 \sim 9$ digits) and EMNIST (26 uppercase English letter) were very high, reaching 96.3% and 91.1%, respectively. The success rate when applied to EMNIST (26 lowercase English letter) was relatively low (86.4%). We later determined that this poor performance can be attributed to volunteers scribbling many of the letters (as shown in Fig. 16(a)). The success rates when applied to CASIA-HWDB and Kuzushiji-49 were low, reaching only 71.8% and 64.5%, respectively. Fig. 16(b) presents the failure cases that occurred when restoring stroke sequence to the Chinese character 'lun' (see Fig. 11).

Parameter Selection in Writing Speed Levels. In the Sec. 4.2, we discussed the influence of writing size, writing speed and writing position on 1D tracking patterns. Thanks to normalization, we can disregard the impact of writing size. As for writing position, we discussed two types of positions (XOY and $(-X)OY$) in our system. In this part, we examine the impact of the parameters of the simulated writing stroke speed, and test it on the digits or lowercase English letters that can be written in one continuous stroke. Linear down- and up-sampling was used to simulate a single speed change (*e.g.*, $\times 1$), three speed changes (*e.g.*, $\times 0.5$, $\times 1$, $\times 1.5$), five speed changes (*e.g.*, $\times 0.3$, $\times 0.6$, $\times 1$, $\times 1.3$, $\times 1.6$), 7 speed changes and 9 speed changes respectively, and generate corresponding artificial datasets and pre-trained RF models. The RF model trained using data that included three speed simulations achieved the highest classification accuracy (72.8%) when applied to data from real-world users. By comparison the RF model without speed simulations was 64.9% and the RF model trained using speed simulations was 70.6%.

Parameter Selection for Simulating Traces of Finger Jumps. In the following, we obtain the pattern presented by the Bezier curve by selecting the coordinates of control point B (see in Fig. 10). We take p points distributed uniformly in the area ($x_2 \in [x_1, x_3], y_2 \in [0, x_3 - x_1]$) to generate p types of corresponding Bezier curves. Similarly, we simulate the user behavior by controlling the angle θ between the plane in which finger jumping occurs and the one in which writing is performed (see in left of Fig. 9). We take q points distributed uniformly in area $\theta \in [0, \pi]$. A combination of p and q allows

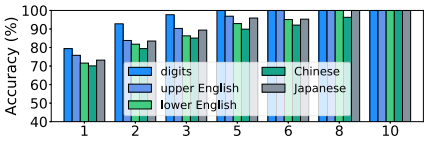


Figure 20: Determine parameter K.

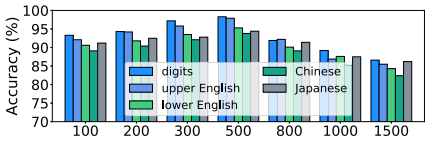


Figure 21: Determine parameter M.

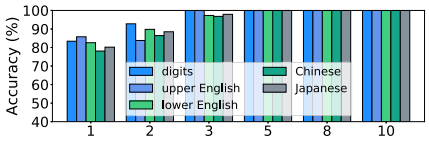


Figure 22: Determine parameter N.

the comprehensive simulation of the jumping movement made by the user's finger when it encounters a disconnected stroke. Fig. 18 illustrates the impact of the RF model on classification accuracy after applying to the dataset simulated trajectories based on various combinations of $[p, q]$. Based on the results, we selected $[10, 3]$ with which to generate the final dataset and corresponding RF model.

6.2.3 Letter Segmentation. The letters were segmented by finding trajectories with a slope of 0. Note that if the users were expected to pause for too long, it would greatly undermine the user experience. By contrast, if the pauses were not long enough, the letter segmentation algorithm would be unable to recognize the letter trajectory associated with the pause. During the collection of data, a letter segmentation algorithm was used to segment each trace and then assess the results according to the length of the resulting segments. The CDF of the user's finger stopping time for letter segmentation is shown in Fig. 19. We found that when the user pauses for $0.8s$, our system can perform almost 100% of the effective letter segmentation on all five language letters.

6.2.4 Letter Classification Mechanism. **Performance of general RF model:** The first stage in the classification process is the warm-up, the performance of which relies on the general RF model trained in advance by the server. The fact that the *Track*MNIST dataset does not account for the writing characteristics of users prevents the general-purpose RF model from performing letter classification with a high degree of accuracy for every user. We therefore adopted the Top-K classification results from which users may select the letter they meant to draw. We trained a general purpose RF model for *Track*MNIST corresponding to each language, and then used letter trace data collected from users to evaluate its classification performance, the results of which are shown in Fig. 20. Overall, we found that the pre-trained RF model for each language achieved an average of 90% accuracy for Top-5 classification on the multi-language letters, and near 100% accuracy of Top-10 classification. Since the users do not require to provide their training data or be involved in the pre-training, our system can support multiple languages without users' effort in the warm-up stage.

Personalized dataset collection: Parameter M indicates the number of traces the system must collect from a given letter trace with the target user's writing habits in order to find to achieve Top-1 classification accuracy using the KNN classifier. Parameter N indicates the minimum number of samples the server must average in order to accurately characterize the writing habits of the target user.

In determining parameter M, we set N at a large enough value (*e.g.*, 20) to ensure that the average trace data is indicative of the user's writing characteristics. From each user, we therefore select 20 trace samples for a given letter and then average them, the result of which is used to represent the user's writing characteristics. We select from *MagicInput* the Top-M traces with the highest degree of similarity for inclusion in the user's personalized dataset, and then use the KNN classifier to evaluate the representativeness of the personalized dataset. As shown in Fig. 21, if M is set too small (*e.g.*, 100), the selected dataset may be insufficiently robust, whereas if M is set too high (*e.g.*, 1500), then the enormous number of irrelevant items would interfere with KNN classification performance. Thus, we set M between 300 and 500 in seeking to obtain the Top-1 classification results.

After setting M to 300, we select N traces for a given letter and use the averaged trace results as the final trace by which to represent the user's writing characteristics. We then collect the Top-300 traces presenting the highest degree of similarity from the *Track*MNIST for inclusion in the personalized dataset. Finally, we use KNN classification accuracy (Top-1) to determine whether N traces provide a representative indication of the target users' writing habits. As shown in Fig. 22, setting N at 3 was sufficient to represent the style of the target user in writing that particular letter.

6.2.5 Effectiveness of Personalized Dataset. We evaluated the efficacy of the personalized dataset sought from the artificial *Track*MNIST, compared with the true collected training dataset from user himself. We did the following experience. We randomly divided the trace dataset collected from three users into a training dataset and a testing dataset (at a ratio of 7:3). This training dataset was used directly as the user's personalized dataset II. The Top-2500 traces with the highest degree of similarity from *Track*MNIST were used for as the personalized dataset Type I. We applied the KNN classifier to the same testing dataset for each user in order to compare the representativeness of the two types of personalized dataset. As shown in Fig. 23, personalized dataset

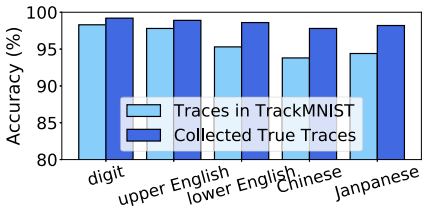


Figure 23: Personalized Dataset I from *Track*MNIST vs. Personalized Dataset II collected by the user.

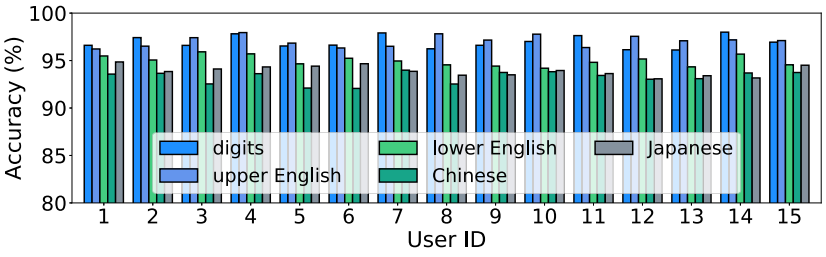
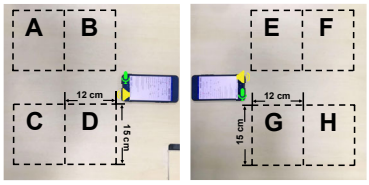
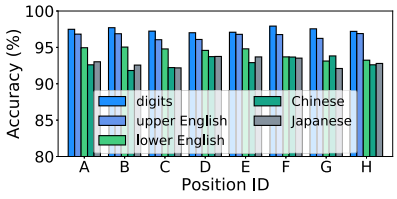


Figure 24: Accuracy of *MagicInput* in classifying characters in multiple languages when meeting unseen users.



(a) Testing Posistions.



(b) Classfication Performation.

Figure 25: Influence of positions between the mobile device and user's finger on performance of *MagicInput*.

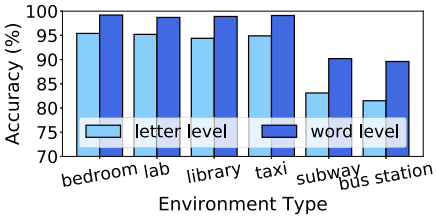


Figure 26: Influence of surrounding environment.

II achieved good classification performance on all five testing datasets with average accuracy of 98.5%. Personalized dataset I also achieved good classification performance on the testing datasets with an average accuracy of 95.9%. It is sufficient proof that the dataset selected from the *Track*MNIST is sufficient to represent the writing habits of the target user.

6.3 MagicInput System Overall Evaluation

6.3.1 Multi-user and Multi-language Letter/Character Input Performance. We evaluated the overall performance of the *MagicInput* for online multilingual letter recognition from unknown users. Fifteen users employed the *MagicInput* app on their smartphones without collecting any training data specific to the target user. We then tested the performance of the input system when applied to five languages, the results of which are shown in Fig. 24. We achieved the following average classification accuracy: 10 digits (95.3%), 26 uppercase English letters (97.8%), 26 lowercase English letters (95.3%), 30 most commonly used Chinese characters (93.8%), and 49 Japanese characters (91.4%).

6.3.2 Impact of Input Positions. The *MagicInput* supports user finger input in multiple positions relative to the mobile device. We evaluated the impact of relative position on the accuracy of the system by having a volunteer write letters while holding the phone in eight different relative positions (see Fig. 25(a)). As shown in Fig. 25(b), the position of the mobile device relative to the finger was shown not to have a significant effect on classification accuracy. This shows that our system has good compatibility with users with different dominant hands, and has no mandatory requirement for input position.

6.3.3 Impact of Environmental Factors. Acoustic-based finger input systems are easily affected by the surrounding environment, due to multi-path effects. After completing the warm up, we had a volunteer use *MagicInput* to input lowercase letters in four different environments, while striving to employ the same writing habits throughout. We then evaluated the impact of the environment on classification performance and word recognition accuracy. As shown in Fig. 26, environments with small human traffic (*e.g.*, labs and libraries) had little impact on classification performance at the letter and word level. Noise environments with with heavy human traffic (*e.g.*, subways and bus stations) were shown to have an obvious detrimental effect on performance. Frequent moving objects around the device may affect the finger 1D tracking accuracy of the Strata system to a certain extent, resulting in distortion of the letter trace pattern and a decrease in the classification accuracy.

6.3.4 Comparison with Other Text Input Systems. The performance of *MagicInput* was compared with that of touchscreens and other acoustic-based systems (on uppercase English recognition), including EchoWrite and Wordrecorder [7,

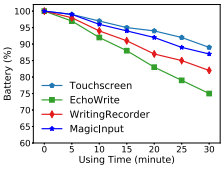


Figure 27: Comparison of power consumption.

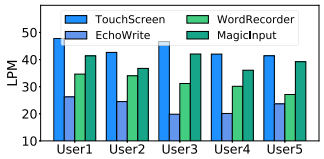


Figure 28: Comparison of LPM rate.

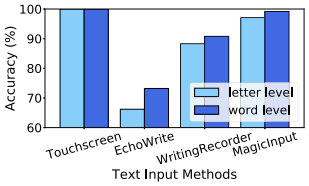


Figure 29: Comparison on letter/word recognition.

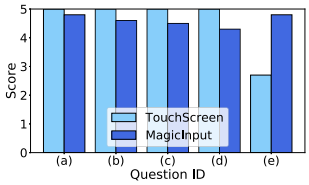


Figure 30: User satisfaction of *MagicInput*.

28]. Like the proposed system, Echowrite identifies letters written in the air, whereas Wordrecorder identify letters by analyzing the audible signal generated by pens moving across the surface of paper. For the sake of fairness, the same volunteer operated all four of the systems, which were installed on the same smartphone (Huawei Honor 9). Again, we did not provide a training dataset specific to the target users. All systems used the same built-in spell checker package to evaluate word recognition accuracy.

Letter and Word Recognition: Fig. 29 presents the recognition rates achieved by the four input methods when applied to uppercase English letters by an unknown user. The touchscreen achieved accuracy of 100%. Echowrite achieved accuracy of 66.2% when applied to uppercase letters, and 73.2% on the words. Wordrecorder achieved accuracy of 88.2% on uppercase letters, 90.1% on the words. *MagicInput* achieved accuracy of 97.3% on uppercase letters, 100% on the words.

Input Efficiency: We used the LPM (letter per minute) indicator to measure the input efficiency, and Fig. 28 illustrates the rate at which five different users were able to input uppercase English letters using the four systems. Touchscreen-based handwriting input method obtained the maximum LPM value. And *MagicInput* achieved the second highest input efficiency in the four methods, which is directly related to the performance of letter classification.

Power Consumption: Fig. 27 shows the battery power information of the Huawei Honor 9 after the same user continuously using these methods for 30 minutes respectively. Our *MagicInput* system does not significantly increase the power consumption of the device, thanks to the small calculations of the Strata 1D finger tracking system and the simple classification model – KNN classifier.

6.4 User Study

Another 15 volunteers are recruited for the user study (7 males and 8 females). We first gave a 15-minute instruction on how to use *MagicInput*. After that, volunteers were free to try *MagicInput* for 10 minutes. The input they made during the period was also used to improve their own models (*i.e.*, personalized dataset that matches with writing habits). Once the instruction is done, we asked volunteers to come and sit in a coffee shop, and use *MagicInput* and touchscreen-based handwriting input method in turns. We conducted an experiment to assess user satisfaction with our proposed *MagicInput*, covering (a) letter classification accuracy, (b) input speed, (c) power consumption, (d) multi-text support, and (e) convenience compared to the touchscreen-based method. Surveys were used Likert-type scoring, as follows: 5 (very satisfied), 4 (satisfied), 3 (neutral), 2 (unsatisfied), and 1 (very unsatisfied).

As shown in Fig. 30, nearly all of the volunteers were satisfied with the letter recognition performance, input speed, and power consumption of *MagicInput*. All of the volunteers expressed amazement that our system supports text input in multiple languages and expressed satisfaction with the accuracy in other languages. Most of the volunteers reported that *MagicInput* would be more convenient than a touchscreen when using small devices (*e.g.*, smartwatch), while driving (speech recognition software is affected by ambient sound), while playing games is full-screen mode.

7 CONCLUSION

This paper presents *MagicInput*, a training-free text input system that support multiple languages using acoustic-based 1D finger tracking technology. *MagicInput* took advantages of the large-scale handwritten image dataset (MNISTs) and designed a novel data augmentation scheme to create a same large-scale artificial dataset (*Track*MNIST). We compensate for the loss of 1D tracking information and similarities in the letter trajectory by creating a personal dataset (from *Track*MNIST) matching the writing habits of each user and improve letter classification using the KNN classifier. In experiments, *MagicInput* achieved outstanding classification accuracy on the letter traces written by unseen users in a variety of languages.

ACKNOWLEDGMENTS

We are grateful to our shepherd and anonymous reviewers for their constructive feedback. We appreciate all participants in our experiments and user study. This work is supported in part by the NSFC (U1736207, 62072306, 61936015), Program of Shanghai Academic Research Leader (20XD1402100), and Startup Fund for Youngman Research at SJTU.

REFERENCES

[1] Fadel Adib, Zach Kabelac, Dina Katabi, and Robert C Miller. 2014. 3d tracking via body radio reflections. In *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*. 317–329.

[2] Baidu. 2021. Baidu Hanyu. https://hanyu.baidu.com/zici/. (2021).

[3] Mingyu Chen, Ghassan AlRegib, and Biing-Hwang Juang. 2015. Air-writing recognition—Part I: Modeling and recognition of characters, words, and connecting motions. *IEEE Transactions on Human-Machine Systems* 46, 3 (2015), 403–413.

[4] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. 2018. Deep learning for classical japanese literature. *arXiv preprint arXiv:1812.01718* (2018).

[5] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. 2017. EMNIST: Extending MNIST to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2921–2926.

[6] Thomas Deselaers, Daniel Keysers, Jan Hosang, and Henry A Rowley. 2014. Gyropen: Gyroscopes for pen-input with mobile phones. *IEEE Transactions on Human-Machine Systems* 45, 2 (2014), 263–271.

[7] Haishi Du, Ping Li, Hao Zhou, Wei Gong, Gan Luo, and Panlong Yang. 2018. Wordrecorder: Accurate acoustic-based handwriting recognition using deep learning. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 1448–1456.

[8] Zhangjie Fu, Jiashuang Xu, Zhuangdi Zhu, Alex X Liu, and Xingming Sun. 2018. Writing in the air with WiFi signals for virtual reality devices. *IEEE Transactions on Mobile Computing* 18, 2 (2018), 473–484.

[9] Zhengxin Guo, Fu Xiao, Biyun Sheng, Huan Fei, and Shui Yu. 2020. WiReader: Adaptive Air Handwriting Recognition Based on Commercial Wi-Fi Signal. *IEEE Internet of Things Journal* (2020).

[10] Teach Handwriting. 2021. Print Letters Animations and Worksheets. https://www.teachhandwriting.co.uk/print-letters-beginners.html. (2021).

[11] Kiran Joshi, Dinesh Bharadia, Manikanta Kotaru, and Sachin Katti. 2015. WiDeo: Fine-grained Device-free Motion Tracing using {RF} Backscatter. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*. 189–204.

[12] Gierad Laput, Robert Xiao, Xiang'Anthony' Chen, Scott E Hudson, and Chris Harrison. 2014. Skin buttons: cheap, small, low-powered and clickable fixed-icon laser projectors. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. 389–394.

[13] Wenzhe Li and Tracy Hammond. 2011. Recognizing text through sound alone. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*. 1481–1486.

[14] Jaime Lien, Nicholas Gillian, M Emre Karagozler, Patrick Amihood, Carsten Schwesig, Erik Olson, Hakim Raja, and Ivan Poupyrev. 2016. Soli: Ubiquitous gesture sensing with millimeter wave radar. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–19.

[15] Cheng-Lin Liu, Fei Yin, Da-Han Wang, and Qiu-Feng Wang. 2011. CASIA online and offline Chinese handwriting databases. In *2011 International Conference on Document Analysis and Recognition*. IEEE, 37–41.

[16] Cheng-Lin Liu, Fei Yin, Da-Han Wang, and Qiu-Feng Wang. 2013. Online and offline handwritten Chinese character recognition: benchmarking on new databases. *Pattern Recognition* 46, 1 (2013), 155–162.

[17] Wenguang Mao, Jian He, and Lili Qiu. 2016. CAT: high-precision acoustic motion tracking. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. 69–81.

[18] Rajalakshmi Nandakumar, Vikram Iyer, Desney Tan, and Shyamnath Gollakota. 2016. Fingerio: Using active sonar for fine-grained finger tracking. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 1515–1525.

[19] 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.

[20] Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *Proceedings of the 14th international conference on frontiers in handwriting recognition*. IEEE, 285–290.

[21] Arik Poznanski and Lior Wolf. 2016. Cnn-n-gram for handwriting word recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2305–2314.

[22] Li Sun, Souvik Sen, Dimitrios Koutsonikolas, and Kyu-Han Kim. 2015. Widraw: Enabling hands-free drawing in the air on commodity wifi devices. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. 77–89.

[23] Wei Wang, Alex X Liu, and Ke Sun. 2016. Device-free gesture tracking using acoustic signals. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*. 82–94.

[24] Teng Wei and Xinyu Zhang. 2015. mtrack: High-precision passive tracking using millimeter wave radios. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*. 117–129.

[25] Martin Weigel, Tong Lu, Gilles Bailly, Antti Oulasvirta, Carmel Majidi, and Jürgen Steimle. 2015. Iskin: flexible, stretchable and visually customizable on-body touch sensors for mobile computing. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2991–3000.

[26] Martin Weigel, Aditya Shekhar Nittala, Alex Olwal, and Jürgen Steimle. 2017. Skinmarks: Enabling interactions on body landmarks using conformal skin electronics. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 3095–3105.

[27] Elliott Wen, Winston Seah, Bryan Ng, Xuefeng Liu, and Jiannong Cao. 2016. UbiTouch: ubiquitous smartphone touchpads using built-in proximity and ambient light sensors. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 286–297.

[28] Kaishun Wu, Qiang Yang, Baojie Yuan, Yongpan Zou, Rukhsana Ruby, and Mo Li. 2020. EchoWrite: An acoustic-based finger input system without training. *IEEE Transactions on Mobile Computing* (2020).

[29] Yi-Chao Wu, Fei Yin, and Cheng-Lin Liu. 2017. Improving handwritten Chinese text recognition using neural network language models and convolutional neural network shape models. *Pattern Recognition* 65 (2017), 251–264.

[30] Chao Xu, Parth H Pathak, and Prasant Mohapatra. 2015. Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. 9–14.

[31] Tuo Yu, Haiming Jin, and Klara Nahrstedt. 2016. Writinghacker: Audio based eavesdropping of handwriting via mobile devices. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 463–473.

[32] Sangki Yun, Yi-Chao Chen, Huihuang Zheng, Lili Qiu, and Wenguang Mao. 2017. Strata: Fine-grained acoustic-based device-free tracking. In *Proceedings of the 15th annual international conference on mobile systems, applications, and services*. 15–28.

[33] Maotian Zhang, Panlong Yang, Chang Tian, Lei Shi, Shaojie Tang, and Fu Xiao. 2015. Soundwrite: Text input on surfaces through mobile acoustic sensing. In *Proceedings of the 1st International Workshop on Experiences with the Design and Implementation of Smart Objects*. 13–17.