From 2D to 3D: Facilitating Single-Finger Mid-Air Typing on QWERTY Keyboards with Probabilistic Touch Modeling

XIN YI*, Institute for Network Sciences and Cyberspace, Tsinghua University; Zhongguancun Laboratory, China
CHEN LIANG†, Department of Computer Science and Technology, Tsinghua University, China
HAOZHAN CHEN†, Department of Computer Science and Technology, Tsinghua University, China
JIUXU SONG, University of California, Santa Barbara, United State
CHUN YU, Department of Computer Science and Technology, Tsinghua University, China
HEWU LI, Institute for Network Sciences and Cyberspace, Tsinghua University; Zhongguancun Laboratory, China
YUANCHUN SHI, Department of Computer Science and Technology, Tsinghua University, China

Mid-air text entry on virtual keyboards suffers from the lack of tactile feedback, which brings challenges to both tap detection and input prediction. In this paper, we explored the feasibility of single-finger typing on virtual QWERTY keyboards in mid-air. We first conducted a study to examine users' 3D typing behavior on different sizes of virtual keyboards. Results showed that the participants perceived the vertical projection of the lowest point on the keyboard during a tap as the target location and inferring taps based on the intersection between the finger and the keyboard was not applicable. Aiming at this challenge, we derived a novel input prediction algorithm that took the uncertainty in tap detection into a calculation as probability, and performed probabilistic decoding that could tolerate false detection. We analyzed the performance of the algorithm through a full-factorial simulation. Results showed that the SVM-based probabilistic touch detection together with a 2D elastic probabilistic decoding algorithm (elasticity = 2) could achieve the optimal top-5 accuracy of 94.2%. In the evaluation user study, the participants reached a single-finger typing speed of 26.1 WPM with 3.2% uncorrected word-level error rate, which was significantly better than both tap-based and gesture-based baseline techniques. Also, the proposed technique received the highest preference score from the users, proving its usability in real text entry tasks.

CCS Concepts: • **Human-centered computing** → **Text input**; **Virtual reality**.

Additional Key Words and Phrases: text entry, virtual reality

*indicates the corresponding author.
†indicates equal contribution.

Authors' addresses: Xin Yi, hhdxs2@163.com, Institute for Network Sciences and Cyberspace, Tsinghua University; Zhongguancun Laboratory, China; Chen Liang, liang-c19@mails.tsinghua.edu.cn, Department of Computer Science and Technology, Tsinghua University, China; Haozhan Chen, chz21@mails.tsinghua.edu.cn, Department of Computer Science and Technology, Tsinghua University, China; Jiuxu Song, jiuxu@ucsb.edu, University of California, Santa Barbara, United State; Chun Yu, chunyu@tsinghua.edu.cn, Department of Computer Science and Technology, Tsinghua University, China; Hewu Li, lihewu@cernet.edu.cn, Institute for Network Sciences and Cyberspace, Tsinghua University; Zhongguancun Laboratory, China; Yuanchun Shi, shiyc@tsinghua.edu.cn, Department of Computer Science and Technology, Tsinghua University, China.

2474-9567/2023/3-ART38 $15.00
https://doi.org/10.1145/3580829

ACM Reference Format:
Xin Yi, Chen Liang, Haozhan Chen, Jiuxu Song, Chun Yu, Hewu Li, and Yuanchun Shi. 2023. From 2D to 3D: Facilitating Single-Finger Mid-Air Typing on QWERTY Keyboards with Probabilistic Touch Modeling. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 1, Article 38 (March 2023), 25 pages. https://doi.org/10.1145/3580829

1 INTRODUCTION

Text entry on head-mounted displays (HMDs) is key to supporting general access to various VR/AR applications. Usually, text entry on HMDs is achieved by using handheld devices (e.g., a controller) or bare-hand mid-air typing. For example, Oculus Quest 2 uses a ray cast from the hand controller to point at keys on a virtual keyboard and the user confirms the selection by pressing a button. Microsoft Hololens 2 allows the users to directly 'touch' on the virtual keyboard to enter text. However, without the incorporation of modern input prediction algorithms, both systems are prone to input errors. Users need to input carefully, which significantly limits the input performance and naturalness.

Input prediction algorithm plays an important role in modern text entry techniques. The most widely-adopted input prediction algorithm was the statistical decoding algorithm first proposed by Goodman et al. [14]. By using probabilistic distribution to model the imprecision in touch behavior, this algorithm was proved effective in numerous scenarios [7, 42, 48]. There are also works that tried to incorporate input prediction algorithms into mid-air typing to facilitate the input performance (e.g., touch typing [48] and gesture typing [25]). However, the requirement of considerable learning effort or typing experience limited the applicability of these techniques in real use.

In this paper, we focused on the most simple and intuitive text entry modality - single-finger typing on a virtual QWERTY keyboard. This solution required minimal prior knowledge from the user and introduced little learning effort. To our knowledge, only Dudley et al. [8] tested single-finger typing with a rule-based detection method, achieving an input speed of 17.75 WPM. As we will show in this paper, with appropriate input prediction algorithms, single-finger typing could even achieve competitive performance with multi-finger typing [7, 48]. Therefore, we believe this technique has great potential for efficient and natural text input on commercial HMD products.

Compared with typing on touchscreens, the absence of tactile feedback in mid-air typing introduced three major challenges: 1) the tapping process was highly mixed with movements between different keys, making it difficult to detect the number of taps; 2) it was difficult to infer the users' intended tap location within a tap given the continuous finger trajectory and the ambiguity in the aiming strategy; 3) the input imprecision in 3D space was greater than on 2D planes, not only resulting in a wider spread of touch points on the projected plane but also introducing the offset in the depth axis (details in Study 1).

Aiming at these challenges, we first conducted a user study to investigate the users' single-finger mid-air typing behavior. Results showed that about 1/5 of all taps did not lead to the intersection between the finger and the keyboard. Regarding the users' aiming strategy, a lowest-vertical-tip aiming model was shown optimal to fit the users' mental model by yielding the smallest standard deviations of the projected touch points on the virtual keyboard. Based on such findings, we trained a support vector machine (SVM) model with RBF kernel to detect the touch points from the continuous input data, achieving an F-1 score of 0.977.

To compensate for the imprecision in touch detection and touch point location, we then proposed a novel input prediction algorithm that incorporated the uncertainty in tap detection into an elastic probabilistic decoding algorithm and treated false detection as insertion and omission errors. We verified the performance of this model through a full-factorial simulation (geometry model of touch point, tap probability, language model, elasticity and tap detection model). Results showed that the SVM-based probabilistic touch detection together with a 2D elastic probabilistic decoding algorithm (elasticity = 2) could achieve the optimal top-5 accuracy of 94.2% on a 10,000-word dictionary.

In the second user study, we evaluated the performance of the proposed technique in a text transcription task with three baselines: Vulture[25], a single-finger gesture typing technique in VR, and two techniques with collision-based tap detection and rigid input prediction algorithms, respectively. The users typed 26.1 WPM using our technique, which was significantly faster than all the baselines. Expert users could even achieve 36.8 WPM. Meanwhile, the uncorrected word-level error rate (3.2%) was competitive with the other techniques. Compared with the two tap-based baseline techniques, our proposed technique led to fewer strokes for correcting errors and more confidence during typing, therefore was the most preferred by the users. We believe this paper provided an exciting demonstration of single-finger mid-air typing's potential in virtual reality.

2 RELATED WORK

2.1 Hand-Based Text Input Techniques on HMDs

Text entry on head-mounted devices (HMDs) is a fundamental input capability that has been widely researched over the past years. Leveraging the multi-modality sensing ability of HMDs (e.g., inertial-based sensing [49] and vision-based sensing) and hand controllers (e.g., 6 DoF tracking of the gripping hand), researchers have investigated techniques with different input modalities (e.g., head rotation [49], pointing with hand controller [37, 44], and bare-hand mid-air typing [8, 25]) to facilitate the text entry experience on HMDs. Among these techniques, hand-based input gained popularity for the experience of physical keyboards and the flexibility of the human hand to support natural and efficient HMD text input [37]. To resolve the absence of tactile feedback in mid-air typing, researchers tended to use auxiliary input devices (e.g., physical keyboards [28], hand controllers [44], or special hardware [18, 32]) to offer supplementary input channels (e.g., a physical key or a pointing device). However, this also impeded real-world deployment and exerted extra learning costs for a specific technique.

Focusing on mid-air text entry with the users' hands, researchers have proposed 3D handwriting techniques using inertial sensors and cameras (e.g., Airwriting [3] and Airstroke [27]). Although with the advantages of intuitiveness and eyes-free input, these techniques suffered from low input speed (e.g., 11 WPM [27]) and recognition accuracy due to input imprecision (e.g., 89% word-level accuracy [3]).

Typing on a virtual keyboard may be the most efficient input method, as users could transfer the muscle memory of typing on physical and touchscreen keyboards. Researchers have proposed both tap-based [8, 48] and gesture-based [25] typing techniques. Vulture [25] enabled gesture typing on a virtual keyboard, reaching an input speed of 20.6 in the first 10 sessions and 28.1 WPM after active practice, which was slower than the gesture typing speed on touchscreens [31] and required considerable training. ATK [48] enabled ten-finger typing in mid-air with a speed of 29.2 WPM though it also required additional practice and expected the user to input with the standard fingering on the physical keyboard. Dudley et al. [7] investigated two modalities (two-finger v.s. ten-finger) for mid-air text entry, where the two-finger method achieved an input speed of 42.1 WPM.

As most related to our work, VISAR [8] explored single-finger tap-based typing on commodity AR HMDs (HoloLens 1) with an iterative interaction design, achieving an input speed of 17.8 WPM. Although such a performance indicated restricted input efficiency probably due to the coarse hand tracking probability, VISAR's effort in finding an optimized solution balancing detection, interaction, and feedback was inspiring. In comparison, our work was going in another direction that allowed the user to perform taps in a natural manner based on their understanding and proposed algorithms to decode their noisy input behavior. To achieve this, we delved deeper into the user's mid-air typing behavior and developed algorithms to detect taps and predict the input word, leading to a significant improvement in typing performance and learnability.

2.2 Probabilistic Modeling for Text Entry

Most modern smart keyboards incorporated the statistical decoding algorithm first proposed by Goodman et al. [14] to support efficient word-level or sentence-level [38–40] text input. Given a series of touch points

$I = I_1 I_2 \cdots I_n$, the algorithm predicted the word W with the maximum $P(W|I)$ value. According to Bayesian rule, this was equivalent to maximizing $P(I|W) \times P(W)$, where $P(W)$ was the language model (e.g., unigram or bigram [45]) and $P(I|W)$ modeled the spatial distribution of touch points. In basic conditions where no insertion or deletion error happens and consecutive touch points are independent, the computation can be:

$$P(I|W) = P(I_1 I_2 \cdots I_n | W_1 W_2 \cdots W_n) = \prod_{i=1}^{n} P(I_i|W_i) \tag{1}$$

where $P(I_i|W_i)$ is usually calculated using bivariate Gaussian distributions (e.g. [4]).

Although effective in the basic condition, this algorithm has also been improved by a number of researchers to fit a larger variety of scenarios. For example, researchers have tried to incorporate auxiliary information that was implicit during typing to facilitate the input performance (e.g., pressure [46], accelerometer [12], gripping posture [13] and finger identity [48]). Foy et al. [11] and Yi et al. [48] investigated the co-activation and co-movement effects for ten-finger mid-air typing that contributed to improving the decoding accuracy. Weir et al. [43] incorporated touch models into decoding for both implicit touch uncertainty and explicit control of touch pressure as indicators of the uncertainty. To handle insertion, omission, and substitution errors of users with Parkinson's disease, Wang et al. [42] proposed an elastic probabilistic model that took the probabilities of these errors into the calculation. And for eyes-free typing scenarios, researchers have proposed a Markov-Bayesian algorithm [21, 30, 35] that considered the relative position between consecutive touches:

$$P(I|W) = P(I_1|W_1) \times \prod_{i=2}^{n} P(I_i|W_i, W_{i-1}, I_{i-1}) \tag{2}$$

To our knowledge, although there exist previous works exploring the effectiveness of statistical decoding in mid-air typing in HMDs[2, 7], they were not optimized for single-finger mid-air typing. To fill this gap, we systematically compared different possible designs of a probabilistic decoding algorithm in this scenario, which led to the incorporation of probabilistic touch detection and elastic probabilistic decoding. Our algorithm, simulation, and evaluation results could serve as a complement to existing works.

2.3 Examining the Typing Behavior on Virtual Keyboards

To build touch models that better represent users' typing behavior on virtual QWERTY keyboards, researchers have conducted studies to investigate how users type on keyboards in different scenarios [25, 34, 47, 48]. Typing patterns [34, 47] and implicit behavioral features [46, 48] were quantitatively characterized by statistical modeling. For example, Yi et al. [47] investigated the touch point distribution on ultra-small input surfaces to validate the feasibility of efficient typing on wearable devices like smartwatches. Findlater et al. [10] examined the typing patterns, contact points, and hand contours of ten-finger typing for expert typists and found their individually spatially consistent keypress distributions. Other work [4, 5, 13, 19] also explored the characteristics of touch point distribution when using different hand postures to type on touchscreens. Despite the analysis of direct touch points, implicit behavioral features were also well-studied by previous work. For example, PalmBoard [46] found a significant effect of touch pressure in accessing different rows of keys. ATK [48] examined fingertip kinematics during tapping and showed fingers' co-movement when performing a touch in ten-finger mid-air typing. Such insights could contribute to incorporating significant factors in designing domain-specific decoding algorithms.

In our work, we focused on examining the single-finger mid-air typing behavior. As we will show in this paper, this was not only different from the single-finger typing pattern on touchscreens [4], but also from multi-finger typing behavior in mid-air [48]. Our results would be beneficial for designing user interaction interfaces and techniques in mid-air.

3 STUDY 1: EXAMINING SINGLE-FINGER MID-AIR TYPING BEHAVIOR ON QWERTY KEYBOARDS

In this section, we examined the users' single-finger mid-air typing behavior, as it yielded distinct patterns from that on touchscreens. We were especially interested in the finger kinematics during tapping, the user's aiming model, and the 3D distribution of touch points.

3.1 Participants and Apparatus

We recruited 12 participants (9 male, 3 female) from the campus with an average age of 22.5 (SD = 1.32). All participants were right-handed and used QWERTY keyboards for daily text entry. Each participant was compensated $15. We use an Oculus Quest 2 as the apparatus, which provided a 1832×1920 virtual reality scene per eye with a FOV of $89° \times 93°$. The hand tracking system on Quest 2 captured the user's hand images with four monochrome VGA fisheye cameras, predicted 21 key points for each hand at each frame, and fit the key points to a 3D articulated hand model [15]. As the result, the system API reported the articulated models of two hands, where we obtained the predefined markers of fingertips and finger pads [1]. The average finger joint angle error was approximately 9.6° [1].

3.2 Experiment Design

The goal of this study was to examine the user's single-finger typing behavior on 2D virtual keyboards in mid-air. Existing works [4, 9, 47] have found that the form factor (e.g., location, angle, size) of the keyboard could affect a user's typing behavior. However, traversing all these factors would yield an unacceptable experiment load and be beyond the scope of this paper. Therefore, we fixed the location and angle of the keyboard and explored how keyboard size impact users' typing comfortness through a pilot study: 8 users manually adjusted the location, angle and size of the keyboard to be the most comfortable value. As for location and angle of the keyboard, we took the average as the final setting. Therefore, the virtual keyboard was set to be 40cm away and 25.2cm lower than the user's head, and would not move with head rotation. This could help relieve motion sickness and tapping offset [37]. Furthermore, the keyboard angle was set perpendicular to the line between the helmet and the keyboard centroid. As for the size of the keyboard, the average width and height of the keyboard were 16.3cm (SD = 2.1) and 7.0 (SD = 0.9cm). We interviewed several participants and they expressed different preferences to different sizes of the keyboard. So we decided to test three levels of keyboard sizes in this study: *small* ($12.1cm \times 5.2cm$), *medium* ($16.3cm \times 7.0cm$), and *large* ($20.5cm \times 8.8cm$) to analyze which kind of keyboard was most acceptable by users.. In comparison, the size of a common physical keyboard was $19.0cm \times 5.4cm$. Similar to existing works [4, 10, 35], we did not employ touch detection algorithms or visual feedback in this study to ensure that we observed the most intrinsic typing behavior without bias towards any specific algorithms.

Another pilot study was done to determine the interactions of our mid-air keyboard. We added several functional buttons (space, delete and backspace) to the right of the keyboard and asked 8 users to pretend typing on the keyboard. Most of the users complained that it was exhausting to reach the space button every time they finish a word. So we decided to use gestures to substitute these functional buttons. Due to the limited hand tracking ability of Oculus Quest 2, the occlusion between fingers can lead to misrecognition, it was impratical to use only one hand to do both typing and gestures. So we use left-hand gestures instead.

3.3 Procedure

Each participant was first given three minutes to familiarize themselves with the experiment platform and then required to complete 3 sessions of text entry tasks, corresponding to the three keyboard sizes. The order of the sessions was counter-balanced among users. Each session contained 3 blocks and each block contained 10 phrases

[1] https://developer.oculus.com/documentation/unity/unity-handtracking

randomly sampled from the MacKenzie and Soukoreff phrase set [23]. A 2-minute break was enforced between blocks. Participants were instructed to type with the index finger of the dominant hand "as fast and as naturally as possible, as if on touchscreen keyboards". To emphasize, the participants were acknowledged that: 1) they were expected to perform natural touches with no requirements on the intersection between their fingers and the keyboard plane. 2) no feedback indicating the correctness of each touch was provided during the input of a word so that the user could type in the most natural manner based on their understanding. We mentioned the typing experience on a touchscreen keyboard where the user approximately taps on the target location instead of reaching the precise key location during typing as a reference to the definition of naturalness. 3) If the user subjectively considered they had input the correct word, they could confirm by pinching the index finger and the thumb of the left hand and the correct word would appear in the input area. 4) Once they felt they had committed any errors (e.g., reaching a wrong character, omitting a character, or inputting a wrong word) while typing, they could pinch the middle finger and the thumb of the left hand and then retype the last entered word.

3.4 Results

Across all participants, we collected a total of 8,952, 9,413, and 8,951 touches on small, medium, and large keyboards, respectively. We manually labeled the touches and the corresponding characters from the raw typing data according to the log data and the recording video.

We established a 3D coordinate system with the origin on the centroid of the keyboard (key 'G'), X and Y axis were parallel to the keyboard plane and were positive in the direction of right and up respectively, Z axis was perpendicular to the keyboard plane and points to the user. For each character, we removed outliers that were more than three standard deviations from the collected centroid in X and Y axis (small: 177 (2.0%), medium: 187 (2.0%) and large: 190 (2.1%)).

3.4.1 Input Speed. We measured the text entry speed using WPM (Words Per Minute), which was calculated as [22]:

$$WPM = \frac{|T| - 1}{S} \times 60 \times \frac{1}{5} \tag{3}$$

where $|T|$ was the length of the transcribed string, and S was the elapsed time in seconds from the first to the last tap in the phrase. The average typing speed for increasing keyboard sizes was 23.2 WPM (SD = 4.5), 22.6 WPM (SD = 5.2) and 24.3 WPM (SD = 3.7) respectively. RM-ANOVA found no significant effect of keyboard size on typing speed ($F_{2,22} = 1.27, p = .30$). This was about 1/3 slower than the index-finger typing speed on smartphones [4], presumably due to the larger keyboard size and the lack of tactile feedback. However, this was close to the single-finger gesture typing [25] speed in mid-air, and even ten-finger typing [48]. This was similar to the finding of Dudley et al. [7], that index-finger typing could achieve competitive or even superior typing speed than ten-finger typing in mid-air. Moreover, since the above typing speed was without visual indication of correctness, users' typing behavior was based on their confidence and expectation of the decoder's capability. Therefore, such a speed could serve as an empirical reference with the speed of our real system in Study 2 to indicate whether the system's capability to decode noisy input was above or below users' expectations.

3.4.2 Finger Kinematics during Tapping. Inferring discrete taps from continuous tapping data was ambiguous, even when there is a virtual keyboard plane for the users to refer to [7, 48]. To facilitate tap detection, we analyzed the finger kinematics during typing. We defined the *start* and *end* point of a tap to be the moment when z-velocity reached zero, and the *duration* of a tap to be the time that elapsed between the start and the end points. Further, *tap amplitude* was defined as the z-distance between the highest and the lowest points within a tap.

Figure 1 showed the average velocity and amplitude during a tap on Z axis. The average duration of a tap was 372.3ms (SD = 82.7), with an average pressing and lifting time being 182.7ms (SD = 51.7) and 189.6ms (SD = 53.3) respectively, which could guide the selection of window size for tap event detection in our algorithms.

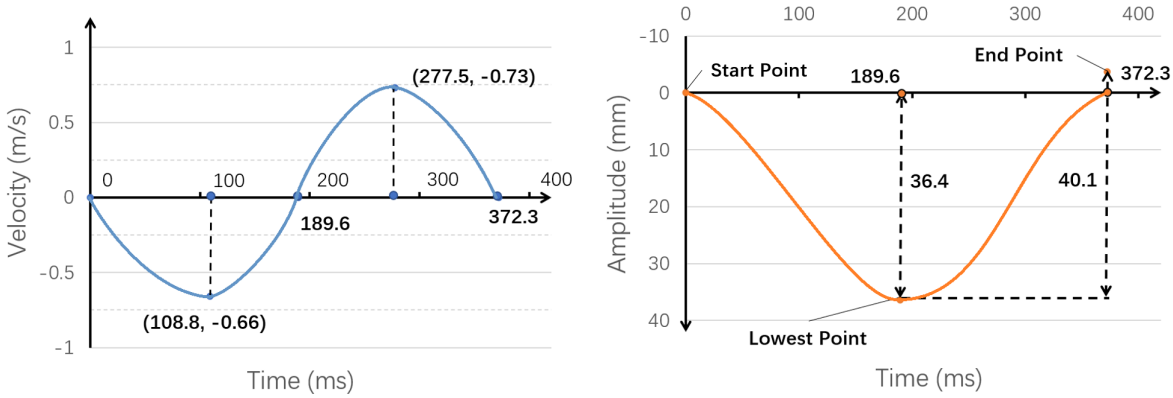


Fig. 1. Finger movement velocity and amplitude during a tap.

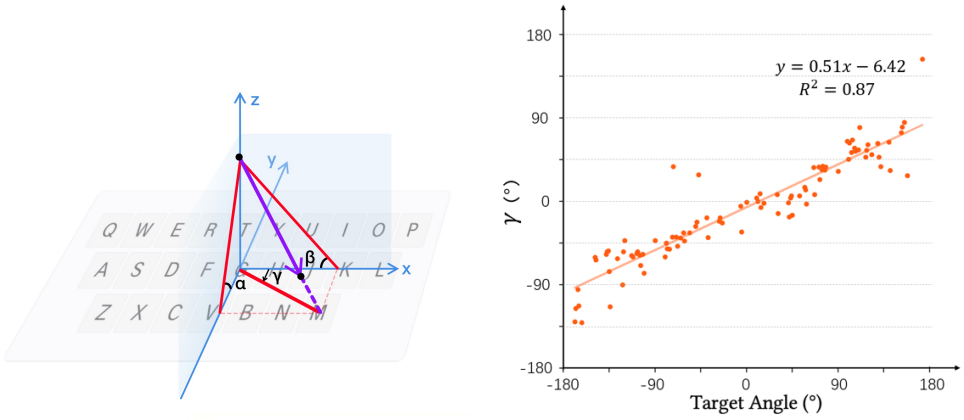
Different from multi-finger typing [48], the pressing phase was slightly shorter than the lifting phase. However, the maximum lifting speed was always greater than the maximum pressing speed (small: 0.60 V.S. 0.54m/s, medium: 0.73 V.S. 0.66m/s, large: 0.86 V.S. 0.77m/s). This speed was higher than the results of Dudley et. al. in two-finger typing [7], suggesting that the users tended to type faster when using fewer fingers, in order to maintain the input speed.

The average amplitude was 30.3mm (SD = 15.1), 35.4mm (SD = 15.8) and 42.3mm (SD = 18.1) for increasing keyboard sizes respectively, which was significantly different ($F_{2,22} = 1973, p < .001$). This denoted that the users tended to tap deeper on larger keyboards. Meanwhile, the mean lifting amplitude was always greater than the pressing amplitude (small: 33.7 V.S. 30.3mm, medium: 40.3 V.S. 36.4mm, large: 46.3 V.S. 42.3mm).

Interestingly, although we did not require the participants to tap through the keyboard plane, the average z coordinate of the lowest point during a tap was still negative (mean = -8.0mm, SD = 9.6). This suggested that some participants still tended to tap across the keyboard plane. However, the average distance between the keyboard plane and the highest point (28.3mm) was much greater than the lowest point (8.0mm), indicating that this ratio was not high. We confirmed this by finding that 83.2% (SD = 12.6%), 80.4% (SD = 16.1%) and 86.2% (SD = 11.3%) of all taps crossed the keyboard plane for increasing keyboard sizes, with no significant difference found among them ($F_{2,22} = 1.21, p = .32$). This result implied that detecting taps by simply observing the intersection led to significant amounts of error and was not applicable in one-finger mid-air typing scenario.

3.4.3 Tapping Direction. We also analyzed the directional characteristics of the taps, which provided more information on the movement of the fingers on the keyboard. Specifically, we calculated a *tap vector* between the start and the lowest point during the pressing phase (see Figure 1), and computed α, β and γ by calculating the angles on different projection planes, as illustrated in Figure 2.

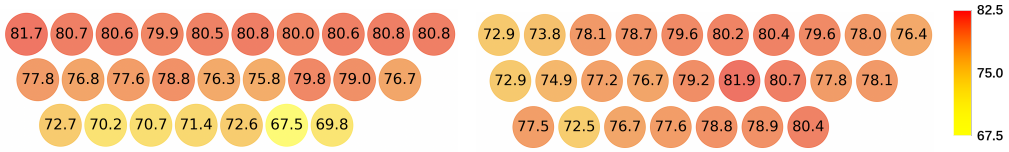
We calculated γ as the angle between the x axis and the tap vector's projection on the x-y plane ($-180° - 180°$). Meanwhile, we calculated the *target angle* as the direction from the previous key to the current key on the keyboard. Figure 2b showed the correlation between γ and target angle, which yielded an R^2 of 0.87. This suggested a generally strong linear correlation between these two values, which confirmed that the finger tapping procedure and movement between different keys were highly mixed. The slope and intercept of the model indicated that the users' taps tended to be squeezed to a smaller angle range ($\sim [-100°, 100°]$) and had a systematic bias, probably due to the use of the right hand, which could potentially serve as features to filter out invalid taps. Despite



(a) (b)

Fig. 2. (a) Illustration of the tapping angle α, β and γ. Black dots indicated the start and lowest point during a tap. Purple indicated the tap vector, red line indicated its projection on different planes. (b) Correlation between the target angle and γ

the tendencies, there were still significant varieties of γ across different users even for the same target angle ($SD = 44.94°$).



(a) (b) (c)

Fig. 3. The average (a) α and (b) β for individual keys, merged from all keyboard sizes. Color indicated the value.

α and β were calculated as the slope of the tap vector in y-z and x-z plane, respectively. Figure 3 showed the average α (vertical) and β (horizontal) for each key across different keyboard sizes. The average α in the top, middle and bottom rows were 79.9° (SD = 7.6), 77.3° (SD = 10.6) and 69.8° (SD = 11.6) respectively. A significant effect of *row* on α was found ($F_{2,22} = 47.8, p < .001$), suggesting that the users' taps became more vertical for higher rows. On x axis, the maximum value of β was reached on key 'H' and 'J', and the value tended to decrease as the keys approached the left or the right edge. These results suggested that the users' finger placement strategy was similar to that on physical keyboards, given that key 'J' was also the default position for the right index finger.

3.4.4 Aiming Model. Similar with touch on touchscreens, there are multiple factors that determine the users' mental model when aiming at the keys [4, 16]: 1) Which part of their fingers do the users use to aim at the keys? 2) Which moment during the tap indicated the intended touch location? 3) How to project the 3D location of the finger onto the 2D keyboard? To explore these questions, we extracted the possible values for each factor, built

candidate models by combining these factors, and tested the systematic offset of different candidate models on the keyboard plane. We believed the model with the smallest offset would represent the user's actual mental model during single-finger mid-air typing [16].

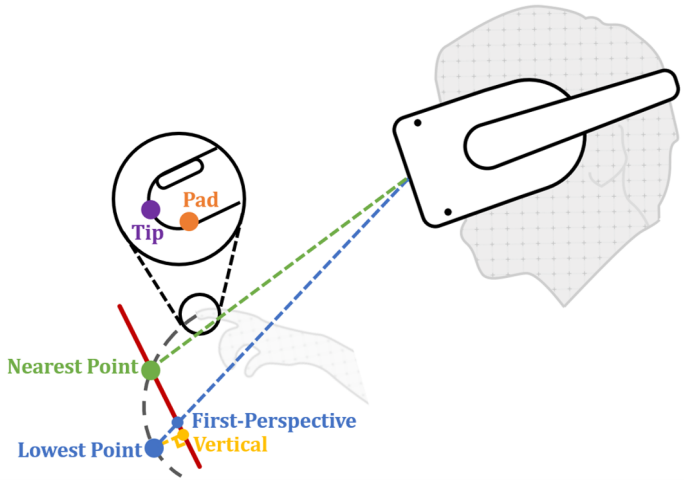


Fig. 4. Illustration of the factors in the projection model.

According to the three problems above, we derived three factors (see Figure 4): *finger position* (tip V.S. pad), *moment* (lowest V.S. nearest) and *projection* (vertical V.S. first-perspective), which generated 8 candidate models. Figure 5 showed the mean offset of all the taps calculated using different candidate models. Paired t-tests found that *finger position* ($t_{11} = -7.48, p < .001$), *moment* ($t_{11} = -3.45, p < .01$) and *projection* ($t_{11} = 2.44, p < .05$) all yielded significant effect on the touch offset. Among all the candidate models, Lowest Point + Vertical Projection + Tip achieved the lowest mean offset on all three keyboard sizes (4.93mm, 4.56mm and 5.38mm for increasing keyboard sizes). This implied that the users tended to use their fingertips for aiming, and they vertically project the lowest point during a tap (no matter whether the tap crossed the keyboard plane) to the keyboard as their intended location. Interestingly, this was also inconsistent with the top-down perspective model on touchscreens [16]. We used this model for inferring the touch points henceforth.

3.4.5 3D Touch Point Distribution. We analyzed the systematic offset and size of spread. The x and y coordinate of the touch points were calculated using the above aiming model, and z coordinate was the value of the lowest point during a tap. Figure 6 illustrated the touch point distribution on x-y and x-z planes respectively. Similar to touchscreen keyboards [4, 46, 47], the touch points roughly followed a Gaussian distribution. And the collected touch points consistently landed to the lower left direction of the target key. However, the z coordinate of the touch points were relatively dense, compared to x and y axis, no observable trend was found on the z axis.

We further analyzed the effect of keyboard size on systematic offset and size of spread, as shown in Figure 7. The large keyboard yielded significantly greater systematic offset than the other two sizes on y ($F_{2,22} = 4.37, p < .05$) and z ($F_{2,22} = 3.48, p < .05$) axis, but no significant difference on x offset was found ($F_{2,22} = 0.86, p = .43$). The x and y offset was greater than on touchscreens [4]. The observable negative z offset indicated that the users perceived the center of the target keys "below" the keyboard plane.

As expected, the size of spread in all the dimensions increased monotonously with keyboard sizes. The spread sizes in x and y axis were close, which were about twice of that on z axis. The spread size in all dimensions was significantly greater than that on touchscreens [4], confirming a more noisy input in mid-air.

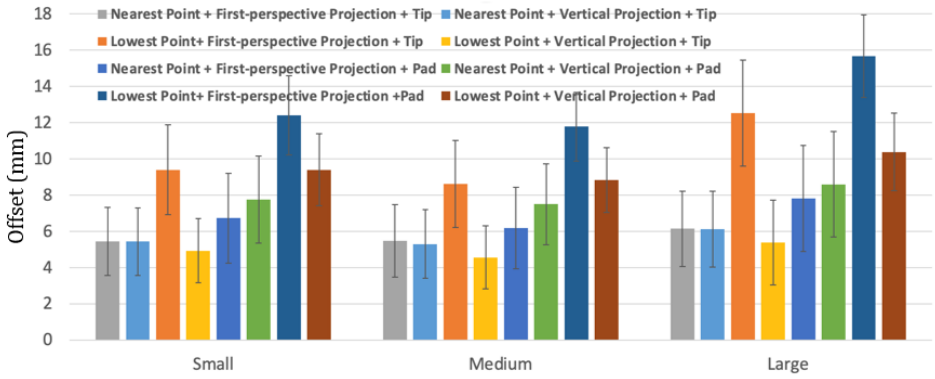


Fig. 5. Mean Offset of different candidate models on different sizes of keyboard. Error bar showed one standard deviation.

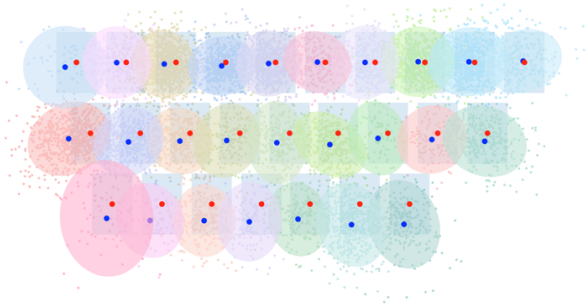
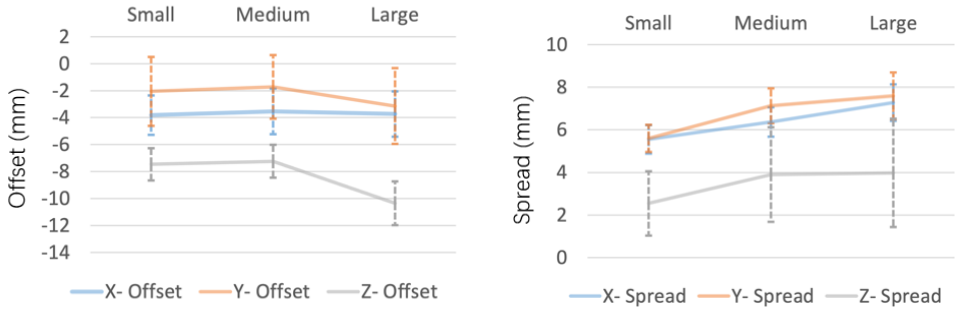


Fig. 6. Collective touch points on the large keyboard with 95% confidence ellipses, red and blue dots indicated the center of the key and the point cloud respectively.



(a) (b)

Fig. 7. (a) Systematic offset and (b) size of spread on different keyboard sizes. Error bar indicated one standard deviation.

3.4.6 Subjective Preference. We used a 7-point Likert-scale questionnaire to gather the participants' subjective ratings on different keyboard sizes. Figure 8 showed the results. Friedman test showed that keyboard size yielded a significant effect on all the dimensions (perceived speed: $\chi^2(2) = 7.95, p < .05$, perceived accuracy: $\chi^2(2) = 23.5, p < .001$, naturalness: $\chi^2(2) = 15.8, p < .001$, fatigue: $\chi^2(2) = 6.84, p < .05$ and overall preference: $\chi^2(2) = 11.9, p < .01$). Generally, the large keyboard received the highest rating on all dimensions except the perceived speed. Interviews confirmed that the participants preferred the larger keyboard for they felt more comfortable and confident when typing on it. In comparison, they typed carefully on the small keyboard, which lead to more physical and mental effort.

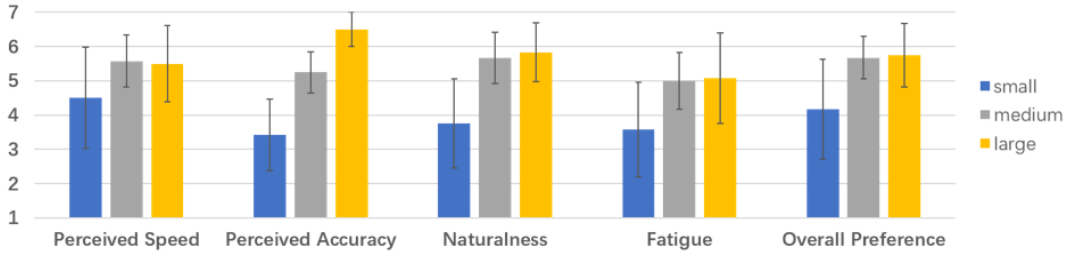


Fig. 8. Subjective ratings of different keyboard sizes (7: most positive, 1: most negative). Error bar indicated one standard deviation. Higher scores indicate better performances or experiences (higher speed, higher accuracy, more natural, less fatigue, and better performance.

4 PROBABILISTIC TOUCH MODELING ALGORITHM

We have shown that single-finger typing in mid-air yielded distinct patterns than touchscreens, which brought difficulty to tap detection and input prediction. In this section, we proposed a novel algorithm that generalized the classical Bayesian decoding algorithm by introducing probabilistic touch detection and elastic probabilistic decoding.

4.1 Algorithm Design

We first formally define the problem of input prediction in single-finger mid-air typing: Given a continuous stream of 3D tracking data of the tapping finger $I = a_1 a_2 \cdots a_N$, where $a_i = [x_i, y_i, z_i]^T$ refers to the 3D position of the tapping finger in the i^{th} frame. The goal is to find the word $W = c_1 c_2 \cdots c_m$ (c_i denoted the i^{th} character of W) in a predefined dictionary with the highest probability $P(W|I)$. According to Bayesian theory, $P(W|I) \propto P(I|W)P(W)$, where $P(W)$ is computed from the desired language model and $P(I|W)$ is the probability term that we focus on.

Note that I is the whole trace of the user's 3D input that contains both the segments with tap intention and the segments indicating word-to-word movement or hesitation. To efficiently compute $P(I|W)$, we aim to derive all the discrete tap events from the continuous input data. To achieve this, we devise a two-stage tap detection algorithm which first 1) assign a probability p_i to each frame a_i as the probability of it being a frame within a tap (rather than traveling between keys) and 2) merges the consecutive frames with tap intention into individual taps. For stage 1, the probability of a certain frame a_k being a tap event can be represented as:

$$p_k = P(TAP|a_k; I) = M([a_{k-l}, a_{k-l+1}, \cdots, a_{k+l-1}]) \tag{4}$$

where M is a tap probability prediction model (e.g., SVM) and $[a_{k-l}, a_{k-l+1}, \cdots, a_{k+l-1}]$ are the neighbouring frames of a_k. Such an input series has an observation window size of $2 \times l$ (with a systematic delay of $l - 1$

frames) that is expected to cover the whole touch-down-to-touch-up process. For stage 2, a cut-off threshold τ is applied to extract all the a_i such that $p_i > \tau$. Suppose $a_r, \cdots, a_{r+s}$ is the local longest consecutive series with tap intention ($a_{r-1} \leq \tau$, $a_{r+s+1} \leq \tau$, and $a_r, \cdots, a_{r+s} > \tau$), the medium element $a_{r+[\frac{s}{2}]}$ and its probability $p_{r+[\frac{s}{2}]}$ is then extracted to represent the series as the tap point. After applying stage 1 and 2, we acquire the tap point candidates $T = a_{r_1} a_{r_2} \cdots a_{r_n} = t_1 \cdots t_n$ ($t_i = [x'_i, y'_i, z'_i]^T$), along with their probability $(p'_1, p'_2, \cdots, p'_n)$.

The above two-stage tap detection algorithm discretizes $P(I|W)$ into $P(T|W)$. Ideally, T and W should have the same length, i.e., $n = m$. In this case, assuming that taps were independent from each other, we could compute

$$P(T|W) = P(t_1 t_2 \cdots t_n | c_1 c_2 \cdots c_n) = \prod_{i=1}^{n} P(t_i|c_i) \tag{5}$$

As t_i not only included the 3D location of the tap, but also quantified the probability of the tap, we have:

$$P(t_i|c_i) = P([x'_i, y'_i, z'_i]|c_i) \times p'_i \tag{6}$$

where $P([x'_i, y'_i, z'_i]|c_i)$ could be calculated using a 3D Gaussian model or 2D Gaussian model by ignoring the z dimension of touch position.

The above decoding algorithm should be effective when $n = m$, though not well suitable for mid-air single-finger typing where "insertion errors" (false positives) and "omission errors" (false negatives) frequently happen during tap detection due to the variety in the tapping process. To tolerate such errors, we proposed an elastic probabilistic decoding algorithm that shared similar design principles with existing works [42]. Specifically, we defined $D_{i,j}$ as the elastic conditional probability of $P(t_1 \cdots t_i | c_1 \cdots c_j)$, which could be calculated as:

$$D_{i,j} = \begin{cases} D_{i-1,j-1} \times (1 - P_{ins} - P_{omi}) \times P(t_i|c_j) & \text{– matching} \\ D_{i-1,j} \times P_{ins} & \text{– insertion} \\ D_{i,j-1} \times P_{omi} & \text{– omission} \end{cases} \tag{7}$$

where $D_{0,0} = 1$, P_{ins} and P_{omi} modeled the false positive and the false negative rate of the tap detection algorithm, respectively. $D_{n,m} = P(T|W)$ indicates the optimal matching. Specifically, if we set $P_{ins} = P_{omi} = 0$, Equation 7 would degenerate to Equation 5. In the implementation, we calculate $D_{n,m}$ using dynamic programming. Combining Equation 6 and Equation 7, our proposed algorithm generalized the existing statistical decoding algorithms by incorporating probabilistic touch detection and elastic probabilistic decoding. To our knowledge, our algorithm is the first to combine probabilistic touch modeling with elastic probabilistic decoding to model mid-air one-finger touch-based text entry.

4.2 Implementation Details

Here we further clarify the implementation details, including the algorithm details, parameter selection, and evaluations of certain components.

4.2.1 Tap Detection. As described above, tap detection was achieved by the two-stage tap detection algorithm. Due to the significant variance in tapping data, simple rule-based algorithms (e.g., based on intersection) could not achieve good performance in the mid-air typing scenario. Therefore, we trained an end-to-end SVM model with RBF kernel for tap detection as the model described in Equation 4. Such a model took the z coordinates in the data and predicted the probabilities of the data being a tap. The predicted probabilities were calibrated by Platt scaling [29] that performed an extra logistic regression on the SVM's scores by cross-validation on the training data.

We built the dataset for training with the data from Study 1. For each tap, we extracted the frames around the labeled tap point (or the lowest point) within a time window. A bigger time window could potentially increase the recognition accuracy, but at the cost of interaction delay in real use. Based on the findings in Sec 3.4.2 that a

full tap typically happened at around 370ms, we set the time window to be 10 frames (or 170ms covering the trajectory near the turning point), with a delay of around 70ms. We further augmented the positive samples by shifting them forward and back for 1–2 frames. We also randomly sampled a similar amount of negative samples from the typing data that were not labeled as taps. In total, we got 44,318 and 44,302 positive and negative samples, respectively.

We followed a typical leave-one-user-out cross validation process to evaluate the performances of the two models. For each fold, we train the SVM on the touch data of $N-1$ users and test it on the remaining user's data. We achieved an average test set accuracy of 99.2% for the SVM model, which is a promising result for recognizing individual frames from the test set.

Moreover, to better evaluate the models' real-time performance and determine the optimal recall threshold τ (see Section 4.1), we conducted a leave-one-user-out real-time simulation with $\tau = 0.1, 0.2, \cdots, 0.9$. For each fold, we train the SVM on the data of $N-1$ users and test it on the remaining user's data. For each trace at each recall threshold, we ran the model on each position and marked whether it referred to a tap based on the target recall threshold. We then followed the second stage of our tap detection algorithm (described in Section 4.1) to merge all the consecutive tap frames into a tap whose center is the median of the centers of the merged frames. Finally, we compared the predicted touch points with the labeled touch points on each trace and computed the F-1 score. Figure 9 showed the average F-1 score with different τ values, with the optimal F-1 = 0.977 reached at $\tau = 0.8$. The corresponding precision and recall were 0.972 and 0.983, respectively. We thus set $\tau = 0.8$.

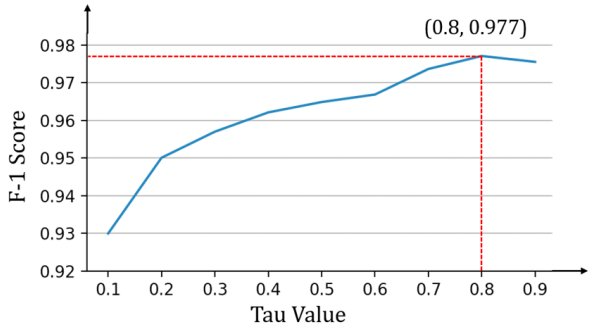


Fig. 9. The average F-1 score with different τ values.

4.2.2 Input Prediction. We used the elastic probabilistic decoder (Equation 7) to predict the target word based on the detected taps, where P_{ins} and P_{omi} were set to 0.03 and 0.02 according to the result above. As this algorithm would traverse the entire dictionary for each input word, which was not necessary in real use, we introduced an *elasticity* parameter to restrict the search space by constraining the maximum number of insertion and omission errors.

To guide the candidate searching procedure, we first built a TRIE tree based on the language model. Each node in the trie represented a valid prefix in the language model with its frequency. Then we used incremental computing and pruning techniques to recursively search the trie. At each searching step, the state was represented as a tuple $([t_1, \cdots, t_n], [c_1, \cdots, c_m]$, p, e). $[t_1, \cdots, t_n]$ refers to the remaining input points to decode and $[c_1, \cdots, c_m]$ referred the character prefix of the current state, which is consistent with the notation in Section 4.1. p referred to the probability of the current state while e was the abbreviation of *elasticity*, representing the remaining errors (i.e., omission and insertion) to tolerate. If the prefix is invalid, the searching path would be abandoned. If

$e = 0$, the rest input points are matched with a rigid decoder. Elsewise, we first fetch the possible character c_{m+1} to append based on the first touch point and recursively update the searching state as follows:

$$([t_1, \cdots, t_n], [c_1, \cdots, c_m], p, e) \rightarrow \begin{cases} ([t_2, \cdots, t_n], [c_1, \cdots, c_m, c_{m+1}], p \times P(t_1|c_{m+1}; LM), e) \\ ([t_2, \cdots, t_n], [c_1, \cdots, c_m], p \times P_{omi}, e-1) \\ ([t_1, \cdots, t_n], [c_1, \cdots, c_m, c_{m+1}], p \times P_{ins}, e-1) \end{cases} \tag{8}$$

where the first line represents a normal update to pop an input point, append a character to the prefix, and update the probability base on the language model. The second and third lines represent an omission update and an insertion update respectively. We accelerated the real-time computation speed by 1) pruning the invalid prefix at each searching step, 2) pruning the character ch such that $P(t_1|c_{m+1}) < \epsilon$ based on a Gaussian model (e.g., pruning the keys far away from t_1), and 3) pre-calculating the Gaussian probability pair $P(t_i|c_j)$.

4.3 Performance Simulation

4.3.1 Simulation Design. In this part, we conducted a simulation to test the performance of our algorithm, as well as explore the performance of different algorithm designs (e.g., parameter value). We used all the typing data on the large keyboard in Study 1 as the input, and calculated the top-1, top-5 and top-20 accuracy of the test models. We built a dictionary with the top 10,000 words in the American National Corpus [24] that cover over 90% of written English words [26], as previous work [45] did. We followed Yang et al. [45]'s work to build a bigram language model from the Google Web 1T 5-gram database [6] smoothed by the Katz's back-off model [17]. We explored four design factors in the simulation:

- *Geometry model of touch point (2D v.s. 3D).* Intuitively, $P([x'_k, y'_k, z'_k]|c_i)$ should be calculated using the 3D Gaussian model for each key computed from the prior (Study 1's) statistics. However, the z value of the touch points exhibited significant variety compared with x and y value (see Figure 6 and Figure 7), which may cause confusion during decoding. Therefore, ignoring the z value may increase the decoding performance.
- *Tap probability (with v.s. without).* Ignoring tap probability means setting $p_k \equiv 1$ in Equation 6, which was equivalent to the classical deterministic modeling for each touch point.
- *Language model (unigram v.s. bigram).* The selection of language model influences how $P(W)$ was estimated. We compared two common language models - unigram and bigram - in our simulation. The unigram model took the normalized word frequency as the probability while the bigram model took the conditional probability $P(W_2|W_1) = \frac{Count(W_1W_2)}{Count(W_1*)}$ from the corpus given the previous input word W_1. Typically, using a language model with higher order could effectively improve the decoding performance [40, 47], but at the cost of more computing cost and larger memory.
- *Elasticity (0, 1 and 2).* A greater elasticity value could increase the tolerance to insertion and omission errors, but may also increase input ambiguity. Elasticity = 0 was equivalent to the classical statistical decoding algorithm [14].
- *Tap detection model.* Although we analyzed the performance of our tap detection model for individual frames and streaming settings in Section 4.2.1, the joint effect of the tap detector with the improved decoder still remained unknown. So we performed an ablation study on the optimal decoder settings (determined in simulation) by replacing our tap detector with a naive z-threshold based detector. All the points lower than the keyboard plane were recognized with tap points and we merged the consecutive points following stage 2 of our tap detector (Section 4.1). We investigated two elasticity settings - 0 and 2 - in the simulation.

The above factors constitute $2 \times 3 \times 2 \times 2 = 24$ possible algorithms along with 2 ablated algorithms. However, when elasticity = 0, all the touches would be involved in the calculation and whether involving tap probabilities

would not change the relative order of different words. Therefore, we merged these two conditions, yielding 20 + 2 (ablation) algorithms in total. For reference, we also involved an "oracle" algorithm that used the manually labeled touches as ground truth for decoding, which indicated the upper bound of the performance.

4.3.2 Results. Table 1 showed the simulation results. Below we analyze the effect of each factor on the decoder's performance.

Geometry model: Generally, using the 2D touch model achieved higher performance than the 3D touch model, confirming that the z value of touch points was too noisy for decoding. This was probably due to users' lack of perception and concentration on the finger movement in the z-axis and the cross-user cognitive inconsistency in tapping behavior, as we observed in Study 1.

Elasticity: Increasing elasticity from 0 to 1 yielded significant improvement in performance, which was not surprising as touch detection was difficult to be perfect in mid-air typing. Further increasing elasticity to 2 led to a slight improvement in performance for both unigram and bigram model, while the improvement was much smaller for the bigram model due to the introduction of a stronger language model. Generally, elasticity = 2 is an optimal selection for performance and efficiency.

Table 1. Top-1, top-5, and top-20 word-level decoding accuracy (%) of different algorithms in simulation. Standard deviations showed in parenthesis. The algorithm with the highest performance was highlighted in bold.

Elasticity	TapProbability	Unigram						Bigram					
		2D			3D			2D			3D		
		Top 1	Top 5	Top 20	Top 1	Top 5	Top 20	Top 1	Top 5	Top 20	Top 1	Top 5	Top 20
0	-	77.7 (3.0)	81.5 (2.8)	81.9 (2.7)	76.6 (2.7)	80.9 (2.1)	81.4 (2.1)	80.6 (1.9)	83.3 (2.3)	83.7 (2.4)	80.4 (2.6)	83.3 (2.6)	83.5 (2.7)
1	Without	83.1 (2.3)	91.7 (2.1)	93.2 (1.8)	81.6 (2.5)	90.2 (1.9)	91.8 (2.2)	85.7 (2.4)	93.9 (1.6)	95.2 (1.5)	85.3 (3.1)	93.7 (1.9)	95.3 (1.7)
	With	83.7 (1.6)	91.8 (1.4)	93.3 (1.7)	81.9 (3.0)	90.4 (2.1)	92.1 (2.1)	**85.8** (1.6)	93.7 (1.4)	95.2 (1.3)	85.5 (2.1)	93.9 (1.7)	95.4 (1.4)
2	Without	83.8 (1.6)	93.3 (1.6)	94.8 (1.0)	82.3 (2.3)	91.6 (1.5)	93.9 (1.8)	85.5 (2.1)	94.2 (1.1)	96.0 (1.0)	84.9 (2.2)	93.7 (1.5)	96.0 (1.2)
	With	84.3 (2.5)	93.3 (1.6)	94.9 (1.4)	82.6 (2.3)	91.7 (1.9)	93.8 (2.0)	85.6 (3.3)	**94.2** (2.1)	**96.1** (1.5)	85.1 (1.8)	93.9 (1.2)	95.9 (1.0)
Oracle		88.4 (2.2)	96.7 (0.8)	97.8 (0.7)	87.9 (1.9)	96.2 (2.3)	97.1 (2.3)	95.1 (1.8)	99.1 (0.6)	99.5 (0.5)	95.1 (1.6)	99.1 (0.8)	99.5 (0.5)

Tap probability: Leveraging the tap probability could further facilitate the performance, as taps with different trajectories were assigned different weights during decoding (e.g., emphasize the deep and fast taps rather than flat and slow taps). We observed that the improvement is significant for top-1 accuracy but insignificant for top-5 and top-20 accuracy, which was understandable because the effect of tap probability was weaker than that of the touch points' geometry probability and the probability decay of insertion and omission so that it would influence more on the local ranking rather than the retrieved points. Also, the effect of tap probability would be overwhelmed by a stronger language model (e.g., bigram) due to its contribution to reranking.

Language model: Regarding the language model, using the stronger bigram model leads to a constant boost in decoding performance compared with the unigram model. However, we also observed that the effect of the other factors (e.g., the optimal elasticity, the use of touch probability, and the 3D Gaussian model) was weakened in the bigram setting due to the dominant effect of the strong language model.

Tap detection model: Removal of the tap detection model while preserving the optimal decoder (2D, with tap probability) yielded top-1, top-5, and top-20 accuracies of 58.5% (SD=21.5%), 61.6% (21.9%), and 61.8% (22.0%) for elasticity = 0 and 70.3% (15.7%), 80.9% (13.9%), and 84.4% (12.4%) for elasticity = 2. Using a naive threshold-based tap detector caused a significant drop (e.g., 70.3% v.s. 85.8%) in decoding accuracy mainly due to the noise of the predicted taps. However, since the naive tap detector yielded more errors, the introduction of the elasticity (0 v.s. 2) led to significant performance improvement (58.5% v.s. 70.3%). We also found that threshold-based algorithms led to a significantly higher deviation in accuracy among different users, revealing the significant cross-user difference in tapping behavior (as validated in Study 1).

Combining the results, the 2D elastic probabilistic decoding algorithm (elasticity = 2) with tap probability under the bigram model yielded the optimal performance among all the algorithms, which was therefore used in the following user study. It was worth emphasizing that this algorithm achieved a top-1 accuracy over 85%, which was significantly higher than the classical statistical decoding algorithm (77.7% under the unigram model or 80.6% under the bigram model).

5 STUDY 2: INTERACTION PERFORMANCE EVALUATION

In this section, we evaluated the performance of the optimal algorithm in real text entry tasks. We were especially interested in the learnability and usability of the technique.

5.1 Participants and Apparatus

We recruited 16 participants from the local campus (10 male, 6 female) with an average age of 22.1 (SD = 1.7). All participants used the QWERTY keyboard for daily text entry and were familiar with the QWERTY layout.

5.2 Experiment Design

Our proposed algorithm has two main features: probabilistic tap detection and elastic probabilistic modeling. To evaluate the effect of the two features separately, we designed two factors for the study: *Tap detection* (probabilistic vs. collision-based) and *Input prediction* (elastic vs. rigid). The collision-based tap detection algorithm detected a tap deterministically by observing the intersection between the user's index finger and the keyboard plane and the rigid algorithm was equivalent to setting $elasticity = 0$ in our algorithm, which was not robust to insertion or omission errors. Combining the two factors, we designed four algorithms: PRO-ELS (probabilistic tap detection and elastic model, same as the optimal algorithm in the previous section), COL-ELS (collision-based tap detection and elastic model), PRO-RIG (probabilistic tap detection and rigid input prediction) and COL-RIG (collision-based tap detection and rigid input prediction).

To validate the efficiency and usability of our technique, we chose a competitive mid-air text entry technique - Vulture [25] - from the various techniques in existing literature and commercial products as the baseline. Such a selection was motivated by: 1) the similarity of interaction scenario and modality (e.g., one-handed mid-air text entry) with our technique and 2) the competitive performance (e.g., Vulture claimed a pick-up text entry speed of 20.6 WPM and a speed of 28.1 WPM after training). Notably, we did not choose techniques with more alike interaction modality such as VISAR [8] because: 1) their reported performances were not as high as Vulture; 2) VISAR adopted a collision-based touch detector with a sentence-level statistical decoder, which was quite similar with our COL-ELS setting; and 3) VISAR's decoder was far more heavier in computation than ours (e.g., a 12-gram character model and a 4-gram word model with more than 1.5 GB model size for VISAR).

An informal pilot study by three experimenters showed poor performance of the COL-RIG setting (14.7 WPM for expert users), which was far lower than the other three techniques and might bring unnecessary workload to the experiment. Therefore, we pruned out such a setting and only tested the other three techniques - PRO-ELS, COL-ELS, and PRO-RIG - along with the Vulture baseline in the evaluation. Our three techniques shared the same interaction and UI design and were only different in the underlying algorithm.

As for the implementation of our techniques, we designed the keyboard layout to show the top-5 candidate words during typing (see Figure 10(a), the top-1 candidate was shown in the input block and the first candidate showed the raw input). Considering hitting the space key for confirmation was most frequently performed, though tiring and inefficient, we bound the confirmation function (e.g., automatically entering the top-1 candidate followed by a space) with a thumb-to-index-finger pinching gesture. Users could also tap on other candidate words to select them or pinch the thumb and middle finger to delete the last entered word. During our implementation, we found the hand tracking system of Quest 2 frequently failed to report or misreported the two pinching events

(a) Interface of our proposed techniques

(b) Interface of the implemented Vulture gesture keyboard

Fig. 10. Experiment platforms of our techniques and Vulture. UI styles were adjusted for better visualization. The figures showed the UIs of free input mode (no targets provided). For the experiment mode, a textbox with target captions was displayed above the text input area.

while the right hand was holding up and performing single-finger typing. Given that the two pinching gestures were designed for the confirmation or delete phrase and rarely overlapped with the tapping events in the temporal domain, we transferred the two gestures to the user's left hand for better text entry fluency (otherwise the misrecognition of the pinching gestures would severely influence the major typing experience provided by our technique).

To implement Vulture [25], we followed the technical details, including the mapping mechanism of the cursor and the SHARK 2 gesture decoder. The only difference was we adopted the same keyboard settings (the size, the position, and the orientation) as our techniques and set the CD-ratio to 1:1 (based on the hand's movement range) instead of placing the keyboard far away with the CD-ratio of 1:2 (Vulture's original settings) mainly to control the keyboard's appearance factors for better comparison with our techniques. Although different from the original settings, we argued that such a design was feasible because our implementation achieved 1) a larger angle of view (e.g., 24° V.S. 15° in horizontal) and 2) the same range and resolution of hand movement (e.g., a maximum horizontal distance of 20 cm for both settings) compared with the original Vulture. While typing with Vulture, the participant's mid-air index fingertip position was projected to the keyboard plane to control a visible cursor dot. The user inputs a word by placing the cursor on the first letter of the word, making a pinch gesture with his right thumb and index finger, drawing an approximate trace to cross every letter of the desired word, and finally releasing the pinch. The cursor trajectory was shown during the whole process (See Figure 10(b)).

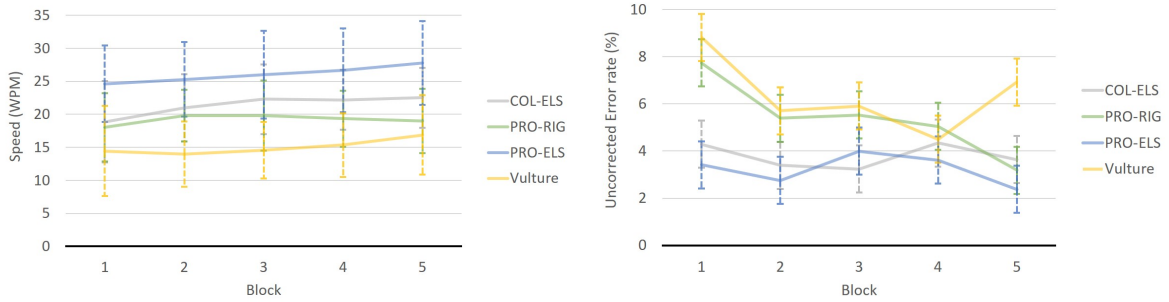
5.3 Procedure

The participants were first given 10 minutes to familiarize themselves with the interaction of both Vulture and our technique. Then they completed four sessions of text transcription tasks with the four techniques to evaluate (PRO-ELS, PRO-RIG, COL-ELS, and Vulture) respectively. The order among these techniques was counterbalanced with a balanced fourth order Latin square. Each session contained 5 blocks. In each block, each participant entered 6 phrases randomly sampled from the MacKenzie and Soukoreff phrase set [23]. They were instructed to type "as quickly and as naturally as possible" and could choose to correct the typing errors or leave them uncorrected freely. A two-minute break was taken between blocks.

5.4 Results

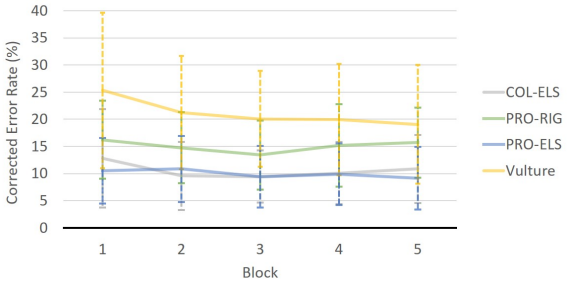
5.4.1 Input Speed. Figure 11(a) showed the text entry speed of different techniques over blocks. The average input speed of PRO-ELS, COL-ELS, PRO-RIG and Vulture over all blocks were 26.1 WPM (SD = 6.3), 21.4 WPM

(SD = 5.3), 19.2 WPM (SD = 4.8), and 15.4WPM (SD = 5.6), respectively. RM-ANOVA found a significant effect of *technique* on the input speed ($F_{3,45}$ = 20.75, p<.001). The text entry speed of PRO-ELS was 18.0% , 26.4%, and 41.0% faster than that of COL-ELS ($t(15) = 4.27, p < 0.01$), PRO-RIG ($t(15) = 5.4, p < 0.01$), and Vulture ($t(15) = 5.39, p < 0.01$) respectively, showing the efficacy of our optimal algorithm involving both probabilistic tap detection and elastic input prediction. Interestingly, the text entry speed of COL-ELS was faster than the PRO-RIG ($t(15) = 2.8, p < 0.05$), indicating that the performance of the decoding algorithms yielded a more dominant effect on the text entry speed than tap detection algorithm. Results showed a lower speed among our participants than the speed reported in Vulture [25] (e.g., 16.9 WPM v.s. 20.6 WPM in the last block), probably due to a mutual effect of proficiency in gesture typing and English text entry (e.g., all the participants were ESL) and the difference in Vulture's UI implementation. Notwithstanding, our results showed the advantage of PRO-ELS over Vulture no matter with the same keyboard layout or in an empirical numerical comparison.



(a) Text entry speed

(b) Uncorrected error rate



(c) Corrected error rate

Fig. 11. (a) Text entry speed, (b) Uncorrected error rate, and (c) Corrected error rate of four techniques over blocks. Error bar indicated one standard deviation.

The input speed of all four techniques tended to increase with blocks, suggesting a learning effect. The average speed of PRO-ELS was 24.6 WPM (SD = 5.8) in block 1 and increased monotonously to 27.8 WPM (SD = 6.4) in block 5. Significant effect was found between block 1 and block 5 on the text entry speed (t(15)=2.816, p<0.02).

PRO-ELS reached a pick-up speed of 24.6 WPM in the first block, which was approximately the same as the speed in Study 1 (assuming a perfect algorithm). This suggested that the keyboard algorithm could effectively

predict the users' input. In block 5, the input speed increased to 27.8 WPM, which was competitive with ten-finger typing [48], double-finger typing[7] and gesture typing [25] in mid-air, and was about 72% of the index-finger typing speed on smartphones [4]. The top 4 users with the highest input speed even reached an average speed of 36.8WPM (SD = 3.2) in block 5, proving the potential of our technique.

5.4.2 Error Rate. Figure 11(b) showed the uncorrected word-level error rate of the different techniques. Across all blocks, the average uncorrected error rate of PRO-ELS, COL-ELS, PRO-RIG, and Vulture were 3.2% (SD=4.4%), 3.8% (SD=5.0%), 5.4% (SD=8.2%), and 6.4% (SD=8.3%), respectively. No significant difference was found between these techniques ($F_{3,45} = 2.45, p = 0.08$). When using PRO-ELS, the average error rate ranged between 2.4% and 4.0%. No significant effect of *block* was found ($F_{4,60} = 0.8, p = .53$). Recalling the results on input speed, we could find that with practice, the users could improve their typing speed without sacrificing accuracy.

Furthermore, we found that the average uncorrected error rate of PRO-RIG was 5.4%, a bit higher than existing work [8, 25, 48]. To figure out the reason, we interviewed several participants about why they left errors unfixed while using PRO-RIG. Most participants complained about the rigid input prediction algorithm made it more difficult for them to type in the correct word than the elastic algorithm so that they were too exhausted to fix the errors.

The average corrected error rate of PRO-ELS, COL-ELS, PRO-RIG, and Vulture were 10.0% (SD = 5.9%), 10.6% (SD = 6.7%), 15.1% (SD = 7.0%), and 20.0% (SD = 11.5%). Significant effect ($F_{3,45} = 17.943, p < .01$) was found between different techniques. Participants tend to correct more errors with Vulture rather than with our techniques. As for PRO-ELS, COL-ELS with PRO-RIG, we found using the elastic predicting algorithm would yield significantly fewer mistakes. For the optimal technique PRO-ELS, no significant effect($F_{4,60} = 0.281, p = .889$) of block on corrected error rate was found.

5.4.3 Interaction Statistics. We looked into more details on how users interacted with the different keyboard techniques. As shown in Table 2, when using PRO-ELS, the most frequent operation was *Match*, i.e., type a word and confirm (83.8%). This was consistent with the input speed and accuracy results above. For 4.5% of the operations, users *Selected* another candidate word from the list. For 6.2% of the operations, users *Cancelled* the taps they had committed, probably because the intended word was not in the candidate list. And for 5.6% of the operations, users *Deleted* an entered word to fix errors.

Table 2. Frequency of different word-affecting actions. (N = 3,790)

Class	Match		Select		Cancel	Delete
Correctness	Yes	No	Yes	No		
PRO-ELS	2423 (76.2%)	243 (7.6%)	125 (4.0%)	15 (0.5%)	196 (6.2%)	179 (5.6%)
COL-ELS	2334 (76.1%)	210 (6.8%)	155 (5.1%)	12 (0.4%)	214 (7.0%)	143 (4.7%)
PRO-RIG	2389 (71.8%)	269 (8.1%)	74 (2.2%)	23 (0.7%)	416 (12.5%)	154 (4.6%)
Vulture	1768 (54.1%)	287 (8.8%)	667 (20.4%)	59 (1.8%)	275 (8.4%)	215(6.6%)

Observably, the ratio of *Cancel* while using PRO-RIG (12.5%) was significantly higher than PRO-ELS (6.2%) and COL-ELS (7.0%). This implied that users had to cancel the incorrect words they entered frequently, probably due to the insertion and omission errors of the tap detection algorithm. Recall that the input speed of COL-ELS was also higher than PRO-RIG, this also highlighted the necessity of the elastic input prediction algorithm. When using Vulture, participants selected candidates more than using PRO-ELS, suggesting Vulture had lower top-1 accuracy than PRO-ELS.

5.4.4 KSPC and Time Intervals between Adjacent Taps. We calculated keystroke per character (KSPC)[36] to measure the input performance for PRO-ELS, COL-ELS, and PRO-RIG (Vulture not included for the absence of keystroke). A lower KSPC indicates that the user expands less effort in modifying the input. The KSPC of PRO-ELS, COL-ELS, and PRO-RIG were 1.12 (SD=0.22), 1.10 (SD=0.23), and 1.23 (SD=0.30) respectively, which was significantly different ($F_{2,30} = 20.975, p < .01$). Post-hoc comparison found significant differences of (PRO-ELS, PRO-RIG) (t(15)=5.38, p<0.01) and (COL-ELS, PRO-RIG) (t(15)=4.93, t<0.01), and no significant difference between PRO-ELS and COL-ELS (t(15)=0.98, p=0.34)

We also calculated the time interval (TI) between adjacent taps for character keys, which could reveal the users' typing speed and confidence towards the algorithm (Vulture not included for the absence of keystroke). The average TI of PRO-ELS, COL-ELS, and PRO-RIG were 361.1ms (SD=218.4), 440.1ms (SD=317.6), and 448.6ms (SD=357.9) respectively, which was significantly different ($F_{2,30} = 9.39, p < .01$). Post-hoc comparison found that the TI of PRO-ELS was significantly shorter than the other two techniques.

Considering the above results of KSPC and TI, we have two observations of users' behavior: (1) significant lower TI of PRO-ELS compared with COL-ELS indicated given the same decoder, users were more confident and tended to type faster with the stronger tap detector; (2) similar KSPC between PRO-ELS and COL-ELS showed users tended to slow down their move with collision-based touch detector to reach a comparable input accuracy; and (3) removing the elasticity meant no tolerant for the touch point error and required users to input the touch point precisely in temporal, leading to additional mental burden, higher KSPC, and lower TI.

5.4.5 Subjective Results. We used the same questionnaire as in Study 1 to gather the participant's subjective ratings towards the four techniques. Figure 12 showed the results. PRO-ELS received an average score over 6 on all dimensions, showing the strong acceptance of our optimal technique among all participants. In comparison, the ratings of COL-ELS, PRO-RIG and Vulture were lower. Significant effect of technique was found in terms of all the dimensions (perceived speed ($\chi^2(3) = 21.7$, p<.001), perceived accuracy ($\chi^2(3) = 26.5$, p<.001), naturalness ($\chi^2(3) = 26.2$, p<.001), fatigue ($\chi^2(3) = 26.4$, p<.001) and overall preference ($\chi^2(3) = 26.6$, p<.001)).

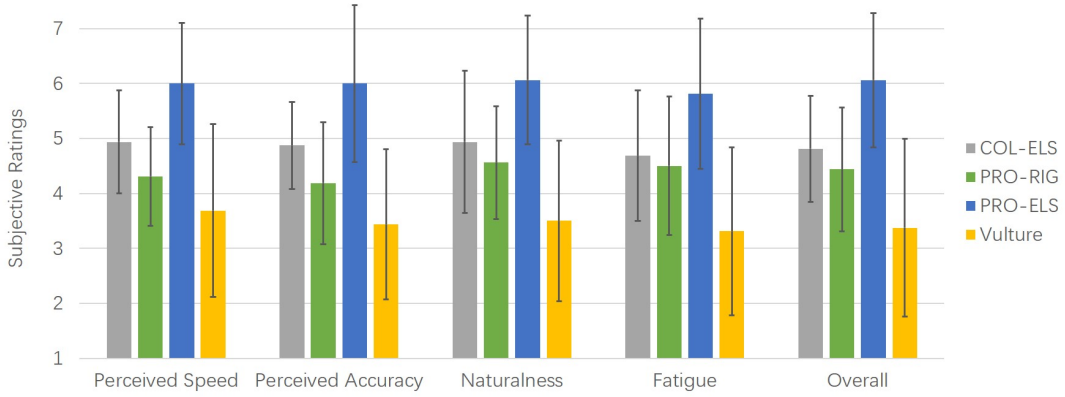


Fig. 12. Subjective ratings of keyboard using four techniques. (7: most positive, 1: most negative). Error bar indicated one standard deviation.

We also interviewed the participants about their comments and feedback. As expected, participants could learn single-finger typing in mid-air easily.

"It was quite similar to the typing experience on the smartphone, which makes me easily master it." (P1) "The technique was intuitive and effective! I could pick up it in a short time to achieve a satisfying performance." (P4)

Participants were satisfied with the input speed and accuracy of the keyboard, which increased their confidence during typing.

"The algorithms correctly predict the word I wanted to enter most of the time. It made it easier to complete text entry task." (P8)

"I was not confident at first, but after some practice, the high accuracy of the technique made me more confident during the task." (P14)

Some participants even hoped to use this keyboard in real life.

"I would be delighted if this keyboard can be deployed on the current VR and AR platform!" (P3)

"It is the ideal keyboard I would like to have for my HMDs." (P4)

Due to the design of single-finger typing, the participants were satisfied with the fatigue of the keyboard.

"Although I cannot type fast in the first block while using PRO-ELS, I can easily master the typing skill in the last block ."(P12)

"I don't have to touch the keyboard every time, which made it more natural while typing." (P9)

"While using gesture input, I have to keep my arm up in the whole process. It cost me so much effort. It is more relaxing when I use tap input method." (P14)

"I like using gesture keyboard, it is just as quick as tapping on the keyboard." (P5)

"It felt like typing on my smartphone. I will be more satisfied if it supports 10 finger typing." (P11)

6 DISCUSSION

6.1 Typing Behavior: 2D v.s. 3D

Although on a virtual plane, the users' typing behavior in mid-air was significantly different from that on 2D touchscreens. This not only resulted from the involvement of depth, but also from the absence of tactile feedback. The results in Study 1 showed that even on the small sized keyboard whose size was similar to smartphone keyboards, the participants' typing speed under ideal condition was only 23.2 WPM, 64% of the index-finger typing speed on touchscreens. This was mainly due to the additional movement distance in the z axis.

The finger kinematics results showed that compared with multi-finger typing [7, 48], single-finger typing yielded shorter tapping duration, faster tapping speed and shorter tapping amplitude. This suggested that the participants were more confident during typing without intentionally performing deep tapping, which was helpful for a more natural and fluent typing experience. The results on tapping direction also verified that users tended to mix the finger movement and tapping procedure to ensure a smooth transition.

As researched in previous work [7, 33], aligning a virtual keyboard to a physical surface restricts 3D typing behavior to a 2D surface, which benefits in both the user's subjective perception and the certainty of touches for the decoder. In our work, the SVM-based touch detection model focused on solving the latter issue - to recognize natural touches from the unrestricted 3D behavior. On the complementary side, an interesting research question is how to provide haptic feedback to the user in 3D typing scenarios to facilitate their perception and performance.

Determining the users' aiming model was a problem for any touch-based interaction [16] due to the fat finger problem, which was more complex in mid-air typing due to the continuous movement of finger. In this paper, we elicited eight candidate models that covered the major factors of the aiming model. Results showed that the participants tended to vertically project the lowest point of the tip during a tap to the keyboard as their intended location. This was consistent with the top-down perspective model on touchscreens [16], and would be helpful for resolving the input ambiguity in other mid-air touch interactions.

The touch point distribution followed well with the QWERTY keyboard on the x-y plane, with the touch points for individual keys following the Bivirate Gaussian distribution. However, both systematic offset and spread size were greater than that on touchscreens [4], confirming that typing in mid-air was more imprecise. Moreover, the

z coordinate of different keys did not yield an observable trend, suggesting that the users mainly perceived depth for tap detection rather than distinguishing different keys.

It was worth noting that the results in this paper were collected mainly on right-handed users, and were on specific keyboard settings (e.g., location and angle). Although changing these settings could affect the results to some degree, we believed that the intrinsic patterns in 3D touch were still valuable for researchers in future works. Further, we believe the result of our work (e.g., the touch kinematics) can be transferred to other more complicated cases such as bimanual text entry and multi-finger text entry, though the behavior of multi-finger typing and multi-finger typing could be different in terms of hand-to-hand and finger-to-finger cooperation and correlation, which is worthy of further research.

6.2 Towards Efficient Text Entry in Mid-air

The distinct pattern of mid-air typing made it difficult to perform tap detection and input prediction. To solve this problem, we proposed a probabilistic touch modeling algorithm based on the results in Study 1. As significant variance existed both in tap amplitude and tap direction, we used an SVM-based algorithm that calculated the possibility of the taps according to the z value of the finger trajectory. With the optimal threshold ($\tau = 0.8$), this algorithm yielded an F-1 score of 0.977 in simulation.

To further improve the input performance, we designed a probabilistic touch modeling algorithm that incorporated the probability of the taps with elastic probabilistic decoding. This was the first decoding algorithm that treated uncertain taps using probability. And compared with existing elastic probabilistic model [42], our algorithm also involved tap probability, and the parameters referred to the false positive and false negative rate of the tap detection algorithm, rather than different kinds of input errors generated by the user.

Simulation and user study results verified the effectiveness of the proposed algorithm. Compared with classical statistical decoding algorithm, our algorithm achieved a 8.5% higher top-1 accuracy. And participants reached a pick-up input speed of 24.6 WPM, close to that under ideal conditions (see Study 1). Also, single-finger typing facilitated learning due to its intuitiveness, allowing the participants to type 27.8 WPM in the last block. This was surprising as single-finger tap typing was expected to be slower than gesture typing [31] or multi-finger typing [4]. We also implemented Vulture as our baseline, and found out that our algorithm achieved evidently higher speed than Vulture without sacrificing accuracy.

This work has demonstrated the feasibility of single-finger mid-air typing, which we argued has the advantages of intuitiveness, efficiency and learnability, and was an attractive solution for commercial HMDs. In real use, the interaction performance could still be improved by using more complex algorithms (e.g., LSTM language model and deep learning for input decoding) or designing multi-modality feedback (e.g., visual and audio [20, 41]). We also planned to further study these aspects to explore the limit of mid-air text entry.

7 LIMITATION AND FUTURE WORKS

In our study, the interaction design (e.g., the confirmation scheme and the keyboard layout) was finalized based on an informal pilot interview with expert researchers, merely containing the minimized components to validate our algorithms. To facilitate practical design, more considerations and studies could be conducted. For example, as some participants mentioned, pinching with the other hand to confirm, which was primarily designed for high confirmation accuracy so that not impeding the typing behavior, could be restricted in specific scenarios where the non-dominant hand is occupied. Therefore, how to adapt a user-friendly interaction scheme to different scenarios is a question worth exploring.

As for the behavior model, there are factors (e.g., user's dominant hand, haptic feedback, and input posture) that could influence the pervasiveness of users' one-finger mid-air typing behavior in VR that are worth further investigation. Moreover, other modalities and techniques such as surface-aligned typing and multi-finger typing

for efficient VR text entry are significant research topics. Due to the limitation of the experiment load in this paper, we could only formally explore a part of them. And we planned further explorations in future work.

Regarding the algorithms, we followed a statistical decoding framework to develop the algorithms in this work. Meanwhile, we also envision other unique perspectives for mid-air text entry. For example, the user's hand and finger movement form a continuous trace which might provide informative clues in other perspectives such as gesture typing.

For the performance evaluation study, our study was conducted within a group of young ESL users. Further study on different user groups (e.g., aged users and motor-impaired users) could gain knowledge on their behavior features and typing performances in such a mid-air typing scenario. Additionally, our implementation on the Vulture baseline adopted a consistent UI design with our proposed techniques, which was different from the original Vulture setting. The difference in UI design might contribute to the drift in performance, although we did not find negative feedback regarding this issue from the users. Further study with more strictly controlled settings could be conducted to analyze the effects of UI design.

8 CONCLUSION

This paper explored the feasibility of efficient single-finger mid-air typing using probabilistic touch modeling. We first examined the users' typing behavior characteristics including fingertip kinematics during tapping, different aiming models and 3D touch point distribution. Based on the results, we incorporated probabilistic tap detection with elastic probabilistic decoding for input prediction, which significantly outperformed classical statistical decoding in simulation. In evaluation, participants typed 26.1 WPM with 3.2% uncorrected error rate after 5 blocks of practice. Our results demonstrated that efficient text entry in mid-air was achievable, and single-finger typing was potentially preferable, despite the absence of tactile feedback.

ACKNOWLEDGMENTS

This work was supported by the Natural Science Foundation of China (NSFC) under Grant No. 62132010, and the grant from the Institute for Guo Qiang, Tsinghua University No. 2019GOG0003.

REFERENCES

[1] Diar Abdlkarim, Massimiliano Di Luca, Poppy Aves, Sang-Hoon Yeo, R. Chris Miall, Peter Holland, and Joseph M. Galea. 2022. A Methodological Framework to Assess the Accuracy of Virtual Reality Hand-Tracking Systems: A case study with the Oculus Quest 2. *bioRxiv* (2022). https://doi.org/10.1101/2022.02.18.481001 arXiv:https://www.biorxiv.org/content/early/2022/02/20/2022.02.18.481001.full.pdf

[2] Jiban Adhikary and Keith Vertanen. 2021. Typing on Midair Virtual Keyboards: Exploring Visual Designs and Interaction Styles. In *IFIP Conference on Human-Computer Interaction*. Springer, 132–151.

[3] Christoph Amma, Marcus Georgi, and Tanja Schultz. 2012. Airwriting: Hands-Free Mobile Text Input by Spotting and Continuous Recognition of 3d-Space Handwriting with Inertial Sensors. In *2012 16th International Symposium on Wearable Computers*. 52–59. https://doi.org/10.1109/ISWC.2012.21

[4] Shiri Azenkot and Shumin Zhai. 2012. Touch Behavior with Different Postures on Soft Smartphone Keyboards. In *Proceedings of the 14th International Conference on Human-Computer Interaction with Mobile Devices and Services* (San Francisco, California, USA) *(MobileHCI '12)*. Association for Computing Machinery, New York, NY, USA, 251–260. https://doi.org/10.1145/2371574.2371612

[5] Sebastian Boring, David Ledo, Xiang 'Anthony' Chen, Nicolai Marquardt, Anthony Tang, and Saul Greenberg. 2012. The Fat Thumb: Using the Thumb's Contact Size for Single-Handed Mobile Interaction. In *Proceedings of the 14th International Conference on Human-Computer Interaction with Mobile Devices and Services* (San Francisco, California, USA) *(MobileHCI '12)*. Association for Computing Machinery, New York, NY, USA, 39–48. https://doi.org/10.1145/2371574.2371582

[6] Thorsten Brants and Alex Franz. 2006. Web 1T 5-gram Ver. 1.

[7] John Dudley, Hrvoje Benko, Daniel Wigdor, and Per Ola Kristensson. 2019. Performance envelopes of virtual keyboard text input strategies in virtual reality. In *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 289–300.

[8] John J. Dudley, Keith Vertanen, and Per Ola Kristensson. 2018. Fast and Precise Touch-Based Text Entry for Head-Mounted Augmented Reality with Variable Occlusion. *ACM Trans. Comput.-Hum. Interact.* 25, 6, Article 30 (Dec. 2018), 40 pages. https://doi.org/10.1145/3232163

[9] Leah Findlater and Jacob Wobbrock. 2012. *Personalized Input: Improving Ten-Finger Touchscreen Typing through Automatic Adaptation.* Association for Computing Machinery, New York, NY, USA, 815–824. https://doi.org/10.1145/2207676.2208520

[10] Leah Findlater, Jacob O. Wobbrock, and Daniel Wigdor. 2011. *Typing on Flat Glass: Examining Ten-Finger Expert Typing Patterns on Touch Surfaces.* Association for Computing Machinery, New York, NY, USA, 2453–2462. https://doi.org/10.1145/1978942.1979301

[11] Conor R. Foy, John J. Dudley, Aakar Gupta, Hrvoje Benko, and Per Ola Kristensson. 2021. Understanding, Detecting and Mitigating the Effects of Coactivations in Ten-Finger Mid-Air Typing in Virtual Reality. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 287, 11 pages. https://doi.org/10.1145/3411764.3445671

[12] Mayank Goel, Leah Findlater, and Jacob Wobbrock. 2012. *WalkType: Using Accelerometer Data to Accomodate Situational Impairments in Mobile Touch Screen Text Entry.* Association for Computing Machinery, New York, NY, USA, 2687–2696. https://doi.org/10.1145/2207676.2208662

[13] Mayank Goel, Alex Jansen, Travis Mandel, Shwetak N. Patel, and Jacob O. Wobbrock. 2013. *ContextType: Using Hand Posture Information to Improve Mobile Touch Screen Text Entry.* Association for Computing Machinery, New York, NY, USA, 2795–2798. https://doi.org/10.1145/2470654.2481386

[14] Joshua Goodman, Gina Venolia, Keith Steury, and Chauncey Parker. 2002. Language Modeling for Soft Keyboards. In *Proc. IUI'02* (San Francisco, California, USA). Association for Computing Machinery, New York, NY, USA, 194–195. https://doi.org/10.1145/502716.502753

[15] Shangchen Han, Beibei Liu, Randi Cabezas, Christopher D. Twigg, Peizhao Zhang, Jeff Petkau, Tsz-Ho Yu, Chun-Jung Tai, Muzaffer Akbay, Zheng Wang, Asaf Nitzan, Gang Dong, Yuting Ye, Lingling Tao, Chengde Wan, and Robert Wang. 2020. MEgATrack: Monochrome Egocentric Articulated Hand-Tracking for Virtual Reality. *ACM Trans. Graph.* 39, 4, Article 87 (aug 2020), 13 pages. https://doi.org/10.1145/3386569.3392452

[16] Christian Holz and Patrick Baudisch. 2011. *Understanding Touch.* Association for Computing Machinery, New York, NY, USA, 2501–2510. https://doi.org/10.1145/1978942.1979308

[17] S. Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 35, 3 (1987), 400–401. https://doi.org/10.1109/TASSP.1987.1165125

[18] Seoktae Kim, Minjung Sohn, Jinhee Pak, and Woohun Lee. 2006. One-Key Keyboard: A Very QWERTY Keyboard Supporting Text Entry for Wearable Computing. In *Proceedings of the 18th Australia Conference on Computer-Human Interaction: Design: Activities, Artefacts and Environments* (Sydney, Australia) *(OZCHI '06)*. Association for Computing Machinery, New York, NY, USA, 305–308. https://doi.org/10.1145/1228175.1228229

[19] Chen Liang, Chi Hsia, Chun Yu, Yukang Yan, Yuntao Wang, and Yuanchun Shi. 2023. DRG-Keyboard: Enabling Subtle Gesture Typing on the Fingertip with Dual IMU Rings. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 6, 4, Article 170 (jan 2023), 30 pages. https://doi.org/10.1145/3569463

[20] Jia-Wei Lin, Ping-Hsuan Han, Jiun-Yu Lee, Yang-Sheng Chen, Ting-Wei Chang, Kuan-Wen Chen, and Yi-Ping Hung. 2017. Visualizing the Keyboard in Virtual Reality for Enhancing Immersive Experience. In *ACM SIGGRAPH 2017 Posters* (Los Angeles, California) *(SIGGRAPH '17)*. Association for Computing Machinery, New York, NY, USA, Article 35, 2 pages. https://doi.org/10.1145/3102163.3102175

[21] Yiqin Lu, Chun Yu, Xin Yi, Yuanchun Shi, and Shengdong Zhao. 2017. BlindType: Eyes-Free Text Entry on Handheld Touchpad by Leveraging Thumb's Muscle Memory. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 2, Article 18 (June 2017), 24 pages. https://doi.org/10.1145/3090083

[22] I. S. Mackenzie. 2015, http://www.yorku.ca/mack/RN-TextEntrySpeed.html. A Note on Calculating Text Entry Speed.

[23] I. Scott MacKenzie and R. William Soukoreff. 2003. *Phrase Sets for Evaluating Text Entry Techniques.* Association for Computing Machinery, New York, NY, USA, 754–755. https://doi.org/10.1145/765891.765971

[24] C. Macleod, N. Ide, and R. Grishman. 2002. The American National Corpus: A Standardized Resource for American English. (2002).

[25] Anders Markussen, Mikkel Rønne Jakobsen, and Kasper Hornbæk. 2014. Vulture: A Mid-Air Word-Gesture Keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 1073–1082. https://doi.org/10.1145/2556288.2556964

[26] I. S. P. Nation and R. Waring. 1997. Vocabulary size, text coverage, and word lists. *n.schmitt m.mccarthy vocabulary description acquisition pedagogy* (1997).

[27] Tao Ni, Doug Bowman, and Chris North. 2011. AirStroke: Bringing Unistroke Text Entry to Freehand Gesture Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 2473–2476. https://doi.org/10.1145/1978942.1979303

[28] Duc-Minh Pham and Wolfgang Stuerzlinger. 2019. HawKEY: Efficient and Versatile Text Entry for Virtual Reality. In *25th ACM Symposium on Virtual Reality Software and Technology* (Parramatta, NSW, Australia) *(VRST '19)*. Association for Computing Machinery, New York, NY, USA, Article 21, 11 pages. https://doi.org/10.1145/3359996.3364265

[29] J Platt. 1999. Probabilistic outputs for SVMs and comparisons to regularized likehood methods. In *Advances in Large Margin Classifiers.* MIT Press.

[30] Daniel R. Rashid and Noah A. Smith. 2008. Relative Keyboard Input System. In *Proceedings of the 13th International Conference on Intelligent User Interfaces* (Gran Canaria, Spain) *(IUI '08)*. Association for Computing Machinery, New York, NY, USA, 397–400. https://doi.org/10.1145/1378773.1378839

[31] Shyam Reyal, Shumin Zhai, and Per Ola Kristensson. 2015. Performance and User Experience of Touchscreen and Gesture Keyboards in a Lab Setting and in the Wild. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea) *(CHI '15)*. Association for Computing Machinery, New York, NY, USA, 679–688. https://doi.org/10.1145/2702123.2702597

[32] Mark Richardson, Matt Durasoff, and Robert Wang. 2020. Decoding Surface Touch Typing from Hand-Tracking. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 686–696. https://doi.org/10.1145/3379337.3415816

[33] Mark Richardson, Matt Durasoff, and Robert Wang. 2020. *Decoding Surface Touch Typing from Hand-Tracking*. Association for Computing Machinery, New York, NY, USA, 686–696. https://doi.org/10.1145/3379337.3415816

[34] Weinan Shi, Chun Yu, Shuyi Fan, Feng Wang, Tong Wang, Xin Yi, Xiaojun Bi, and Yuanchun Shi. 2019. *VIPBoard: Improving Screen-Reader Keyboard for Visually Impaired People with Character-Level Auto Correction*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300747

[35] Weinan Shi, Chun Yu, Xin Yi, Zhen Li, and Yuanchun Shi. 2018. TOAST: Ten-Finger Eyes-Free Typing on Touchable Surfaces. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 1, Article 33 (March 2018), 23 pages. https://doi.org/10.1145/3191765

[36] R. William Soukoreff and I. Scott MacKenzie. 2003. Metrics for Text Entry Research: An Evaluation of MSD and KSPC, and a New Unified Error Metric. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Ft. Lauderdale, Florida, USA) *(CHI '03)*. Association for Computing Machinery, New York, NY, USA, 113–120. https://doi.org/10.1145/642611.642632

[37] Marco Speicher, Anna Maria Feit, Pascal Ziegler, and Antonio Krüger. 2018. *Selection-Based Text Entry in Virtual Reality*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3173574.3174221

[38] Keith Vertanen, Crystal Fletcher, Dylan Gaines, Jacob Gould, and Per Ola Kristensson. 2018. *The Impact of Word, Multiple Word, and Sentence Input on Virtual Keyboard Decoding Performance*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3174200

[39] Keith Vertanen, Dylan Gaines, Crystal Fletcher, Alex M. Stanage, Robbie Watling, and Per Ola Kristensson. 2019. *VelociWatch: Designing and Evaluating a Virtual Keyboard for the Input of Challenging Text*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3290605.3300821

[40] Keith Vertanen, Haythem Memmi, Justin Emge, Shyam Reyal, and Per Ola Kristensson. 2015. *VelociTap: Investigating Fast Mobile Text Entry Using Sentence-Based Decoding of Touchscreen Keyboard Input*. Association for Computing Machinery, New York, NY, USA, 659–668. https://doi.org/10.1145/2702123.2702135

[41] James Walker, Bochao Li, Keith Vertanen, and Scott Kuhl. 2017. *Efficient Typing on a Visually Occluded Physical Keyboard*. Association for Computing Machinery, New York, NY, USA, 5457–5461. https://doi.org/10.1145/3025453.3025783

[42] Yuntao Wang, Ao Yu, Xin Yi, Yuanwei Zhang, Ishan Chatterjee, Shwetak Patel, and Yuanchun Shi. 2021. *Facilitating Text Entry on Smartphones with QWERTY Keyboard for Users with Parkinson's Disease*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411764.3445352

[43] Daryl Weir, Henning Pohl, Simon Rogers, Keith Vertanen, and Per Ola Kristensson. 2014. Uncertain Text Entry on Mobile Devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 2307–2316. https://doi.org/10.1145/2556288.2557412

[44] Andrew D. Wilson and Maneesh Agrawala. 2006. *Text Entry Using a Dual Joystick Game Controller*. Association for Computing Machinery, New York, NY, USA, 475–478. https://doi.org/10.1145/1124772.1124844

[45] Zhican Yang, Chun Yu, Xin Yi, and Yuanchun Shi. 2019. Investigating Gesture Typing for Indirect Touch. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 3, Article 117 (Sept. 2019), 22 pages. https://doi.org/10.1145/3351275

[46] Xin Yi, Chen Wang, Xiaojun Bi, and Yuanchun Shi. 2020. *PalmBoard: Leveraging Implicit Touch Pressure in Statistical Decoding for Indirect Text Entry*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376441

[47] Xin Yi, Chun Yu, Weinan Shi, and Yuanchun Shi. 2017. Is it too small?: Investigating the performances and preferences of users when typing on tiny QWERTY keyboards. *International Journal of Human-Computer Studies* 106 (2017), 44–62. https://doi.org/10.1016/j.ijhcs.2017.05.001

[48] Xin Yi, Chun Yu, Mingrui Zhang, Sida Gao, Ke Sun, and Yuanchun Shi. 2015. ATK: Enabling Ten-Finger Freehand Typing in Air Based on 3D Hand Tracking Data. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software amp; Technology* (Charlotte, NC, USA) *(UIST '15)*. Association for Computing Machinery, New York, NY, USA, 539–548. https://doi.org/10.1145/2807442.2807504

[49] Chun Yu, Yizheng Gu, Zhican Yang, Xin Yi, Hengliang Luo, and Yuanchun Shi. 2017. *Tap, Dwell or Gesture? Exploring Head-Based Text Entry Techniques for HMDs*. Association for Computing Machinery, New York, NY, USA, 4479–4488. https://doi.org/10.1145/3025453.3025964