RingGesture: A Ring-Based Mid-Air Gesture Typing System Powered by a Deep-Learning Word Prediction Framework

Junxiao Shen, Roger Boldu, Arpit Kalla, Michael Glueck, Hemant Bhaskar Surale, and Amy Karlson

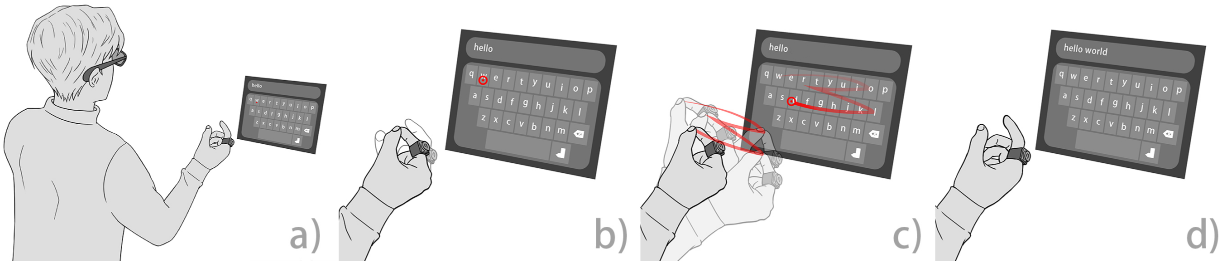


Fig. 1: *RingGesture*, a ring-based mid-air gesture typing system, enables users to input text both quickly and accurately. The process unfolds as follows: a) The process begins when the user articulates their wrist, positioning the cursor over the initial letter of the desired word. b) Then, the user performs a pinch gesture with their thumb and index finger, marking the start of the cursor's trajectory. c) Subsequently, the user gestures the word's trajectory in mid-air to complete the input by articulating their wrist. d) Upon releasing the pinch, the deep-learning word prediction framework, *Score Fusion*, predicts Top-K words, with the Top-1 word being pre-selected.

Abstract—Text entry is a critical capability for any modern computing experience, with lightweight augmented reality (AR) glasses being no exception. Designed for all-day wearability, a limitation of lightweight AR glass is the restriction to the inclusion of multiple cameras for extensive field of view in hand tracking. This constraint underscores the need for an additional input device. We propose a system to address this gap: a ring-based mid-air gesture typing technique, *RingGesture*, utilizing electrodes to mark the start and end of gesture trajectories and inertial measurement units (IMU) sensors for hand tracking. This method offers an intuitive experience similar to raycast-based mid-air gesture typing found in VR headsets, allowing for a seamless translation of hand movements into cursor navigation. To enhance both accuracy and input speed, we propose a novel deep-learning word prediction framework, *Score Fusion*, comprised of three key components: a) a word-gesture decoding model, b) a spatial spelling correction model, and c) a lightweight contextual language model. In contrast, this framework fuses the scores from the three models to predict the most likely words with higher precision. We conduct comparative and longitudinal studies to demonstrate two key findings: firstly, the overall effectiveness of *RingGesture*, which achieves an average text entry speed of 27.3 words per minute (WPM) and a peak performance of 47.9 WPM. Secondly, we highlight the superior performance of the *Score Fusion* framework, which offers a 28.2% improvement in uncorrected Character Error Rate over a conventional word prediction framework, *Naive Correction*, leading to a 55.2% improvement in text entry speed for *RingGesture*. Additionally, *RingGesture* received a System Usability Score of 83 signifying its excellent usability.

Index Terms—Text entry, augmented reality, word prediction, language models

1 Introduction

This paper focuses on one-handed text entry methods, necessitated by the occasional unavailability of both hands [13, 25, 29, 79], for lightweight augmented reality (AR) glasses in contrast to fully-fledged AR headsets (such as Apple Vision Pro [7, 15], Quest Series [48], and HoloLens Series [14, 49]). We have analyzed prior research on one-handed text entry and found that most existing methods struggle with a range of limitations. These limitations include learnability challenges, lower performance ceiling, and intricate device setup. Learnability challenges stems from an indirect correlation between hand movement and keyboard key selection, and the introduction of numerous new keyboard layouts (with new key arrangements) [23, 25–27, 30, 37, 56]. Additionally, these methods typically demonstrate low entry speeds below 15 words per minute (WPM). Some other methods also require intricate device setup procedures that require specific auxiliary devices like capacitive sensors on the fingertips [53, 75]. Our observations highlight that gesture typing with the cursor directly mapped from body movements tends to yield the highest text entry rates, as evidenced by the systems developed by Markussen et al. [45] using hand control (20.6 WPM), Yu et al. [77] using head control (19.0 WPM), and Zhao et al. [79] using arm control (16.4 WPM). This efficiency can be attributed to the simplicity and intuitiveness of direct cursor projection combined with the rapid text entry facilitated by gesture typing. Among these, Vulture [44] methods offer the highest text entry rates, and are theoretically more ergonomic and less fatiguing when compared to using head and arm. However, they utilized OptiTrack [21] for hand tracking, an outside-in tracking method that comes with inherent deployment challenges. It requires the instrumentation of the user's entire arm to track movements of the arm, wrist, and fingers, and may at times lose tracking due to marker occlusion.

Consequently, we leveraged a ring device proposed by Kienzle et al. [32] to track hand positioning. This ring can track hand movements using its built-in IMU and detect stateful pinch actions using integrated electrodes [32]. To this end, we refined our design space to concentrate on evaluating word-level gesture typing versus phrase-level gesture typing, the latter being less explored [74]. We conducted a comparative study with 32 participants. Results showed no significant text entry performance difference between the two methods, but a preference for word-level typing emerged. We also tackled the Heisenberg Effect challenge, associated with input cross modality [72] while performing mid-air pointing when with the discrete pinch actions, by introducing a customized filter-based algorithm. Even still, IMU-based tracking inevitably introduces noise and drift into the cursor's trajectory, leading to inaccurate gesture typing decoding. To guarantee swift and

- *Junxiao Shen is with Reality Labs Research, Meta and University of Bristol.*
- *Roger Boldu, Arpit Kalla, Michael Glueck, Hemant Bhaskar Surale, and Amy Karlson are with Reality Labs Research, Meta.*

Study	Interaction	Device Setup	Typing method (QWERTY)	Entry Rate (WPM)
Markussen et al. [44]	*Directly Mapped Cursor* - Hand	OptiTrack Hand Tracking	Gesture Typing	20.6
Yu et al. [76]	*Directly Mapped Cursor* - Head	Headset IMU Orientation	Gesture Typing	19.0
Zhao et al. [79]	*Directly Mapped Cursor* - Arm	Wristband IMU Tracking	Gesture Typing	16.4
Gu et al. [26]	*Direct Touch*	Ring IMU Tracking	Gesture Typing	13.8
Henderson et al. [28]	*Directly Mapped Cursor* - Finger	Smartphone Screen	Gesture Typing	13.2
Xu et al. [75]	*Direct Touch*	On-Fingertip Sensors	Touch Typing	11.9
Wang et al. [71]	*Direct Touch*	Vicon 3D Hand Tracking	Touch Typing	10.0
Chen et al. [41]	*Direct Touch*	Ring IMU Tracking	Gesture Typing	9.9
Gupta et al. [27]	*Indirectly Mapped Cursor*	Ring IMU Tracking	Gesture Typing	9.2

Table 1: Summary of previous studies on one-handed text entry methods for QWERTY keyboards, with entry rates near or above 10 WPM. *Directly Mapped Cursor* maps hand, head, arm or finger movements to cursor movement with direct projection. In contrast, *Indirectly Mapped Cursor* involves an intermediate algorithm between the physical movement and cursor movement. Studies involving *Indirectly Mapped Cursor* typically demonstrate text entry rate under 10 WPM. Note that we only report the text entry rate from novice users, as the experimental setup varies among users when measuring expert performance.

precise gesture typing, we proposed a novel deep-learning word prediction framework, *Score Fusion*, which predict user's indented words based on not only user's gestured trajectories but also keyboard spatial information and contextual information from previous conversations. This framework integrated three crucial components through fusing the probabilistic scores of word candidates from the components: 1) a word-gesture decoding model; 2) a spatial spelling correction model; and 3) a lightweight contextual language model.

We conducted the second user study involving 16 participants with two primary objectives: 1) to evaluate the effectiveness of the *RingGesture* system, and 2) to compare the proposed *Score Fusion* algorithm with a conventional word prediction baseline, *Naive Correction*, from [63]. The results indicated that, firstly, *RingGesture* achieves an average text entry rate of 27.3 WPM and a peak performance of 47.9 WPM. These results are comparable to mobile phone gesture typing performances, which are near 30 WPM [57]. Secondly, we highlight the superior performance of the *Score Fusion* framework, which offers a 55.2% improvement in text entry speed over *Naive Correction* (only achieving 17.6 WPM), due to improved word prediction from the *Score Fusion*. This underscores the significance of the *Score Fusion* framework for enabling a fast *RingGesture* system. Additionally, *RingGesture* received a System Usability Score of 83, signifying its excellent usability.

In conclusion, our contributions are threefold:

1. We propose a fast, accurate and easy-to-learn ring-based mid-air gesture typing system, *RingGesture*, which enables users to perform text entry at rates (average entry rate: 27.3 WPM, novice entry rate: 26.4, expert entry rate: 32.5 WPM) comparable to mobile phone gesture typing rate.

2. We propose a novel deep-learning word prediction framework, *Score Fusion*, which includes a word-gesture decoding model enabled by a novel data transformation process, a spatial spelling correction model enabled by a novel keyboard-layout-aware edit distance, and a novel pre-trained contextual language model while still being lightweight.

3. We conducted two studies to understand the value of our design decisions, the *RingGesture* system, and the *Score Fusion* framework: Study 1 to explore word-level gesture typing versus phrase-level gesture typing under *Directly Mapped Cursor* interaction mode; Study 2 to demonstrate the efficiency of *RingGesture*, and underscore the significant improvement of the *Score Fusion* framework over a conventional word prediction baseline *Naive Correction* (28.2% improvement in uncorrected Character Error Rate, leading to 55.2% improvement in text entry speed).

2 Related Work

Various studies of text entry methods in AR/VR suggest typical entry rates of 5 to 26 WPM [18, 19, 63, 64, 73]. It is evident that leveraging users' existing typing skills with QWERTY keyboards and simple interactions provides the best performance [17, 20, 33], while abstract mappings and complex designs hinder text entry rate [30, 38, 40, 51, 67, 78].

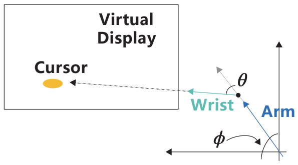


Fig. 2: 2D cursor control from arm and wrist. The cursor controlled by a ring is a direct mapping from θ, and the cursor controlled by a wristband is a direct mapping from ϕ.

2.1 One-Handed Text Entry

This section critically reviews and compares notable one-handed text entry research. Our focus is narrowed to those methods that have achieved performance rates near or surpassing 10 WPM and utilize QWERTY keyboards. Table 1 presents a comparative overview of these one-handed text entry studies.

Markussen et al. [44] proposed Vulture which is a mid-air gesture typing keyboard, and utilized OptiTrack hand tracking for *Directly Mapped Cursor* control via hand movements, achieving an entry rate of 20.6 wpm. This method exemplifies high efficiency in text entry by leveraging intuitive hand movements, closely mirroring physical interactions in the real world. The direct manipulation facilitated by this approach suggests a significant potential for enhancing user experience in AR and VR environments through natural interaction paradigms. Please note the term *Directly Mapped Cursor* does not refer to the fingertip directly touching the keyboard. Instead, *Directly Mapped Cursor* refers to the direct mapping between cursor movement and finger movement, rather than direct physical contact.

Zhao et al. [79] also employed a similar *Directly Mapped Cursor* control-based mid-air gesture typing technique, but utilized arm movements instead, incorporating a wristband with in-built IMU to track the arm's position. They achieved an entry speed of 16.4 WPM. Despite introducing the 'Speedup' method, which accelerates the cursor towards the user's gaze fixation point to enhance text entry rate, the final improved speed reached only 17.1 WPM, which is still significantly lower than the 20.6 WPM achieved by Vulture [44]. One factor to this reduced speed is that using the arm for control introduces more extensive movement, inherently leading to slower speeds. Additionally, this method results in greater fatigue as more torque is needed for the arm (α) as compared to the wrist (ϕ) because the arm's greater length, which then demands more force for the same orientation change (see Figure 2.).

Similarly, Yu et al. [76] introduced a technique based on head orientation control via a headset to navigate the cursor, with an entry rate of 19.0 wpm. This method diverges from hand-based interaction by utilizing head movements for text entry, presenting an alternative that,

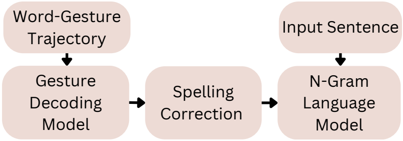


Fig. 3: The conventional word prediction framework, *Naive Correction*, operates in a sequential fashion: The predictions from the word-gesture decoding model are firstly corrected for misspellings by the edit-distance-based spelling correction model, which functions by calculating the edit distance with the word candidates in the corpus. The corrected candidates are then re-ranked by an N-Gram language model [31] based on the previously enter N-1 words.

while effective, introduces a different set of ergonomic considerations and potential user adaptation challenges.

Another study closely related to our work is RotoSwype by Gupta et al. [27], which also explores mid-air gesture typing through a ring device. However, unlike our *Directly Mapped Cursor* interaction, RotoSwype utilizes an *Indirectly Mapped Cursor* mode, implementing an indirect mapping strategy where wrist rotations correspond to cursor movements. This method, however, presents considerable challenges in user adaptation due to its non-intuitive mapping system. Gupta et al. [27] noted, 'Participants found it difficult at first to understand the mapping of angular movements to the flat pointer motion on-screen, especially for diagonal motion.' Consequently, the text entry speed for novices was recorded at 9.2 WPM, underscoring the inherent learning curve associated with this innovative interaction technique.

Additionally, Henderson et al. [28] utilized smartphones for AR cursor interaction, achieving a 13.2 WPM rate, suggesting mobile tech integration can ease text entry access. Yet, this contradicts the goal of AR glasses reducing mobile phone dependence. Similarly, wearable tech for text entry was explored by Xu et al. [75], Gu et al. [26] and Chen et al [41], with fingertip and ring-based sensors reaching 11.9 WPM, 13.8 WPM and 9.9 WPM, respectively. These methods present innovative, albeit less adaptable AR interaction solutions.

The top 3 text entry methods in Table 1 are based on the *Directly Mapped Cursor*-based mid-air gesture typing approach. However, each method has its drawbacks. Vulture [45] relies on OptiTrack for hand tracking which is a 'Wizard-of-Oz' technique, Yu et al. [77]'s system uses head movements for cursor control, which introduces ergonomic challenges, and Zhao et al. [79]'s system can cause arm fatigue due to the involvement of whole arm movements. Our system, *RingGesture*, improves upon Vulture [45]'s mid-air gesture typing approach by using a ring equipped with Inertial Measurement Units (IMUs), allowing for cursor control through hand movements alone. Recognizing that IMUs may lead to noise and drift, resulting in inaccurate cursor control and subsequently inaccurate word-gesture decoding, we proposed a deep-learning word prediction framework, *Score Fusion*, to enhance word prediction accuracy, thereby increasing the rate of text entry.

2.2 Word Prediction in Text Entry

Word prediction in text entry systems involves word decoding, spelling correction, and language modeling. The decoding model interprets the user's input patterns into raw predictions, forming a sequence of characters, but it may introduce spelling errors. The spelling correction model rectifies these inaccuracies, while language modeling further enhances word prediction by considering the context provided by previous words. Shen et al. [63] connect these components sequentially, as illustrated in Figure 3. The word prediction system, referred to as *Naive Correction* in this paper, enhances accuracy through a sequential process involving edit-distance-based spelling correction and N-Gram language model re-ranking. Despite this system being the state-of-the-art word prediction framework for gesture typing, it has several drawbacks: it only considers limited contextual scope, ignoring global probabilities across the entire dataset. Additionally, it lacks deep semantic understanding, leading to potential inaccuracies in certain contexts. We propose a novel word prediction system that integrates correction and language modeling through a probabilistic approach, allowing the system to consider global probabilities rather than processing them sequentially. Additionally, our approach enhances spelling correction by incorporating spatial information from the keyboard and advances the N-Gram language model by utilizing much longer contexts, while maintaining a lightweight design.

General spelling correction encompasses a wide array of contexts, including document editing and processing digital texts in databases or on online platforms. Its primary objective is to identify and correct errors throughout entire sequences of words [8, 12, 54]. In contrast, spelling correction in text entry systems on mobile devices is specifically tailored for real-time user inputs. It focuses on correcting errors in single words as they are typed, where the errors not only come from users but also from text decoders in a probabilistic text entry system [20]. Spelling correction in text entry systems commonly relies on the edit distance method, prized for its ease of integration with custom word corpora and modifications to achieve microsecond latency [36, 50]. However, its accuracy for QWERTY-based text entry systems is limited [8]. One of the major reasons is that it fails to consider the spatial information of the keyboard. This oversight leads to reduced correction accuracy by offering phonetically or orthographically similar but irrelevant corrections, and to inefficient error prediction due to the inability to accurately anticipate mistyped words based on common finger movements and miskeying patterns, thus diminishing its overall effectiveness. Our paper addresses this gap by proposing a novel spatial edit distance that incorporates the spatial information of a gesture typing keyboard. Additionally, we transformed the spatial edit distance into a probabilistic-based measure, allowing seamless integration with a probabilistic word decoder.

Language modeling represents another important approach for enhancing word error correction. N-gram language models are commonly utilized for meeting latency demands owing to their ease of implementation and explainability [31, 52]. With the advancements in deep learning technologies, models based on deep learning have been progressively incorporated into language modeling [9, 22, 68]. However, there has been limited research on contextual language modeling for text entry systems. Contextual information is a crucial element in text entry, which includes elements such as conversation history and other context tags like places, times, and hobbies of the users, etc. Shen et al. [65] proposed for the first time a contextual language model based on GPT-2 for Augmentative and Alternative Communication (AAC) use cases. However, the Generative Pre-trained Transformer-2 (GPT-2) [55] model has significant latency when operated on a mobile device, while this latency is acceptable for the AAC use case in Shen et al. [65]. We propose a novel method that transforms contextual language modeling through pre-training with Long Short Term Memory (LSTM) models instead of pre-training with transformers. This results in a lightweight model architecture, with the final contextual language model being only 7 megabytes (MB) in size and capable of running in real-time on a mobile device.

3 User Study 1: Word-Level versus Phrase-Level Gesture Typing

We began by investigating whether the simple act of removing the delimitation requirements between words could accelerate mid-air gesture input under the *Directly Mapped Cursor* interaction mode. As we are proposing a novel and comprehensive text entry system that ranges from interaction design to backend architecture design, the choice between word-level and phrase-level typing is a fundamental component of the interaction design. Therefore, it is important to conduct a study to explore this aspect.

While Xu et al. [74] have conducted studies comparing phrase-level to word-level gesture typing on smartphones, their approach relies on *Direct Touch*. This differs from our *Directly Mapped Cursor* mode. Consequently, Xu et al.'s findings [74] may not be directly applicable to our context. Therefore, we conducted our own study using a touchpad to simulate *Directly Mapped Cursor* interaction. Controlling a cursor by swiping on a touchpad directly translates fingertip movements into cursor movements on the screen. This method is particularly advanta-

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2024.3456179

geous because it provides an accurate representation of our swipe path, serving as the ground truth. In contrast, most other methods, such as those based on inertial measurement units (IMUs) or camera tracking, introduce noise, detracting from the fidelity of tracking. Thus, using a touchpad enables us to simulate perfect tracking, which is crucial for the precision required in our comparisons of gesture typing at the phrase level and at the word level within this *Directly Mapped Cursor* interaction mode. The following are the details of the study:

1. **Participants**: We recruited 32 volunteers as participants through an internal mailing list, who had an average age of 33 (range 18-64, standard deviation 10.91). The group comprised 18 males, 13 females, and 1 participant who chose not to disclose their gender. 27 participants are right-handed, and 5 participants are left-handed.
2. **Apparaturs**: Participants controlled a cursor using a Sensel Touchpad [2] placed on the table in front of a monitor, with the cursor displayed on a virtual keyboard on a monitor. The monitor was connected to a Lenovo PC (ThinkStation) equipped with an Intel Xeon processor.
3. **Phrase Set**: The phrase set used was collected from two sources: the Enron Mobile Corpus [34] and the MacKenzie phrase set [43]. This combined phrase corpus encompassed a total of 42,612 unique phrases. Each phrase in this corpus exhibited an average length of 5.3 words, with a minimum length of 2 words and a maximum length of 7 words.
4. **Procedure**: Participants in the study were directed to execute tasks under two distinct conditions: gesture typing at the word level and at the phrase level. To ensure impartiality, these conditions were counterbalanced. Each condition consisted of 40 phrases, selected uniformly from the aforementioned phrase set. Under word-level gesture typing, participants are instructed to delimit after swiping for each word by lifting up the finger from the touchpad. In contrast, under phrase-level gesture typing, participants are instructed to delimit only when the entire phrase is completed. We implemented a pseudo-decoder, based on the model proposed by Shen et al. [62], that simulates an ideal decoder. This decoder predicts correct words as long as at least 70% of the gesture trajectory passes through the designated tolerance region for each character in the swiped word or phrase. To simulate the decoder's capability to manage ambiguous inputs effectively, the key region is defined to be four times larger than the actual size of the keys. Before starting each condition, participants were allowed to practice with 5 phrases. During the condition, participants could rest for up to 2 minutes after every 10 phrases. To advance to the next phrase, participants press the `Space Bar` button on the keyboard. At the end of the study, participants were invited to fill out a post-study questionnaire. This included a Likert scale rating on various aspects for both word-level and phrase-level gesture typing: 1) *Easy to Type*, 2) *Easy to Learn*, 3) *Fast to Type*, 4) *Prediction is Accurate*, 5) *Hand Feels Fatigued*, 6) *Eyes Feel Fatigued*. Additionally, participants were asked the following open-ended question: 'Between word-level typing and phrase-level typing, did you find one method superior to the other? If so, which one and why?' Each condition for one participant took around 30 minutes to complete.
5. **Evaluation Measures:** We report the results of the studies using the following metrics:

- Words Per Minute (WPM) is represented mathematically as:

$$WPM = \frac{\text{Total Words Typed}}{\text{Time in Minutes}}$$

- Character Error Rate (CER) can be quantified using the formula:

$$\frac{\text{Minimum Number of Insertions, Deletions, and Substitutions}}{\text{Length of Stimulus Text}}$$

- Uncorrected Character Error Rate (Uncorrected CER) and Corrected Character Error Rate (Corrected CER) are defined for the predicted text output before and after correction interventions, respectively, with corrections including word deletions and re-entries, as follows:

$$\text{Uncorrected CER} = \frac{\text{Number of Errors in Initial Prediction}}{\text{Length of Stimulus Text}}$$

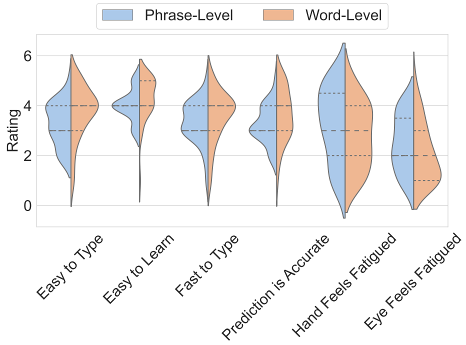


Fig. 4: Violin plots of answers to subjective rating questions scored on 5-point Likert scales. Violin plots are modified box plots that add estimated kernel density plots to the summary statistics displayed by box plots. The 5-point Likert scales ranged from 1 (strongly disagree) to 5 (strongly agree).

$$\text{Corrected CER} = \frac{\text{Number of Errors After Corrections}}{\text{Length of Stimulus Text}}$$

Initially, our analysis revealed no significant difference in text entry speed between word-level and phrase-level gesture typing, which were measured at 24.7 WPM and 25.5 WPM, respectively, accompanied by uncorrected character error rates of 7.8% and 9.2%. To further explore the impact of typing conditions on text entry speed and accuracy, we performed an ANOVA analysis. The findings from this analysis indicated that there were no significant differences in both text entry speed (F= 2.14, p = 0.13) and accuracy (F = 1.89, p = 0.24) across the two conditions. This underscores the similarity in performance between word-level and phrase-level gesture typing in terms of both speed and precision.

Then we analyze the subjective feedback collected from the post-study questionnaire.

- *Ratings on Six Aspects*: Figure 4 illustrates a comparison between phrase-level and word-level input methods across various aspects. Observing the shapes and distributions, Phrase-level input generally have a broader spread in ratings for ease of typing and learning, which indicate a more varied user experience. Word-level input, on the other hand, score consistently higher for being perceived as fast to type and having accurate predictions, with ratings clustered around higher medians, suggesting users may find it more efficient in these respects. When it comes to fatigue, both hand and eye fatigue are reportedly lower with word-level input, as reflected by the denser concentration of lower ratings. Overall, while there's some overlap in user responses, the data suggests a preference for word-level input in terms of speed, accuracy, and reduced fatigue.
- *Preferences*: We further analyze the responses of participants regarding their preference between word-level typing and phrase-level typing. There were 19 participants who explicitly favored word-level typing. The reasons include: 1) **Familiarity** (6 mentions): Word-level typing is more similar to traditional typing methods, where each word is separated by 'hitting the spacebar'. 2) **Cognitive Load** (5 mentions): Some users mentioned that their brains think in terms of words rather than phrases, making word-level typing more natural. 3) **Less Fatigue** (3 mentions): Some respondents indicated that word-level typing is less tiring because it doesn't require the user to hold and drag for long periods. 4) **Feedback** (2 mentions): Users get immediate feedback after typing each word, which helps them correct errors on the go. 5) **Coordination** (3 mentions): A few respondents found it easier to coordinate their eyes and hands while typing at the word level. However, 8 participants preferred phrase-level typing. The reasons include: 1) **Efficiency** (3 mentions): Phrase-level typing allows users to type longer sentences more quickly, as it eliminates the need to delimitate between words. 2) **Convenience** (3 mentions): Users found it convenient that phrase-

level typing automatically segmented the words on their behalf. 3) **Accuracy** (2 mentions): Some users found phrase-level typing more accurate. Finally, 5 respondents did not express a clear preference for either word-level or phrase-level typing.

Given that phrase-level gesture typing did not yield any notable enhancements over word-level gesture typing as indicated, coupled with the fact that a larger user base preferred the latter, we opted for word-level gesture typing.

4 RingGesture

Study 1 investigated the interaction design of the *RingGesture* system. This section provides a comprehensive overview of the backend architecture design of the *RingGesture* system. It includes the algorithm designed to overcome the Heisenberg Effect challenges associated with input cross modality while performing mid-air pointing when with the discrete pinch actions [72], and our novel deep-learning word prediction framework, *Score Fusion*.

4.1 Ring-Based Mid-Air Pointing and Selection

We created a ring device with a reference design from ElectroRing proposed by Kienzle et al. [32]. This ring detects *touch* and *release* events of a pinch gesture by monitoring changes in an electrical signal. Furthermore, the ring uses an IMU for 2D cursor tracking by transforming accelerometer and gyroscope data into quaternions, converting these into polar coordinates, and then mapping them to Cartesian coordinates. The gain parameter for the control display is set to 1.8.

While we effectively utilize the ElectroRing design [32] for pinch detection and IMU-based 2D cursor tracking, our system encountered a challenge in the context of mid-air gesture typing: Heisenberg Effect associated with input modality crosstalk [72]. This issue arises when a discrete input like a pinch inadvertently alters the virtual cursor's position, resulting in an inaccurate selection point during mid-air pointing and selection interactions.

Therefore, to counteract the abrupt displacement introduced by pinch actions, we suggest a filter-based strategy. This approach dynamically determines the filtering level within an exponential smoothing filter by resolving the subsequent optimization problem:

$$\alpha_0 = \arg\min_{\alpha} \left\{ \lambda\sigma \underbrace{\sqrt{\frac{\alpha}{2-\alpha}}}_{\text{Noise rejection}} + (1-\lambda) \underbrace{\frac{(1-\alpha)\Delta}{\alpha}}_{\text{Tracking error}} \right\},$$

In this equation, σ signifies the level of sensing noise, Δ provides an estimate of the signal's velocity, and λ serves as a parameter that balances noise rejection (the left term in the minimization above) and infinite horizon tracking error in response to an input ramp (the right term in the minimization above). The λ parameter is preset to 0.75 with preliminary experiments.

4.2 Score Fusion

To mitigate the issues posed by input signal noise, such as hand jitter and IMU drift that lead to inaccuracies in word-gesture decoding, we have developed a deep-learning framework for word prediction, *Score Fusion*. This framework consists of three distinct components that compute the logarithmic probability of a word within a corpus based on a given word-gesture trajectory, as illustrated by Figure 5. These individual scores are then consolidated to provide a composite score for the words across the corpus. Subsequently, we reorder these scores to present the highest-ranked words as the suggested options.

4.2.1 Word-Gesture Decoding Model

Deep-learning-based decoders [6, 63] have demonstrated significant advancements over traditional shape-matching-based decoders [35]. Motivated by these advancements, we aimed to train a deep-learning-based decoder tailored to our specific use case. However, deep learning models necessitate a substantial volume of training data to avoid overfitting. To address this challenge, we utilized a large-scale, publicly available gesture typing dataset, How We Swipe Dataset [39]. Given that this dataset was collected from a different keyboard layout, we proposed a novel method to transform the trajectories to match our customized keyboard layout as follows:

- **Training Dataset:** How We Swipe Dataset [42] is a large-scale gesture typing dataset that was collected via a web-based custom virtual keyboard, involving 1,338 users who submitted 11,318 unique English words. However, the dataset used a keyboard layout customized to a mobile phone. Our preliminary analysis revealed that merely normalizing the trajectory data by dividing it by the keyboard's width and height leads to training data that is not viable for effective use. Therefore, we employ a novel transformation method which is a piecewise affine transformation to transform the dataset to our keyboard layout.
- **Piecewise Affine Transformation:** This transformation process, illustrated by Figure 6, adjusts the swipe path onto a standard keyboard layout (known as the model layout) before it is processed by the model. This transformation relies on two corresponding sets of anchor points, one set on the input layout and another on the model layout. In our approach, these anchor points include 1) the central points of all keys that have the same labels across layouts, and 2) an additional six anchor points surrounding the keyboard, each positioned three key distances from the English-letter region's border (denoted by 'TL', 'ML', 'BL', etc. in Figure 6). These additional points are crucial for adjusting for swipe paths that extend beyond the keyboard border. Once we've established the anchor points, the area they cover is partitioned into a grid. Within each grid subregion, which is enclosed by four nearby anchor points, spatial coordinates are modified in a manner akin to perspective transformation seen in photo editing. This transformation method improves the test accuracy of a word-gesture decoding model, on the word-level gesturing

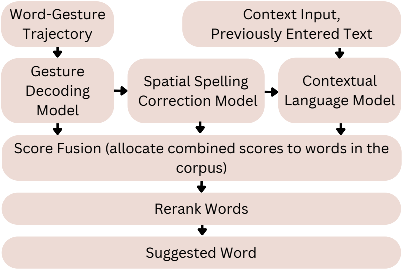

Word-Gesture Trajectory
Context Input, Previously Entered Text
Gesture Decoding Model
Spatial Spelling Correction Model
Contextual Language Model
Score Fusion (allocate combined scores to words in the corpus)
Rerank Words
Suggested Word

Fig. 5: Our novel deep-learning word prediction framework, *Score Fusion*, operates in an integrated fusion process: This fusion process evaluates each word suggestion by considering its initial decoding score, its likelihood of being a spatial spelling correction, and its contextual relevance. The resulting blended score aims to ensure that the final suggestions are derived from an accurate word-gesture decoding model while also being enhanced for typographical precision, keyboard-layout-awareness, and contextual relevance.

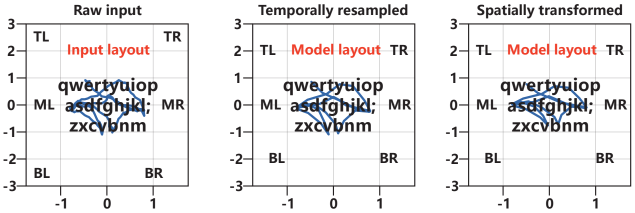


Fig. 6: The novel process of converting word-gesture trajectory data from one keyboard layout to another. It demonstrates an example of a trajectory (for the word 'available') that undergoes both temporal and spatial transformations. The keyboard layouts vary in terms of key spacing, bottom row shifts, and the presence or absence of the apostrophe key.

dataset collected from Study 1, by 76% compared to when using training data processed through the previously mentioned normalization method.

- **Training Model:** We build our gesture decoding model using the Attention-Enhanced Bi-directional LSTM with CTC loss (AE-BLSTM-CTC) architecture proposed by Shen et al. [63]. Then we trained our model on the previously transformed dataset. We used the same training hyperparameters as in Shen et al. [63]. The training hyperparameters for the word-gesture decoding model is directly adopted from Shen et al. [63].

4.2.2 Spatial Spelling Correction Model

The word-gesture decoding model is a character-level model that predicts the probability of classes (26 characters plus the blank class) at each timestep of the input trajectory sequence. As such, the prediction may contain spelling errors, caused by noise from the model as well as from the user's input, thus necessitating an auto-correction model to correct the misspelled words. Therefore, we propose a probabilistic edit distance that incorporates keyboard spatial information to address these shortcomings. The computation of this spatial-aware probabilistic edit distance involves three steps:

1. **Calculation of the insertion probability** $P_{insert}(i)$. This probability measures the likelihood of an insertion at the i-th position of the input string. If the insertion occurs at the end of the input, the probability is $\log(1)$, otherwise, it is equivalent to the omission probability P_{omit}. P_{omit} represents the probability that a user omits a character when typing. This is modeled as a logarithmic probability with a base value of 0.06, yielding a logarithmic probability of -1.22.
2. **Calculation of the deletion probability** $P_{delete}(i)$. This probability measures the likelihood of a deletion at the i-th position of the intent string and is equivalent to the stray probability P_{stray}. P_{stray} represents the probability that a user accidentally adds an extra character. This is also modeled as a logarithmic probability with a base value of 0.06, yielding a logarithmic probability of -1.22.
3. **Calculation of the substitution probability** $P_{sub}(i, j)$. This probability measures the likelihood of a substitution at the i-th position of the intent string and the j-th position of the input string. If the i-th character of the intent string is equal to the j-th character of the input string, the substitution probability is $\log(1)$. Otherwise, the substitution probability is equivalent to the substitution probability P_{sub}. P_{sub} represents the probability that a user substitutes one character for another. This measure differentiates between adjacent keys and non-adjacent keys on the keyboard. For adjacent keys (e.g., 'q' and 'w'), the base probability is 0.17, yielding a logarithmic probability of -0.77. For non-adjacent keys, the base probability is 0.01, yielding a logarithmic probability of -2.

We obtain the base probability through an estimation of character error rates on publicly available experiment data from a mid-air gesture-typing keyboard [63]. The spatial-aware probabilistic edit distance is then calculated as a composite function of these probabilities:

$$P_{ED} = P_{omit}^{n_{ins}} \cdot P_{stray}^{n_{del}} \cdot P_{sub}(s_1) \cdot P_{sub}(s_2)$$

Taking the logarithm of both sides, we get:

$$\begin{aligned}\log(P_{ED}) = {} & n_{ins} \cdot \log(P_{omit}) + n_{del} \cdot \log(P_{stray}) \\ & + \log(P_{sub}(s_1)) + \log(P_{sub}(s_2))\end{aligned}$$

where n_{ins} and n_{del} denote the number of insertions and deletions, respectively, and s_1 and s_2 denote the substitutions.

4.2.3 Contextual Language Model

We employed a bi-directional LSTM (Bi-LSTM) [24, 58, 60] model to generate predictions of subsequent word based on previous tokens. The input and output of the model are illustrated in Figure 7. Our model structure comprises four key components: an embedding layer, a representation layer, a decoder layer, and a contextual encoder.

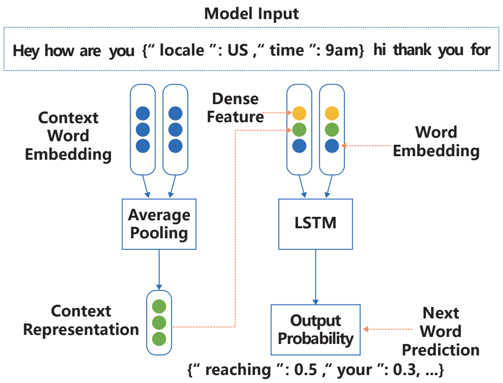


Fig. 7: Contextual LSTM-based language model structure, with model input including a) previous conversation history (eg. 'Hey how are you'), b) context tags (eg. 'location': US, 'time':9am), c) previously entered text (eg. 'hi thank you'). This model predicts the next word based on probabilities. It assigns a probability to each word in the corpus; for example, the probability of 'reaching' is 0.5, 'your' is 0.3, and there are smaller probabilities for many other words.

Language Model Architecture	Perplexity
Baseline Bi-LSTM LM	70.91
Contextual Bi-LSTM LM	42.82
Pre-trained Contextual Bi-LSTM LM	37.16

Table 2: Experiments for contextual language model. Perplexity is a measure of how well a probability model predicts a sample, with lower values indicating better predictive accuracy.

1. Embedding Layer: We utilize a Convolutional Neural Network (CNN) [70] for our embedding layer with an embedding dimension of 38, 100 kernels of size 3. We opted against using dilation and weight normalization [59] in the CNN to retain the original characteristics of the input data. These hyperparameters were determined through a grid-search-based hyperparameter optimization process.
2. Representation Layer: For this layer, we employ a BiLSTM. Our BiLSTM has two layers and a dimension of 2048. We incorporate a dropout of 0.001 to prevent overfitting.
3. Decoder Layer: The final layer of our model is an MLP decoder. This layer transforms the high-level features learned by the previous layers into the final output. Our MLP has a hidden dimension of 1024 and leverages ReLU as the activation function. A dropout rate of approximately 0.00092 is used to further mitigate overfitting.
4. Contextual Encoder: To efficiently incorporate long context information, we introduced a contextual encoder. This encoder first performs average pooling on the context word embeddings and then concatenates these with the word embeddings. The contextual encoder is co-trained with the remaining language model (LM) modules.

We employ perplexity, as defined by [61], to assess the performance of language models. This metric is calculated as the exponentiation of the average negative log-likelihood of the test set words, normalized by the number of words. As evident from Table 2, integrating contextual information significantly enhances model perplexity. Furthermore, pre-trained language models, as highlighted by [16] and [69], demonstrate exceptional utility in scenarios where training data is significantly limited. Our approach involved initially pre-training the language model using diverse sources such as public comments and posts [1, 3–5], followed by fine-tuning on the training dataset outlined by [65]. As demonstrated in Table 2, the pre-trained contextual language model substantially outperforms basic models, thereby validating the effectiveness of pre-training coupled with subsequent fine-tuning. After quantization, the final exported contextual language model is only 7MB, enabling real-time execution on contemporary mobile phone

Algorithm 1 *Score Fusion*

```
Require: Trajectory, SwipeCorrectionCoeff, LmCoeff, NumSuggestions, vocab, context
Ensure: Sorted suggestions
$raw\_decodings \leftarrow WordGesture\_Decoder(Trajectory)$
Initialize suggestions as an empty dictionary
for each $raw\_word$, $raw\_score$ in $raw\_decodings$ do
    $text\_probabilities \leftarrow Context\_Language\_Model(context)$
    $typo\_probabilities \leftarrow Spatial\_Spelling\_Correction(raw\_word)$
    for $i$ in 0 to $len(typo\_probabilities) - 1$ do
        $correction \leftarrow vocab[i]$
        $index \leftarrow find(correction, vocab)$
        $lm\_score \leftarrow text\_probabilities[index]$
        $blended\_score \leftarrow (1 - SwipeCorrectionCoeff - LmCoeff) * raw\_score + LmCoeff * lm\_score + SwipeCorrectionCoeff * typo\_probabilities[i]$
        if $correction$ in $suggestions$ then
            $blended\_score \leftarrow max(suggestions[correction], blended\_score)$
        $suggestions[correction] \leftarrow blended\_score$
$sorted\_suggestions \leftarrow sort(suggestions, byValue, descending)$
if $NumSuggestions < len(sorted\_suggestions)$ then
    $sorted\_suggestions \leftarrow sorted\_suggestions[0 : NumSuggestions]$
return $sorted\_suggestions$
```

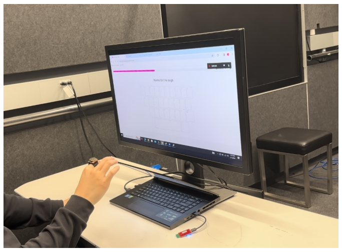


Fig. 8: Experiment Setup for Study 2: A participant is seated in front of a monitor, wearing a ring on their index finger. They can comfortably rest their arms on the armrests of the chair and freely move their wrist to perform mid-air gesture typing.

processors.

4.2.4 Implementation Details

We use PyText [47] to implement the contextual language model. We employ the Adam optimizer to train our model, with a learning rate of 0.001, epsilon of 1e-8, and weight decay of 0.00001. The model is trained for a total of 25 epochs, with an early stopping criterion set after 5 epochs. We accumulate gradients over 4 batches before updating the model parameters, and each epoch consists of 4700 such batches. Our model is designed to leverage distributed training with a world size of 8, effectively utilizing multiple GPUs to speed up the training process. The *Score Fusion* framework combines the log probabilities from the three models to assign a score to each word, creating a list of suggestions. More specific details are illustrated by Algorithm 1.

5 User Study 2: Longitudinal Evaluation of RingGesture

Our second study was driven by two primary objectives. First, we compared two text entry conditions: one that used *Score Fusion* when entering phrases and another that used *Naive Correction*. Second, we evaluated the potential text-entry performance of *RingGesture*, which involved assessing both the users' initial proficiency and their progress over time. Through the analysis of the learning curve, we aimed to gain a deeper understanding of the system's usability and the time required for users to reach proficiency.

1. **Participants**: We recruited 16 volunteers as participants through an internal mailing list. The details of their demographics are listed below. 13 participants are right-handed, and 3 participants are left-handed. The ages of the participants ranged from 21 to 49.5 years, with an average age of 37.72 years. The standard deviation in the age distribution was 10.30 years. In terms of gender, 6 participants (37.5%) identified as male, 9 participants (56.25%) identified as female, and 1 participant (6.25%) preferred not to disclose their gender. Among the participants, 12 (75%) reported they would be wearing the device on their right arm, while the remaining 4 (25%) would wear it on their left arm. When asked about the frequency of using gesture typing, 5 participants (31.25%) reported always using it (at least once a day), 7 participants (43.75%) sometimes (at least once a week), 2 participants (25%) seldom (less than once a month), and the remaining 2 participants never used it.

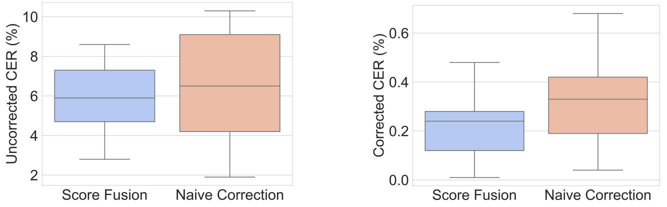
2. **Phrase Set**: The studies utilize the phrase set derived from the ConvAI2 challenge dataset [11], which consists of a total of 42,612 unique phrases. Each unique phrase is accompanied by two additional elements: context tags, which include speaker persona, and conversation history. For example, one unique phrase in the dataset might be 'I read books in the afternoon.' This phrase would be accompanied by context tags such as 'love reading' for speaker persona, and a conversation history element like 'How are you?'. The dataset's unique characteristics, namely its conversational basis and inclusion of persona information, make it an ideal tool for evaluating the contextual capabilities of intelligent text entry systems. In the study, the contextual information, which is pre-defined with the stimulus phrase, is fed automatically as additional input to the word error correction frameworks. Allowing participants to freely enter text and use their own conversational language necessitates a large-scale, in-the-wild study to ensure a fair comparison between the two conditions. However, our current implementation of the ring device does not support running such a large-scale study in a natural setting for the comparison of these two conditions. By pre-establishing the conversational context, our study offers valuable insights into the realistic text entry rates of upcoming systems that will account for the historical context of use.

3. **Apparatus**: Participants controlled the cursor using the ring, and the cursor was displayed on a virtual keyboard on a monitor. The delimitation is performed when detecting a pinch. The monitor was connected to a Lenovo PC (ThinkStation) equipped with an Intel Xeon processor. We chose a computer over AR glasses to allow demonstrators and participants to view the same screen. This setup enabled participants to pose questions and receive immediate feedback about the on-screen scenes, and provided a more effective platform for demonstrators to explain the swiping and delimitation techniques during the practice stage.

4. **Baseline**: *Naive Correction* acts as the baseline in this study. It is a state-of-the-art word prediction framework for gesture typing that was used in Shen et al. [63], as illustrated by Figure 3.

5. **Procedure**: Each participant was seated before a computer screen displaying a keyboard interface as illustrated in Figure 8. Participants initially practiced with a set of 5 phrases, during which they were encouraged to ask any questions. Subsequently, they completed four sessions, each consisting of four blocks. Each block contained two conditions: with *Score Fusion* and with *Naive Correction*. In each condition, participants input 10 phrases, resulting in a total of 20 phrases per block. The conditions were counterbalanced, and the sessions were scheduled across two weeks on separate days. In each of the four sessions, participants were given time to familiarize themselves with the functionality of *RingGesture* before beginning their typing. During each session, participants were instructed to type the phrases 'as swiftly and accurately as possible, as if typing an email to a colleague.' A break of up to three minutes was allowed between each block of 20 phrases. Similar to Study 1, we also use `Space Bar` to proceed to the next phrase. The average duration of each session was approximately 30 minutes. At the end of the study, participants were invited to complete a post-study questionnaire similar to Study 1. This included a Likert scale rating on the same six aspects in study 1: *Ease of Typing/Effort*, *Ease of Learning*, *Perceived Speed*, *Perceived Accuracy*, *Hand Fatigue* and *Eye Fatigue/Attention Switch*. Additionally, participants were requested to complete a standardized System Usability Score form [10].

This article has been accepted for publication in IEEE Transactions on Visualization and Computer Graphics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2024.3456179



(a) Uncorrected Character Error Rate

(b) Corrected Character Error Rate

Fig. 9: Box plots depicting mean, median and quartiles of the participants' performance including corrected and uncorrected character error rates under the two conditions.

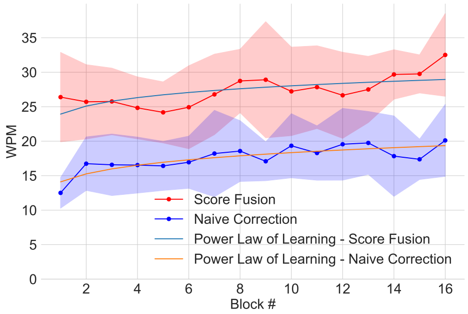


Fig. 10: Comparative analysis of *Score Fusion* and *Naive Correction* conditions over 16 longitudinal blocks.

Finally, they were posed the open-ended question: 'Did wearing the ring influence your swipe behavior or task performance?'

5.1 Performance Analysis

5.1.1 Error Rate Analysis

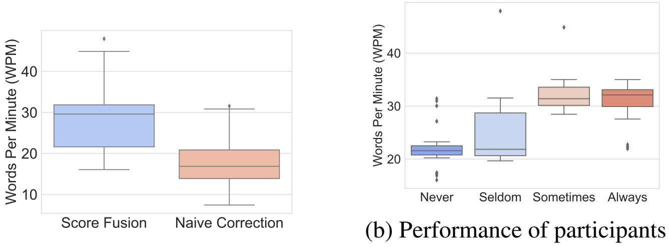
Figure 9a and Figure 9b shows that *Score Fusion* also has a lower and more consistent uncorrected and corrected CER compared to *Naive Correction*, indicating it's a more effective method for error prevention before correction is even applied and thereby reducing errors. Overall, *Score Fusion* outperforms *Naive Correction* by 28.2% in uncorrected CER, and 29.0% in corrected CER.

5.1.2 Entry Rate Analysis

Figure 10 illustrates the performance of the *RingGesture* system when coupled with the *Score Fusion* framework. The system achieved an average entry rate of 27.3 words per minute (WPM), starting at 26.4 WPM in the initial block and rising to 32.5 WPM in the final block. This progression showcases the improvement from novice to expert levels of performance.

Figure 11a indicates that *Score Fusion* shows a higher median entry rate and a tighter interquartile range compared to *Naive Correction*, suggesting it enables faster text entry and provides more consistent performance across different blocks or users. Figure 11b demonstrates a general trend of increasing entry rates with more frequent use. The spread of entry rates (as shown by the interquartile ranges and outliers) also seems to generally decrease with more experience, indicating that users become not only faster but also more consistent with practice.

Additionally, we use Repeated Measures ANOVA (RM-ANOVA) [46] and the power law of learning [66] to analyze gesture typing performance under two conditions: *Score Fusion* and *Naive Correction*. RM-ANOVA is chosen for its efficacy in handling within-subject variance across repeated observations, allowing us to assess the impact of the two conditions over time and the consistency of participant performance. We also use the power law of learning to get insight into improvement rates and learning dynamics, improving our understanding of how participants adapt to each condition.



(a) Comparison between the two conditions: *Score Fusion* and *Naive Correction*.

(b) Performance of participants with different gesture typing proficiency under *Score Fusion* condition.

Fig. 11: Box plots depicting mean, median, and quartiles of the text entry rate performance in Study 2 showing overall entry rates (a) and by self-assessed experience level (b).

Factor	F Value	Num DF	Den DF	Pr > F
Condition	30.276	1.000	7.000	0.001
Block	4.108	15.000	105.000	0.000
Condition:Block	1.333	15.000	105.000	0.196

Table 3: ANOVA-RM results for assessing the impact of condition and block on the text entry rate across repeated measures.

- *Between Condition*: The RM-ANOVA analysis revealed a significant main effect of condition (F=30.276, p=0.001), indicating a substantial difference in typing performance between the *Score Fusion* and *Naive Correction* conditions. This statistical significance is further indicated by the performance metrics, where *Score Fusion* exhibited a mean text entry rate of 27.3 WPM, outperforming *Naive Correction*'s mean of 17.6 WPM. This difference translates to a notable 55.2% improvement in favor of*Score Fusion*, emphasizing not just a statistical but a practical superiority in typing efficiency.
- *Between Blocks*: Additionally, the RM-ANOVA showed a significant effect for blocks (F=4.108, p=0.000), suggesting variability in typing performance over time which could be caused by learning effects, yet no significant interaction between condition and block (F=1.333, p=0.196) was observed, indicating that the performance advantage of *Score Fusion* is consistent across different time points. This consistency, backed by *Score Fusion*'s superior statistics (with a standard deviation of 5.9, minimum of 16.1, and maximum of 47.9) compared to *Naive Correction*'s (standard deviation of 4.7, minimum of 7.4, and maximum of 31.5), highlights how different word prediction frameworks in gesture typing not only influence overall performance but also ensure sustained efficiency across blocks.
- *Power Law of Learning*: As there is a significant variability in typing performance over time, we analyze the learning dynamics through the power law of learning for *Score Fusion* and *Naive Correction* conditions. Figure 10 also plots the power law of learning for the two conditions. We find distinct patterns in participants' improvement rates. The R-squared values are 0.415 for *Score Fusion*, and 0.697 for *Naive Correction*, suggesting that learning under *Naive Correction* is slightly more predictable over time than under *Score Fusion*. Initial performance levels, indicated by $a = 23.9$ for *Score Fusion* and $a = 14.1$ for *Naive Correction*, show that participants start off better with *Score Fusion*. However, the rate of learning, represented by b, is faster in *Naive Correction* ($b = 0.115$) than in *Score Fusion* ($b = 0.069$), despite the higher initial performance in the latter. The more predictable learning effect, and the higher learning rate might be caused by participants' adaptation to the *Naive Correction* framework, whereas *Score Fusion* offers accurate predictions, eliminating the need for user adaptation.

5.2 Subjective Ratings & Feedback

- **System Usability Score**: The overall System Usability Scale (SUS) score for the gesture typing system is 83. This indicates that users

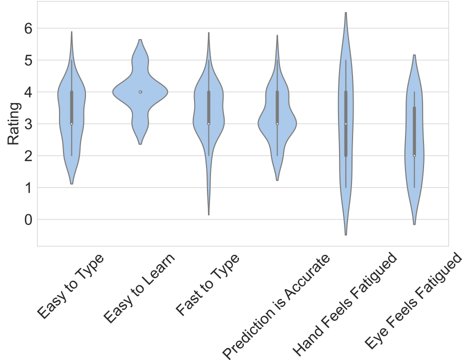


Fig. 12: Violin plots of answers to subjective rating questions scored on 5-point Likert scales. The 5-point Likert scales ranged from 1 (strongly disagree) to 5 (strongly agree).

found the system to be highly usable. A score above 68 is considered above average, and anything above 80 is an indication of excellent usability. In this context, a score of 83 suggests that the gesture typing system was not only easy to use but also met or exceeded the expectations of most users in terms of efficiency and satisfaction. This score aligns with the ratings of the six aspects, suggesting that the gesture typing method is easy to learn and efficient to use.

- **Ratings of Six Aspects**: The post-study questionnaire used a Likert scale to assess six aspects, as illustrated in Figure 12. Participants generally found gesture typing moderately easy, with an interquartile range (IQR) from 3 to 4, suggesting a moderate consensus among them. The IQR for ease of learning is narrow, centered on a score of 4, implying minimal variance and indicating that participants found the method easy to learn. The distribution of ratings on perceived speed is broader, indicating greater variability in how participants perceive the speed of gesture typing. The majority of users perceived gesture typing as a relatively quick input method, which aligns with our previously discussed quantitative results. Perceived accuracy has a distribution similar to that of perceived speed, with few participants rating as having low accuracy. This suggests a contribution from the intelligent text entry framework, *Score Fusion*, in providing accurate text entry predictions. Additionally, many reported minimal hand fatigue, suggesting that the ring-based gesture typing may offer an ergonomic advantage. The plot also indicates that participants had neutral feelings regarding eye fatigue.
- **Feedback for Improvement**: While we received many positive comments and overall feedback sentiment, we also recognize areas for improvement. Several participants highlighted the need for refinement. Specifically, *Participant 3* shared, 'I had to concentrate to keep my middle finger out of the way,' due to the system's method of detecting a pinch gesture, which essentially requires connecting the thumb and index finger to form a closed loop. If the middle finger inadvertently touches the index finger and the sensor signal change surpasses the threshold, the system mistakenly registers this as a pinch gesture, leading to inaccuracies. However, such incidents were infrequent, observed only on rare occasions with *Participant 3* and *Participant 8*.

5.3 Ablation Analysis of Score Fusion Components

To understand the contribution from each component in the *Score Fusion* framework, we performed an ablation analysis on the logged word-trajectory from the Study 2. The results are summarized in Table 4. These results underscore the significance of each component, as each addition led to substantial improvements in accuracy.

6 Discussion

In this paper, we have presented *RingGesture*, a novel ring-based text entry system for lightweight AR glasses. *RingGesture* incorporates an intuitive ring-based mid-air pointing and selection technique that allows users to perform mid-air gesture typing. It also introduces a deep learning word prediction framework, *Score Fusion*, that significantly enhances text entry accuracy and speed. Through two studies, we have demonstrated the effectiveness and usability of *RingGesture*. Study 1 revealed that word-level gesture typing was preferred by users over phrase-level gesture typing. Study 2 demonstrated that *RingGesture*, particularly when utilized with the *Score Fusion* framework, facilitates efficient one-handed text entry, achieving text entry rate of 27.3 WPM. This performance is comparable to mobile phone gesture typing, which also averages around 30 WPM [57], despite the challenges posed by noise and drift in IMU tracking. When using the conventional word prediction framework *Naive Correction*, the average text entry rate for *RingGesture* drops to 17.6 WPM. The enhanced performance with *RingGesture* and *Score Fusion* is due to *Score Fusion*'s accurate word prediction ability, which effectively compensates for the tracking limitations and maintains high performance. The longitudinal evaluation also indicated that users can quickly learn and improve their proficiency with the system over time.

Components	Character Error Rate
GDM	27.86% ± 4.35%
GDM + SSCM	12.12% ± 3.67%
GDM + SSCM + CLM	**5.56**% ± 1.33%

Table 4: Ablation analysis of *Score Fusion*'s components using the dataset logged from Study 2, with outcomes reported as Character Error Rate (CER). Components include GDM (Gesture Decoding Model), SSCM (Spatial Spelling Correction Model), and CLM (Contextual Language Model).

7 Limitations and Future Work

Our system is specifically designed for text entry on lightweight AR glasses equipped with MicroLED technology. A lightweight AR glass creates a virtual screen at a specific distance in front of the user, offering only 3DOF experience. Viewing a traditional monitor, which is placed directly in front of the user and has a fixed screen position, closely mimics the experience of wearing these AR glasses. Additionally, the current iteration of lightweight AR glasses faces battery life limitations, posing challenges for conducting extended user studies. This similarity in the viewing experience supports the argument that using a monitor as a proxy in user studies can effectively replicate the visual setup of these AR glasses. The use of monitors for conducting text entry studies on time-machine heads-up displays has been implemented in several studies [26,45]. However, we acknowledge that this assumption holds true primarily in controlled lab settings and may not extend to real-world scenarios, especially when the user is in motion. Therefore, we plan to assess the *RingGesture* system in real-life, once lightweight AR glasses are enhanced with longer battery life and become more readily usable, as part of our follow-up work. This will involve assessing the performance of the *RingGesture* system in actual AR experiences, particularly in "in the wild" settings such as typing while walking, in a car, or lying on a bed.

The *Score Fusion* framework has the potential to be applied to other decoding methods beyond gesture typing, such as touch typing decoding. Its ability to integrate multiple probabilistic models to enhance word prediction accuracy could benefit various text entry systems. While we did not test *Score Fusion* with other text entry techniques, we consider this an avenue for future work.

8 Conclusion

RingGesture presents a novel ring-based mid-air gesture typing system for lightweight AR glasses, leveraging an intuitive pointing and selection technique. The deep learning word prediction framework, *Score Fusion*, significantly enhances text entry accuracy and speed. User studies demonstrate *RingGesture*'s effectiveness and usability, with entry rates approaching 30 WPM, outperforming previous one-handed text entry techniques for AR/VR. Thus, *RingGesture* demonstrates significant potential for enabling fast text entry in lightweight AR glasses.

References

[1] Enron Email Dataset. https://www.cs.cmu.edu/~./enron/. Dataset. 6

[2] Haptic capacitive. https://sensel.com/product/#haptic-capacitve. 4

[3] Reddit Data via pushshift.io. https://pushshift.io. Dataset. 6

[4] Wikipedia Talk Page Data. https://dumps.wikimedia.org. Dataset. 6

[5] Yelp Dataset. https://www.yelp.com/dataset. Dataset. 6

[6] O. Alsharif, T. Ouyang, F. Beaufays, S. Zhai, T. Breuel, and J. Schalkwyk. Long short term memory neural network for keyboard gesture decoding. pp. 2076–2080, 04 2015. 5

[7] Apple Inc. Apple Vision Pro. https://www.apple.com/vision-pro/, 2023. Accessed: 2024-03-10. 1

[8] Y. Baba and H. Suzuki. How are spelling errors generated and corrected? a study of corrected and uncorrected spelling errors using keystroke logs. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pp. 373–377. Association for Computational Linguistics, 2020. 3

[9] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003. This paper introduces a foundational approach to language modeling using neural networks, setting the stage for the integration of deep learning in language modeling. 3

[10] J. Brooke. Sus: A quick and dirty usability scale. *Usability Evaluation in Industry*, pp. 189–194, 1996. 7

[11] M. Burtsev, V. Logacheva, V. Malykh, I. V. Serban, R. Lowe, S. Prabhumoye, A. W. Black, A. Rudnicky, and Y. Bengio. The first conversational intelligence challenge. In *The NIPS'17 Competition: Building Intelligent Systems*, pp. 25–46. Springer, 2018. 7

[12] F. Cai and M. de Rijke. A survey of query auto completion in information retrieval. *Foundations and Trends in Information Retrieval*, 10(4):273–363, 2016. 3

[13] X. Chen, T. Grossman, and G. Fitzmaurice. Swipeboard: a text entry technique for ultra-small interfaces that supports novice to expert transitions. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pp. 615–620, 2014. 1

[14] W. contributors. Microsoft hololens 2 - wikipedia. 2019. Accessed: 2024-03-10. 1

[15] W. contributors. Apple vision pro - wikipedia. 2023. Accessed: 2024-03-10. 1

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 1:4171–4186, 2019. 6

[17] J. Dudley, H. Benko, D. Wigdor, and P. O. Kristensson. Performance envelopes of virtual keyboard text input strategies in virtual reality. In *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 289–300, 2019. doi: 10.1109/ISMAR.2019.00027 2

[18] J. Dudley, J. Zheng, G. Aakar, H. Benko, M. Longest, R. Wang, and P. O. Kristensson. Evaluating the performance of hand-based probabilistic text input methods on a mid-air virtual qwerty keyboard. In *IEEE Transactions on Visualization and Computer Graphics: forthcoming*, 2023. 2

[19] J. J. Dudley, K. Vertanen, and P. O. Kristensson. Fast and precise touch-based text entry for head-mounted augmented reality with variable occlusion. *ACM Trans. Comput.-Hum. Interact.*, 25(6), article no. Article 30, 40 pages, Dec. 2018. 2

[20] J. J. Dudley, K. Vertanen, and P. O. Kristensson. Fast and precise touch-based text entry for head-mounted augmented reality with variable occlusion. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 25(6):1–40, 2018. 2, 3

[21] J. S. Furtado, H. H. Liu, G. Lai, H. Lacheray, and J. Desouza-Coelho. Comparative analysis of optitrack motion capture systems. In *Advances in Motion Sensing and Control for Robotic Applications*, pp. 15–31. Springer, 2019. doi: 10.1007/978-3-030-17369-2_2 1

[22] Y. Goldberg. *Neural Network Methods for Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2017. An in-depth exploration of how neural network methods are applied in natural language processing, including language modeling. 3

[23] J. Gong, Z. Xu, Q. Guo, T. Seyed, X. Chen, X. Bi, and X.-D. Yang. Wristext: One-handed text entry on smartwatch using wrist gestures. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pp. 1–14, 2018. 1

[24] A. Graves, S. Fernández, and J. Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. In *Artificial Neural Networks: Formal Models and Their Applications–ICANN 2005*, pp. 799–804. Springer Berlin Heidelberg, 2005. 6

[25] T. Grossman, X. A. Chen, and G. Fitzmaurice. Typing on glasses: Adapting text entry to smart eyewear. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, 9 pages, p. 144–152. Association for Computing Machinery, 2015. 1

[26] Y. Gu, C. Yu, Z. Li, Z. Li, X. Wei, and Y. Shi. Qwertyring: Text entry on physical surfaces using a ring. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(4):1–29, 2020. 1, 2, 3, 9

[27] A. Gupta, C. Ji, H.-S. Yeo, A. Quigley, and D. Vogel. Rotoswype: Word-gesture typing using a ring. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pp. 1–12, 2019. 1, 2, 3

[28] J. Henderson, J. Ceha, and E. Lank. Stat: Subtle typing around the thigh for head-mounted displays. In *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM, 2020. 2, 3

[29] H. Jiang, D. Weng, X. Dongye, and Y. Liu. Pinchtext: One-handed text entry technique combining pinch gestures and hand positions for head-mounted displays. *International Journal of Human–Computer Interaction*, pp. 1–17, 2022. 1

[30] H. Jiang, D. Weng, Z. Zhang, and F. Chen. Hifinger: One-handed text entry technique for virtual environments based on touches between fingers. *Sensors*, 19(14), 2019. doi: 10.3390/s19143063 1, 2

[31] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Draft, 3 ed., 2019. A comprehensive guide to the field of speech and language processing, covering both traditional and modern approaches including N-gram models and deep learning techniques. 3

[32] W. Kienzle, E. Whitmire, C. Rittaler, and H. Benko. Electroring: Subtle pinch and touch detection with a ring. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2021. 1, 5

[33] T. Kim, A. Karlson, A. Gupta, T. Grossman, J. Wu, P. Abtahi, C. Collins, M. Glueck, and H. B. Surale. Star: Smartphone-analogous typing in augmented reality. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–13, 2023. 2

[34] B. Klimt and Y. Yang. The enron corpus: A new dataset for email classification research. In *European conference on machine learning*, pp. 217–226. Springer, 2004. 4

[35] P.-O. Kristensson and S. Zhai. Shark2: a large vocabulary shorthand writing system for pen-based computers. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, pp. 43–52, 2004. 5

[36] K. Kukich. Techniques for automatically correcting words in text. *ACM Computing Surveys (CSUR)*, 24(4):377–439, 1992. 3

[37] L. H. Lee, K. Y. Lam, Y. P. Yau, T. Braud, and P. Hui. Hibey: Hide the keyboard in augmented reality. In *2019 IEEE International Conference on Pervasive Computing and Communications (PerCom*, pp. 1–10. IEEE, 2019. 1

[38] Y. Lee and G. J. Kim. Vitty: Virtual touch typing interface with added finger buttons. In *International Conference on Virtual, Augmented and Mixed Reality*, pp. 111–119. Springer, 2017. doi: 10.1007/978-3-319-57987-0_9 2

[39] L. A. Leiva, S. Kim, W. Cui, X. Bi, and A. Oulasvirta. How we swipe: a large-scale shape-writing dataset and empirical findings. In *Proceedings of the 23rd International Conference on Mobile Human-Computer Interaction*, pp. 1–13, 2021. 5

[40] G. Lepouras. Comparing methods for numerical input in immersive virtual environments. *Virtual Reality*, 22(1):63–77, 2018. doi: 10.1007/s10055-017-0312-5 2

[41] C. Liang, C. Hsia, C. Yu, Y. Yan, Y. Wang, and Y. Shi. Drg-keyboard: Enabling subtle gesture typing on the fingertip with dual imu rings. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 6(4), article no. 170, 30 pages, jan 2023. doi: 10.1145/3569463 2, 3

[42] C.-W. Liu, R. Lowe, I. Serban, M. Noseworthy, L. Charlin, and J. Pineau. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. p. 13, 03 2016.

doi: 10.18653/v1/D16-1230 5

[43] I. S. MacKenzie and R. W. Soukoreff. Phrase sets for evaluating text entry techniques. In *CHI'03 extended abstracts on Human factors in computing systems*, pp. 754–755, 2003. 4

[44] A. Markussen, M. R. Jakobsen, and K. Hornbæk. Vulture: a mid-air word-gesture keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 1073–1082, 2014. 1, 2

[45] A. Markussen, M. R. Jakobsen, and K. Hornbæk. Vulture: A mid-air word-gesture keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, 10 pages, p. 1073–1082. Association for Computing Machinery, New York, NY, USA, 2014. doi: 10.1145/2556288.2556964 1, 3, 9

[46] S. E. Maxwell and H. D. Delaney. *Designing Experiments and Analyzing Data: A Model Comparison Perspective*. Lawrence Erlbaum Associates Publishers, 2 ed., 2004. 8

[47] Meta AI. Pytext: A natural language modeling framework based on pytorch. https://github.com/facebookresearch/pytext, 2018. Accessed: insert date here. 7

[48] Meta Platforms, Inc. Oculus Quest Series. https://www.oculus.com/quest/. Accessed: 2024-03-10. 1

[49] Microsoft Corporation. Microsoft HoloLens 2. https://www.microsoft.com/en-us/hololens, 2019. Accessed: 2024-03-10. 1

[50] R. Mitton. *English Spelling and the Computer*. Longman Group, Harlow, Essex, UK, 1996. 3

[51] T. Ogitani, Y. Arahori, Y. Shinyama, and K. Gondow. Space saving text input method for head mounted display with virtual 12-key keyboard. In *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, pp. 342–349, 2018. doi: 10.1109/AINA.2018.00059 2

[52] A. Pauls and D. Klein. Faster and smaller n-gram language models. In *Annual Meeting of the Association for Computational Linguistics*, 2011. 3

[53] A. Peshock, J. Duvall, and L. E. Dunne. Argot: A wearable one-handed keyboard glove. In *Proceedings of the 2014 ACM international symposium on wearable computers: adjunct program*, pp. 87–92, 2014. 1

[54] T. A. Pirinen and K. Lindén. State-of-the-art in weighted finite-state spell-checking. In *Computational Linguistics and Intelligent Text Processing*, vol. 8404, pp. 519–532. Springer, 2014. 3

[55] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019. 3

[56] G. Rakhmetulla and A. S. Arif. Swipering: Gesture typing on smartwatches using a segmented qwerty around the bezel. In *Graphics Interface 2021*, 2020. 1

[57] S. Reyal, S. Zhai, and P. O. Kristensson. 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*, pp. 679–688, 2015. 2, 9

[58] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee. Recent advances in recurrent neural networks, 2017. 6

[59] T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *ArXiv*, abs/1602.07868, 2016. 6

[60] M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. 6

[61] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948. 6

[62] J. Shen, J. Dudley, and P. O. Kristensson. Simulating realistic human motion trajectories of mid-air gesture typing. In *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 393–402. IEEE, 2021. 4

[63] J. Shen, J. Dudley, and P. O. Kristensson. Fast and robust mid-air gesture typing for ar headsets using 3d trajectory decoding. *IEEE Transactions on Visualization and Computer Graphics*, 2023. 2, 3, 5, 6, 7

[64] J. Shen, J. Hu, J. J. Dudley, and P. O. Kristensson. Personalization of a mid-air gesture keyboard using multi-objective bayesian optimization. In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 702–710. IEEE, 2022. 2

[65] J. Shen, B. Yang, J. J. Dudley, and P. O. Kristensson. Kwickchat: A multi-turn dialogue system for aac using context-aware sentence generation by bag-of-keywords. In *27th International Conference on Intelligent User Interfaces*, pp. 853–867, 2022. 3, 6

[66] G. S. Snoddy. Learning and stability: a psychophysiological analysis of a case of motor learning with clinical applications. *Journal of Applied Psychology*, 10(1):1–36, 1926. 8

[67] M. Speicher, A. M. Feit, P. Ziegler, and A. Krüger. Selection-based text entry in virtual reality. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, 13 pages, p. 1–13. Association for Computing Machinery, New York, NY, USA, 2018. doi: 10.1145/3173574.3174221 2

[68] M. Sundermeyer, R. Schlüter, and H. Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012. 3

[69] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017. 6

[70] R. Venkatesan and B. Li. *Convolutional Neural Networks in Visual Computing: A Concise Guide*. CRC Press, 2017. Archived from the original on 2023-10-16. Retrieved 2020-12-13. 6

[71] C. Y. Wang, W. C. Chu, P. T. Chiu, M. C. Hsiu, Y. H. Chiang, and M. Y. Chen. Palmtype: Using palms as keyboards for smart glasses. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, 8 pages, p. 153–160. Association for Computing Machinery, 2015. 2

[72] D. Wolf, J. Gugenheimer, M. Combosch, and E. Rukzio. Understanding the heisenberg effect of spatial interaction: A selection induced error for spatially tracked input devices. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, 10 pages, p. 1–10. Association for Computing Machinery, New York, NY, USA, 2020. doi: 10.1145/3313831.3376876 1, 5

[73] W. Xu, H.-N. Liang, A. He, and Z. Wang. Pointing and selection methods for text entry in augmented reality head mounted displays. In *2019 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 279–288. IEEE, 2019. 2

[74] Z. Xu, Y. Meng, X. Bi, and X.-D. Yang. Phrase-gesture typing on smartphones. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, UIST '22, article no. 55, 11 pages. Association for Computing Machinery, New York, NY, USA, 2022. doi: 10.1145/3526113.3545683 1, 3

[75] Z. Xu, P. C. Wong, J. Gong, T.-Y. Wu, A. S. Nittala, X. Bi, J. Steimle, H. Fu, K. Zhu, and X.-D. Yang. Tiptext: Eyes-free text entry on a fingertip keyboard. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, pp. 883–899, 2019. 1, 2, 3

[76] C. Yu, Y. Gu, Z. Yang, X. Yi, H. Luo, and Y. Shi. Tap, dwell or gesture? exploring head-based text entry techniques for hmds. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 4479–4488. ACM, 2017. 2

[77] C. Yu, Y. Gu, Z. Yang, X. Yi, H. Luo, and Y. Shi. Tap, dwell or gesture? exploring head-based text entry techniques for hmds. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, 10 pages, p. 4479–4488. Association for Computing Machinery, New York, NY, USA, 2017. 1, 3

[78] D. Yu, K. Fan, H. Zhang, D. Monteiro, W. Xu, and H.-N. Liang. Pizzatext: Text entry for virtual reality systems using dual thumbsticks. *IEEE Transactions on Visualization and Computer Graphics*, 24(11):2927–2935, 2018. doi: 10.1109/TVCG.2018.2868581 2

[79] M. Zhao, A. M. Pierce, R. Tan, T. Zhang, T. Wang, T. R. Jonker, H. Benko, and A. Gupta. Gaze speedup: Eye gaze assisted gesture typing in virtual reality. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, pp. 595–606, 2023. 1, 2, 3