# Fast and Robust Mid-Air Gesture Typing for AR Headsets using 3D Trajectory Decoding

Junxiao Shen, John Dudley, and Per Ola Kristensson

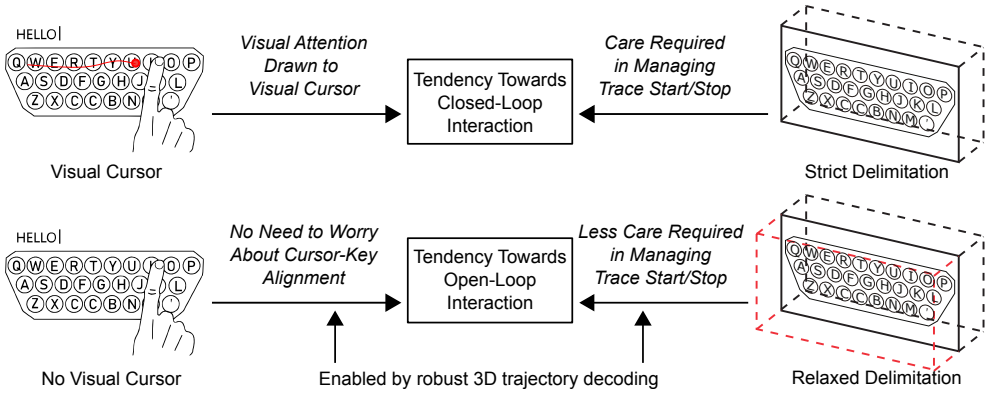



Fig. 1: The conventional approach for supporting mid-air gesture typing in AR is to utilize a cursor and trace visualization in combination with strict delimitation of gesture start/stop based passing into or out of the keyboard plane. These qualities of the interface introduce a tendency towards closed-loop interaction due to visual attention being drawn to the visual elements and the care required in indicating the start and end of a gesture. Eliminating the visual cursor and relaxing the way in which gesture start/stop is delimited may help promote open-loop interaction and allow users to type more quickly. However, removing the cursor and relaxing the delimitation can introduce additional error into the articulated trace which must be mitigated by more advanced decoding. We therefore introduce a novel 3D trajectory decoding method that directly translates users' 3D trajectories into text without relying on the trajectories projected onto the 2D keyboard plane.

**Abstract**— We present a fast mid-air gesture keyboard for head-mounted optical see-through augmented reality (OST AR) that supports users in articulating word patterns by merely moving their own physical index finger in relation to a virtual keyboard plane without a need to indirectly control a visual 2D cursor on a keyboard plane. To realize this, we introduce a novel decoding method that directly translates users' three-dimensional fingertip gestural trajectories into their intended text. We evaluate the efficacy of the system in three studies that investigate various design aspects, such as immediate efficacy, accelerated learning, and whether it is possible to maintain performance without providing visual feedback. We find that the new 3D trajectory decoding design results in significant improvements in entry rates while maintaining low error rates. In addition, we demonstrate that users can maintain their performance even without fingertip and gesture trace visualization.

**Index Terms**—Text Entry, Machine Learning, Augmented Reality

## 1 Introduction

Text entry is a fundamental activity [20], and an essential one for optical see-through augmented reality (OST AR) to become mainstream. One potentially promising solution to text entry in AR is the gesture keyboard [26, 43], which allows users to write quickly by articulating word patterns in relation to a virtual keyboard.

The success of the gesture keyboard has been investigated in prior work and can at the high-level be attributed to two particular traits: 1) its relatively high entry rate in relation to competing text entry methods on mobile devices; and 2) its high familiarity with existing methods, thus minimizing any learning effort from users [19, 21]. Gesture keyboards can achieve this by exploiting a familiar QWERTY keyboard layout. Novice users can exploit this familiarity and write by sliding from key to key. In doing so they gradually build up motor patterns of frequent word gesture trajectories due to motor memory consolidation. Once these patterns have been learned they can be quickly recalled directly from motor memory [26, 43]. Thus a gesture keyboard provides a continuum between a complete novice having to trace each word gesture trajectory on the keyboard, to a complete expert able to rapidly recall gestures for words directly from motor memory. From a human motor control point-of-view, the user experiences a smooth transition from slow closed-loop visual feedback-based writing to fast open-loop direct recall from motor memory. It is this latter quality—the ability to write open-loop—that enables the high entry rates that are achievable with practice [26, 31, 44].

Thus, for a gesture keyboard to be successfully transplanted to another medium, such as an AR headset, it is important to maintain this quality of allowing users to quickly recall the word gesture trajectories directly from motor memory. However, such recall is fast and imprecise and thus challenging for a system to interpret correctly. It is also vital the system does not rely on closed-loop interaction, in particular visual feedback or difficult to manage trace gesture delimitation, for the user to be able to accurately use the system to write. The unique qualities of the keyboard interface that can either promote or hinder open-loop interaction are summarized in Figure 1. This paper proposes a novel approach to gesture keyboard decoding and interface design for AR headsets that tackles these design requirements. As we will demonstrate, our approach delivers mean entry rates of 27 WPM, which can be compared with the 33 to 45 WPM observed by Reyal et al. [31] for users gesture typing on their own personal phone in the wild.


- *Junxiao Shen is with University of Cambridge. E-mail: js2283@cam.ac.uk.*
- *John Dudley is with University of Cambridge. E-mail: jjd50@cam.ac.uk.*
- *Per Ola Kristensson is with University of Cambridge. E-mail: pok21@cam.ac.uk.*

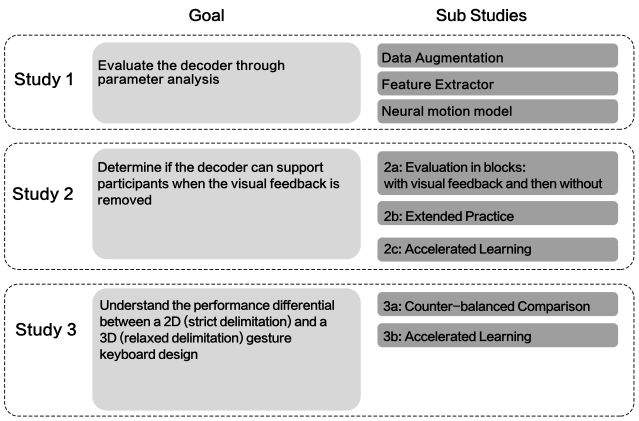



Fig. 2: Summary of the studies presented in this paper.

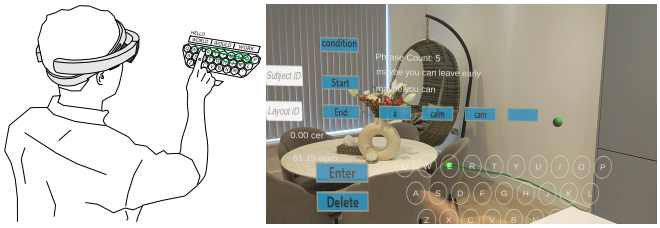



Fig. 3: **Left:** Illustration of a user gesturing on a mid-air gesture keyboard in AR. **Right:** Appearance of the mid-air gesture keyboard in AR. The cursor and trace indicate the tracked fingertip position. The color of the cursor and the sphere on the top right corner of the keyboard indicates the lift on/off states. Green represents lift on and red represents lift off. The buttons such as 'condition', 'start', 'end', and the input boxes such as 'Subject ID', 'Layout ID' are only used for experimental purposes. Below the stimuli, there are alternative word choices. Pressing "Enter" confirms the selection and proceeds to the next stimulus. Pressing "Delete" removes the previously entered word. When we remove the visual feedback, it is only the visual cursor and gesture trace that is removed. The sphere on the top right corner of the keyboard still indicates the lift on/off states.

The central idea is to present the user with a mid-air virtual keyboard and allow the user to articulate word gesture patterns by merely moving their physical index finger in the vicinity of a mid-air keyboard plane. Thus, the user does not need to rely on closed-loop interaction to articulate word gesture trajectories. This provides users with an experience that eliminates the need for the user to have any concern regarding whether the system is lagging in its tracking of the finger position or whether their 3D articulations are intersecting with a virtual keyboard plane or not. This produces a true physical-virtual text entry experience unencumbered by current hand tracking limitations and we hypothesize that realizing such a system enables entry rates that are potentially comparable to entry rates observed on mobile phones using commercial gesture keyboards.

However, realizing such a system means tackling three challenges. First, unlike a touchscreen gesture keyboard, there is a *delimitation* challenge: the system has to automatically determine what part of the user's 3D trajectory actually constitutes the beginning, middle and end of a word pattern gesture on the virtual keyboard plane. Further, for the system to be effective the system must be able to perform this task at a very high accuracy.

The second challenge is that users cannot be expected to be able to align their own physical index finger in relation to the virtual keyboard plane. This *alignment* problem arises due to both imprecise tracking and users' inability to determine their finger location in relation to a virtual plane. Again, the system must be able to solve this alignment problem consistently at a very high accuracy.

The third challenge is a *lack of data*. Unlike traditional touchscreen-based gesture keyboards, an AR gesture keyboard that successfully tackles both the delimitation and the alignment challenge at a consistent high accuracy necessitates a deep learning-based decoder that will require a large amount of representative training data. However, modern AR headsets are still in their infancy and the only feasible way to collect raw training data is to recruit participants and manually collect training data, which does not scale.

This paper demonstrates that these three challenges can be successfully tackled, thereby enabling users to write very fast using an AR gesture keyboard. This is achieved using a novel decoder consisting of a neural motion model that uses an attention mechanism to learn a strong alignment between gestural 3D trajectories and text. The model is further trained using a new data augmentation scheme and feature extractor to tackle the lack of training data. This decoding method is the first technique that directly translates 3D word-gesture trajectories into text.

We study the efficacy of this system in three studies (Study 1–3), as summarized in Figure 2. These studies demonstrate that the decoder is able to accurately infer users' intended text, users are able to write quickly and accurately using the system, and users are able to maintain their performance even without any fingertip cursor or visual gesture trace feedback. In an accelerated motor learning task where participants are asked to repeatedly write single sentences, six out of the eight participants achieve error-free peak entry rates surpassing 50 WPM and one participant reached over 70 WPM, illustrating the human performance potential of this AR gesture keyboard approach. We also carry out two studies demonstrating that the new 3D design for an AR gesture keyboard results in a significant 13.1% relative improvement in entry rate compared to a conventional gesture keyboard derived from Microsoft Hololens 2 system keyboard. Using an accelerated motor learning task, we observe a significant 8.7% relative improvement in favor of the 3D design.

In summary, this paper makes the following contributions:

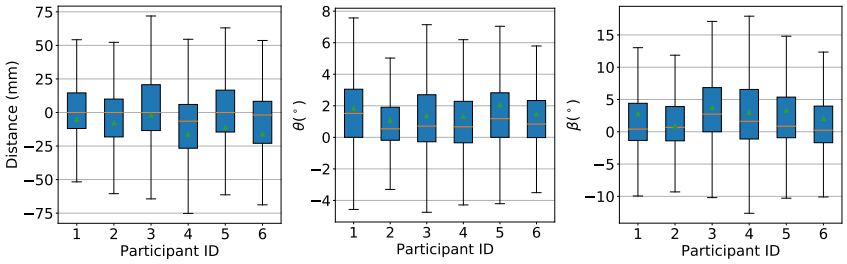
1. We present a novel word-gesture decoding architecture underpinned by a) our neural motion model, AE-BLSTM-CTC; and b) our data augmentation method and feature engineering strategy. The decoder is the first of its kind in directly translating mid-air 3D word-gesture trajectories into text and it achieves this by overcoming several critical challenges encountered when using a mid-air word gesture keyboard on an AR headset.
2. We propose novel design concepts, including removing the visual cursor and the *3D* gesture keyboard design. The robustness of the novel word-gesture decoding architecture plays a significant role in facilitating the evaluations of these novel interface designs in Study 2 and Study 3. Additionally, it should be noted that the impact of these design changes has not been previously investigated.
3. We open-source this novel word-gesture decoding architecture as well as the training data to enable designers, developers and researchers to prototype different mid-air gesture keyboard designs.

## 2 Related Work

The increasing uptake and growing capabilities of modern AR/VR Head-Mounted Displays (HMDs) has stimulated greater interest in supporting efficient text input on these devices. Given the focus of this paper we limit our discussion of related work to text entry using mid-air virtual keyboards. Even within this narrow scope, a wide variety of interaction methods and keyboard designs have been explored.

### 2.1 Mid-Air Virtual Keyboards

In virtual reality (VR), Yu et al. [40] investigated the performance of three mid-air text entry alternatives for VR by utilizing the head movement to control a head-fixed gaze cursor. The three alternatives are: TapType which requires a user to select a letter by pointing to it




(a) Distance (b) theta (c) beta

Fig. 4: Box plots of the three alignment metrics between the gesture plane and rendered keyboard plane for the six participants. The green triangle shows the mean.

and tapping a button on a gamepad, DwellType which requires a user to select a letter by pointing to it and dwelling over it for a period of time, and GestureType which requires a user to perform word-level input using a gesture typing style (also using a gamepad to indicate gesture start and finish events). Yu et al. [40] found that users typed at 10.59 WPM, 15.58 WPM, and 19.04 WPM with DwellType, TapType, and GestureType, respectively. The Wizard-of-Oz study performed by Dudley et al. [9] suggests that much higher entry rates in the range of 40 WPM may be achievable with direct touch interaction using two index fingers on a mid-air VR keyboard. Yi et [39] explored the feasibility of single-finger typing on mid-air virtual QWERTY keyboards in VR, addressing the lack of tactile feedback by using a novel input prediction algorithm and achieving an entry rate of 26.1 WPM with 3.2% uncorrected word-level error rate.

In optical see-through augmented reality (OST AR), Dudley et al. [11] presented the VISAR keyboard, which is a keyboard for AR HMDs supporting text entry via a virtualized input surface. After several design iterations, the system was able to support a mean entry rate of 17.75 WPM with a mean character error rate of less than 1%. Xu et al. [38] evaluated alternative text entry methods in AR with different combinations of selection methods (head, hand, hybrid, and controller) and input mechanisms (Swype and Tap). The performances of the different combinations varied from 5 WPM to 15 WPM [38]. Both Dudley et al. [11] and Xu et al. [38] noted that the system's performance is affected by tracking latency and inaccuracy caused by the hardware (tracking cameras) and the software (gesture detection algorithms). Dudley et al. [11] reported that the typical lag in hand position tracking is approximately 220 ms on the HoloLens 1. Although the tracking capability on the Microsoft HoloLens 2 is much improved, there is still a noticeable lag and inaccuracy.

The various studies of mid-air text entry methods in AR/VR suggest typical entry rates of 5 to 26 WPM [10,11,36,38]. Such entry rates may be inadequate for high-volume text input, especially when compared to simply using physical keyboards in VR which have been shown to support fast text entry of around 45 to 67 WPM [18]. Touch typing on an uninstrumented flat surface also demonstrates fast text entry of 73 WPM in an offline study [32].

### 2.2 Keyboard Gesture Recognition

Gesture keyboards [19, 43] provide an efficient text entry method by mapping word gestures drawn on a keyboard layout into words [31]. Gesture keyboards have, since their invention, become an established input method in modern smartphones [5, 42, 45], smartwatches [12], tablets [3] and Head Mounted Displays (HMDs) [41]. Reyal et al. [31] performed a comparative study between touch and gesture text input on a smartphone both in a lab setting and 'in the wild'. The text entry rate reached an average speed of 28 to 39 WPM, depending on the method and the user's experience level. Reyal et al. [31] observed that users tend to prefer gesture entry over touch entry in both focused and mobile situations. Leiva et al. [27] conducted a large-scale web-based study and found that gesture typing entry rates typically varied from around 50 WPM for everyday users to around 40 WPM for users who never otherwise use a gesture keyboard.

SHARK$^2$ [26] is a template matching algorithm for word-gesture keyboards. Vulture [29], a word-gesture keyboard designed to operate in mid-air, also used a template matching algorithm. Although the Vulture keyboard supported mid-air interaction, entry was performed by moving the hand to control a cursor projected onto the wall and so the trace actually decoded was 2D rather than 3D. Gesture keyboards are often unimanual, but Bi et al. [3] created a novel bimanual gesture text entry system that extends the gesture keyboard paradigm from one finger to multiple fingers. They designed a multi-stroke gesture recognition algorithm based on a unimanual gesture recognition algorithm that is similar to SHARK$^2$. However, this approach introduces learnability issues and is more applicable in situations where mobile device users need to hold the device with both hands and type with two thumbs.

Alsharif et al. [2] present a machine learning-based gesture recognizer designed to address some of the challenges encountered in mobile gesture typing, such as elision, co-articulation, and high variability. This recognizer comprises a recurrent neural network and a Connectionist Temporal Classification (CTC) loss [14]. This approach provided a recognition accuracy of 89.2%, a 22% absolute improvement relative to a baseline which is a template-matching-based system similar to SHARK$^2$. Biju et al. [4] proposed a transformer-based gesture decoding model to support mobile gesture typing in Indic languages. The overall accuracy of this model across seven Indic languages varied from 70-95%. Both Alsharif et al. [2] and Biju et al. [4] were fortunate in having access to sufficient training data thanks to the maturity of mobile-phone-based gesture typing. Alsharif et al. [2] collected over 50,000 trajectories, and Biju et al. [4] collected over 190,000 trajectories. No such datasets are readily available or readily collected for mid-air gesture typing on an AR HMD. Further, the two models proposed by Alsharif et al. [2] and Biju et al. [4] were designed to decode 2D trajectories, whereas our method decodes 3D trajectories directly to text. To our knowledge, direct decoding of 3D word-gesture trajectories into text has not been previously explored.

## 3 The Three Key Challenges

In this section, we revisit the three key challenges that motivate the design of our AR mid-air gesture keyboard before introducing the decoding system that addresses these challenges in Section 4.2.

As highlighted in the Introduction, developing a robust and effective mid-air gesture keyboard for AR requires overcoming three key challenges: delimitation, alignment and lack of data. We examine each of these key challenges below with concrete reference to observed user behaviors when typing in AR on a mid-air gesture keyboard.

We captured typical user behavior by performing a Wizard-of-Oz study [7] to collect high-quality trajectory data from different subjects gesturing on different keyboard geometries. We deployed the SHARK$^2$ recognizer [26] into the mid-air gesture keyboard and implemented a simulated robust error correction mechanism. The keyboard geometry, in terms of width and aspect ratio, were periodically changed to capture data across a variety of scales and sizes. The keyboard width ranged from 100 mm to 600 mm and height-width aspect ratio ranged from 0.1875 to 0.5. Six participants (three male, three female, age mean =

22.7, standard deviation = 3.9) participated in the study. All 6 participants spent approximately 3 hours performing mid-air gesture typing during data collection. Data analysis confirmed that participants consolidated their experience with the keyboard after 30 minutes of usage. The stimulus phrase set was taken from the Enron Mobile Corpus [17] and the MacKenzie phrase set [28]. Overall, we captured 2,428 unique words and 1,598 phrases (each phrase contains an average of 7.05 words; maximum: 15 words, minimum: 4 words). The stimuli phrases were selected randomly from the stimulus phrase set. We used this collected data to train the neural motion model of our decoder, which will be explained in the next subsection. The keyboard presented to the participants provided visual feedback indicating the tracked fingertip.

### Challenge 1: Delimitation

The major issue observed in the collected dataset regarding delimitation was the presence of extraneous points at the beginning and end of a word trace. We refer to the extraneous points at the start of the trace as the 'head' and the extraneous points at the end of the trace as the 'tail'. These heads and tails are caused by the user lifting 'on' and 'off' from the gesturing plane. Figure 5 shows the head and tail of the trajectory for the word 'sweet' in both the keyboard plane and 3D space.

### Challenge 2: Alignment

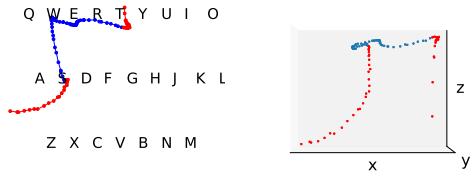
Our dataset suggests that different users have different perceptions of the keyboard plane as displayed in the Microsoft HoloLens 2. The trajectories of different users fit into different gesture planes with various positional and orientation offset from the rendered virtual keyboard plane. We examined these gesture planes according to three metrics:

1. Distance: the distance of the center of the gesture plane from the keyboard plane.
2. theta $\theta$: the angle between the normal vector to the gesture plane and the $z$-axis in the $z$-$x$ plane. In other words, the rotation angle between the gesture plane and the keyboard plane with the $y$-axis.
3. beta $\beta$: the angle between the normal vector to the gesture plane and the $z$-axis in the $z$-$y$ plane. In other words, the rotation angle between the gesture plane and the keyboard plane with the $x$-axis.

Our paper presents a novel contribution by introducing metrics aimed at quantifying the alignment between the virtual keyboard plane and the virtual gesturing plane. From a geometrical standpoint, the relative position of one plane to another can be adequately described by the distance and rotational angles (denoted by lowercase Greek letters, theta $\theta$, and beta $\beta$). Figure 4 shows box plots of these three metrics for the six participants in the initial data collection. We observe individual differences in the gesture plane. Most people tend to gesture in front of the rendered keyboard plane, as indicated in Figure 4a. Figure 4b and 4c show that users are better at aligning their gesture plane and the rendered keyboard with the $x$-axis than the $y$-axis.

### Challenge 3: Lack of Data

A gesture keyboard is subject to various sources of noise and ambiguity. For example, different words may have spatially similar trajectories. Cognitive and motor errors may also manifest as misspellings, character insertions, deletions or substitutions. When gesture typing in AR, trajectories can be erratic and inconsistent due to the large operational space and the lack of tactile feedback. Deep learning offers a potential method for decoding these noisy and ambiguous trajectories into text but would typically demand a large annotated corpus of data. Ideally every token in the supported lexicon is represented to achieve a high level of robustness. However, a key requirement for training a deep-learning-based gesture keyboard recognition model is access to numerous representative user trajectories. The collection and annotation of such a dataset is time-consuming and prohibitively expensive. Further, due to the limited adoption of AR headsets such data cannot be easily collected by, for example, crowdsourcing.




(a) In the keyboard plane. (b) In 3D space.

Fig. 5: Illustration of the head and tail of the gesture trace for the word 'sweet'. Blue dots comprise the core of the trajectory in the keyboard plane while the red dots show the head and tail. The $z$-axis is normal to the keyboard plane

## 4 Mid-Air Gesture Keyboard for AR

In this section, we firstly reveal in detail of the 3D gesture keyboard design we proposed and discuss the different between it with a conventional 2D gesture keyboard design from Microsoft HoloLens 2 system keyboard. Then we present the decoding architecture which translates 3D gesture trajectories into text.

### 4.1 User Interface

The *3D* mid-air gesture keyboard was developed for deployment on the Microsoft HoloLens 2, a commercially available AR headset, and leverages the integrated hand tracking available on-device. Word gestures are performed by moving the tip of the index finger over the keyboard layout. To start a new gesture the user moves their finger towards the keyboard plane and the gesture is completed by retracting the finger. This start and end delimitation is governed by a distance threshold such that individual word gestures are roughly segmented. The segmented 3D trajectories of the fingertip provide the raw observation sequences to the system's decoder that translates these 3D trajectories into the user's intended text. The output of the decoder is a ranked list of word hypotheses. The top result is automatically inserted into the input field and the next most likely hypotheses are presented as selectable alternatives displayed in the keyboard interface. The implementation allows for easy configuration of the visual features presented to the user. Figure 3 shows the appearance of the gesture keyboard as viewed through the Microsoft HoloLens 2.

### 4.2 Gesture Decoding Architecture

We now present the decoding architecture which translates 3D gesture trajectories into text. Section 4.2.1 introduces the neural motion model which serves as the gesture recognition model. It maps the input, consisting of the 3D coordinates of the gesture trajectory concatenated with additional crafted features (detailed in subsection 4.2.3), to a sequence of vectors containing normalized probabilities for the set of 26 English characters (plus one additional 'blank' pseudo-character indicating no output). Finally, subsection 4.2.2 describes the data augmentation approach used to generate the synthetic gesture traces in order to increase the amount of training data available to the neural motion model.

#### 4.2.1 Neural Motion Model

We propose a neural motion model we call Attention-Enhanced Bi-directional-LSTM with CTC loss (AE-BLSTM-CTC) to overcome the challenges facing a mid-air gesture keyboard for AR. This network's structure is illustrated in Figure 6. The key component in the AE-BLSTM-CTC is the attention mechanism. An attention mechanism is an important aspect in sequence to sequence (seq2seq) problems [14, 35].

The attention mechanism is used to focus on certain parts of the trajectories and thus it can learn to ignore the heads and tails we previously identified in the 3D gesture trajectories. In addition, it can learn from both spatial and temporal correlations. It also allows the recognizer to learn positional dependencies and to focus on local or global

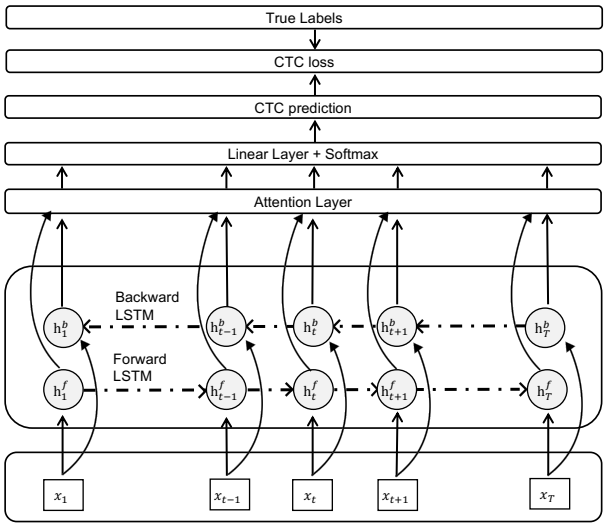



Fig. 6: Network structure of the proposed AE-BLSTM-CTC recognition model. $x_1, x_2, ..., x_T$ is the input sequence of length $T$ passed to the Bidirectional-LSTM layer. $h_1^f, h_2^f, ..., h_T^f$ and $h_1^b, h_2^b, ..., h_T^b$ are the output sequences of length $T$ from the Forward LSTM layer and Backward LSTM layer respectively. They are also the input to the attention layer. The attention layer implements the attention mechanism.

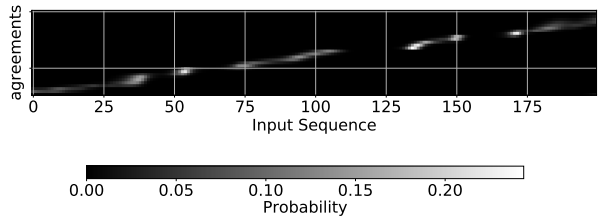



Fig. 7: Attention map for gesture of the word 'agreements'. This is the heat map of the attention weight at different input locations with respect to the target output. It shows where in the output word the model is concentrating its attention throughout the gesture's input sequence.

features. Therefore, we expect the attention mechanism to better tackle the different challenges encountered with a mid-air gesture keyboard, such as the heads and tails, the high levels of articulation noise and the ambiguities in user input. We produce Figure 7 by taking the attention weights of the trajectory input from the AE-BLSTM-CTC. An attention weight is a probability of how strongly the model should attend to relate the sequential position of the input sequence to the output sequence. A greater attention weight means that there is strong attention from the model between the positional information of the input and output sequence. Such attention is learned from the training data. We can observe that the model can estimate accurately where the attention should be along the input sequence.

Gesture recognition has similarities to handwriting recognition and thus it is unsurprising that a connectionist temporal classification (CTC)-based model from that problem domain can be used to decode gestural trajectories on mobile gesture keyboards [14]. We leverage the AE-BLSTM-CTC model as the spatial neural model to map the input to logits, a sequence of vectors of raw (non-normalized) predictions of the input sequence. This input sequence is then passed into a softmax function to generate a sequence of vectors of normalized probabilities with one value for each possible class of the 26 English characters and an extra class for blank. The blank class is used to encode duplicate characters. This sequence is of the same length to the input. The output from the softmax function is then decoded to text.

#### 4.2.2 Data Augmentation

Shen et al. [33] proposed the Imaginative-GAN (Generative Adversarial Network) as a model for generating synthetic hand gesture skeleton data to improve hand gesture recognition models. Shen et al. [34] subsequently leveraged this GAN model, along with other models, to synthesize gestural trajectories for different scenarios, such as deep neural network training and design optimization. However, they did not implement the methods into actual decoders and they also did not release their dataset. We extend this prior work by fine-tuning the approaches and developing actual functional decoders. To be more specific, we utilized the GAN-Imitation model from Shen et al. [34] given its suitability for deep-learning training and the fact that the latent attributes extracted from GAN models were found to be the closest matches to real data. We use this model to synthesis trajectories for data augmentation to improve training accuracy of the keyboard recognition model.

#### 4.2.3 Feature Vector

We further augment the input to the model by extracting additional features. The feature vector contains six low-level geometric metrics (curvature, aspect, curliness, linearity, slope, and offset) used by Shen et al. [34] for characterizing gestural trajectories. The feature vector also contains key activations following a similar approach to Alsharif et al. [2]. This is a $26 \times 1$ boolean vector such that there is one element for each key in the layout, and element $i = 1$ if the $(x, y)$ coordinate falls within the corresponding key boundary.

## 5 Evaluation

We evaluate the mid-air gesture keyboard in three studies. These studies assess the effectiveness of the gesture decoding architecture introduced in Section 4. Note that the focus of this paper is examining the interface features that influence a user's tendency towards either open-loop or closed-loop interaction when gesture typing. Comparing the performance of our keyboard against alternative input methods, such as touch typing or speech [1], is relevant future work but beyond the scope of the current investigation. Study 1 is a parameter analysis that examines the sensitivity of different design parameters in the decoder. Study 2 is a series of user studies (Study 2a–2c) that investigate the performance of the decoder with real users in a real-time setting and simultaneously allows us to examine the entry rates that are achievable. As an additional objective, Study 2 also investigates whether it is possible to turn of visual feedback and maintain performance. Finally, Study 3 is a pair of user studies (Study 3a and 3b) that investigates the user performance differential obtained by using 3D trajectory decoding for mid-air gesture typing instead of the state-of-the-art 2D approach.

We report the results of the studies using the following metrics:

- Words Per Minute (WPM) indicates the number of words a person can type within a minute.
- Character Error Rate (CER) is the minimum number of character insertion, deletion, and substitution operations that transform the response text into the stimulus text, divided by the length of the stimulus text.
- Uncorrected CER and Corrected CER represent the CER in the predicted text output before and after applying any corrections, respectively. The corrections include selecting word alternatives, deleting words and re-entering (see Figure 3).
- Word Error Rate (WER) is the number of errors at the word level divided by the total number of words in the stimulus text. Both CER and WER are reported as percentages.

### 5.1 Study 1: Parameter Analysis

We here present an analysis of the decoder's sensitivity to variation of the main training and design parameters including the data augmentation method, the feature extractor and the neural motion model. Such an analysis offers insights into the different components of the system and how they impact performance [22, 24].

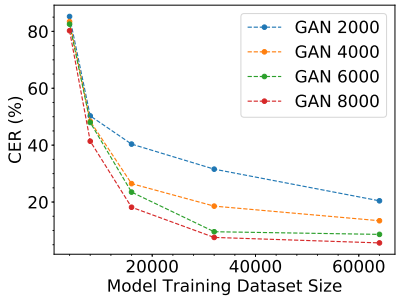



Fig. 8: The recognition accuracy in CER ($y$-axis) of the model trained from different sizes of dataset ($x-$axis) which are generated from different GAN-Imitation models. 'GAN 2000' represents a GAN-Imitation model trained from 2000 real trajectories. 'GAN 4000' represents a GAN-Imitation model trained from 4000 real trajectories.

Table 1: Recognition results of the neural motion model trained from synthetic data generated from different data augmentation models.

| Data Augmentation Method | CER | WER |
|---|---|---|
| GAN [30] | 8.19% | 27.91% |
| Jerk-Minimization [2] | 40.12% | 72.43% |
| **GAN-Imitation** (Ours) | **5.41%** | **18.99%** |

Unless otherwise noted, we use a random 5% of the entire dataset as the test dataset, another random 5% as the validation dataset, and the rest as the training dataset. A validation dataset is used to give an estimate of model performance in evaluation. The performance results are reported from test data.

#### 5.1.1 Data Augmentation

The trajectories synthesized from the GAN-Imitation model not only reflect the spatial correlations (positional information) but also the temporal correlations (speed profiles). Different variants of the synthesized trajectories can be produced by setting different random seeds. Therefore, one real trajectory can theoretically produce infinitely many different variants with slightly altered spatial and temporal information. We evaluated the following two parameters controlling the amount of data used to both train the GAN-Imitation model and the amount of data generated by the GAN-Imitation model to train the decoder model:

1. The number of real trajectories used to train the GAN-Imitation model.

2. The number of trajectories synthesized from the GAN-Imitation model to train the decoder model.

Note that we have an upper limit of 10,253 real trajectories from the dataset collection described in Section 3 for training the GAN-Imitation model. Figure 8 shows the impact of varying these two values on decoder performance. We observe that the more data the recognition model and the GAN-Imitation model are trained on, the better the recognition performance. For subsequent parameter analysis, and in the deployed system itself, we use 8,000 real trajectories to train the GAN-Imitation and 64,000 generated trajectories to train the decoder model.

To assess the benefit brought by using the GAN-Imitation method over alternative data augmentation methods, we introduce two alternative methods for comparison: a GAN model proposed by Mehra et al. [30]; and a Jerk-Minimization model from Alsharif et al. [2]. Table 1 shows that GAN-Imitation outperforms the other two approaches.

#### 5.1.2 Feature Extractor

Next we evaluate the contribution of the different features in the feature extractor. Section 4.2.3 introduced the different features that make up the input vector. These are the coordinates of the trajectory, the key activations and a collection of low-level geometry features (curvature, aspect, curliness, linearity, slope, and offset). Using only coordinates result in 13.6% CER while introducing key activations reduces the CER to 7.3%. Adding geometry features reduces CER further to 5.4% demonstrating that having all features provide a clear benefit to recognition performance.

Table 2: Recognition accuracy of alternative decoder models.

| Recognition Model | CER |
|---|---|
| SHARK2 [26] | 35.34% |
| BLSTM-CTC [2] | 6.53% |
| Joint Transformer/RNN | 6.10% |
| Speech-Transformer | 23.23% |
| LAS | 26.43% |
| RNNT | 26.53% |
| **AE-BLSTM-CTC** (Ours) | **5.41%** |

#### 5.1.3 Neural Motion Model

We compared the performance of our neural motion model against a range of different models including those from Biju et al. [4] and Alsharif et al. [2], which are Joint Transformer/RNN and LSTM-CTC respectively. We further leveraged different state-of-the-art models, including Speech-Transformer [8], Listen, attend, spell (LAS) [6] and Recurrent Neural Network-Transducer (RNN-T) [13] from Automatic Speech Recognition (ASR) as the goal of ASR is also a sequence decoding problem. Although handwriting recognition also serves the same purpose, there are very few advanced deep learning-based models in the literature. We also used SHARK$^2$ [26] as another baseline by projecting the 3-dimensional trajectory to the keyboard plane and using the projected trajectory as the input to the SHARK$^2$ decoder.

Note that the training/test/validation dataset split, the data augmentation method, and the feature extractor all remained the same throughout the experiments. Only the models were varied.

Table 2 shows that the attention mechanism in our model helps to improve the BLSTM-CTC model by 1.12% and no models leveraged from ASR perform well. The improved character error rate achieved by our neural motion model, compared to BLSTM-CTC which shares a similar backbone structure but lacks the attention mechanism, highlights the proficiency of our model in recognizing the heads and tails of the input sequence. Additionally, Figure 7 provides further evidence of the attention mechanism's effectiveness in disregarding the heads and tails of the input sequence. The model concentrates on establishing a strong correlation between the gesture trajectory and the corresponding character position in the output.

### 5.2 Study 2: User Performance with and without Visual Feedback

Study 2 consists of three investigations, which we will call Study 2a, 2b and 2c.

#### 5.2.1 Study 2a: User Performance

Study 2a has two objectives. The first objective is to evaluate the mid-air AR keyboard's text entry performance in a user study. The second objective is to examine if it is indeed possible to turn off the cursor indicating the tracked finger position and the gesture trace and still maintain the same performance. Study 2a was structured as follows:

1. **Participants**: We recruited 18 participants using opportunity-sampling from our university campus. 14 were male and 4 were female and the average age was 22.8 (s.d. 2.7).

2. **Materials**: We used stimulus phrases from the no-number phrases subset of the Enron mobile message dataset [17]. The phrases subset were filtered to those with 40 or fewer characters, 4 words or more, in line with prior research. [11]. We produced two stimulus phrase sets consisting of 40 phrases each, and the order of the stimulus phrase sets for the two blocks was counter-balanced. Also, in line with prior research (e.g. [11]) we controlled the perplexity of the two phrase sets, which essentially means that on average both phrase sets were roughly

Table 3: Entry and error rate descriptive statistics for Study 2a (*mean* ± *SD* [*min*, *max*]).

| Block | Entry Rate (WPM) | Uncorrected Character Error Rate | Corrected Character Error Rate |
|---|---|---|---|
| With Visual Feedback | 18.36 ± 4.10 [12.66, 25.65] | 6.18 ± 1.83 [2.83, 9.39] | 0.88 ± 0.88 [0, 3.26] |
| Without Visual Feedback | 21.39 ± 6.15 [13.17, 32.65] | 6.16 ± 2.10 [1.71, 9.58] | 0.62 ± 0.64 [0, 2.36] |

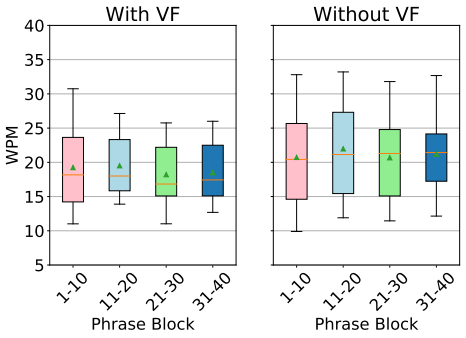



Fig. 9: Box plots of entry rate (WPM) over phrase subsets 0-10, 10-20, 20-30 and 30-40 in Study 2a. The green triangles show the group mean.

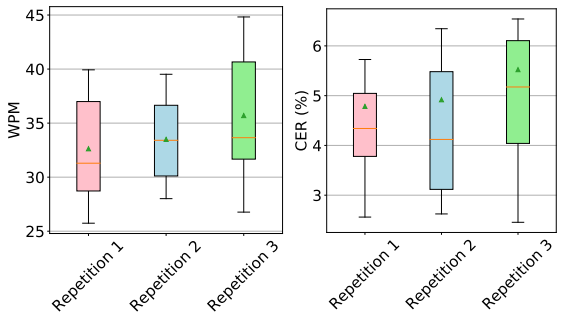



Fig. 10: Box plots of the participants' mean entry and character error rates over the three repetitions of the 40 phrases in Study 2b. The green triangles show the group mean.

equally difficult to predict for the system. The stimulus phrases were sampled randomly and filtered to ensure there were no duplicate phrases. Finally, the two phrase sets contained different phrases compared to the phrases used to train the decoder. There were in addition eight unique words in the two phrase sets that were not included in the data used to train the decoder. This is to test if the decoder is capable of generalizing to unobserved words. We used a language model combined with an auto-correction module of vocabulary size over 50,000 [37]. We used a Microsoft HoloLens 2 as the AR headset and decoded the trajectory data from the Microsoft HoloLens 2 on a dedicated PC. The connection was wireless using Wi-Fi.

3. **Protocol**: The user study had two blocks. In the first block participants wrote with visual feedback turned on, which means the tracking cursor and a gesture trace was present. In the second block participants wrote with this visual feedback turned off. Note that the order of these two blocks was not counter-balanced. This is because this visual feedback is necessary at the time of initial exposure for users to understand how the system works. However, we hypothesize that after some initial exposure, the visual feedback can be turned off without any detrimental effect to performance. At the start of each block, participants were asked to gesture fifteen practice phrases that were not included in any of the two stimulus phrase sets. Participant were encouraged to ask questions and make sure they understood the interaction mechanism and the keyboard functionality during practice. Having completed the practice stage, participants were asked to enter stimulus phrases as quickly and as accurately as possible. In addition, they were asked to slow down if theirs error rate was consistently above 10%. The keyboard displayed entry rate (WPM) and character error rate (CER) for each phrase to participants. Stimulus phrases were sampled randomly from the current phrase set. At the completion of the study, participants were asked to comment on their experience of using the keyboard with or without the visual feedback.

### 5.2.2 Results

We calculate entry rate based on the measured entry time from the first 'lift on' movement for the first character until the last 'lift off' movement for the last character in the sentence. Table 3 shows the entry and error rate for both blocks (with and without visual feedback). Participants achieved a mean entry rate of 18.4 WPM in the first block with visual feedback and 21.4 WPM in the second block without visual feedback. We do not argue that this represents a significant improvement in performance and we avoid running a statistical analysis because these two blocks were performed in sequence and were not counterbalanced. Rather, we report these values to illustrate that the decoder does indeed support participants in maintaining their entry rates when the visual feedback is removed.

Figure 9 shows box plots of the participants' entry rates for four sub-blocks with 10 phrases each in both the with and with visual feedback blocks. Figure 9 shows that participants were quickly able to reach a reasonable entry rate and maintain this rate across the sub-blocks and across both blocks. This indicates that the immediate usability of the gesture keyboard is high, which is encouraging considering that users would both have to learn to interact with an AR headset and interact with a novel refinement of a gesture keyboard in the user study.

In general, the results show that users can use the gesture keyboard to enter words at an average rate of 20 WPM with a corrected error rate of less than 1%.

### 5.2.3 Study 2b: Extended Practice

Study 2b has a single objective: examine what performance we may observe from the fastest participants in Study 2a if they are provided an opportunity to further practice. We invited the ten fastest participants from Study 2a based on entry rate to return to participate in this study. Eight participants were willing to return for both this study as well as the later Study 2c.

We asked the participants to enter 40 phrases from one of the phrase sets in Study 2a. This process was repeated three times. Visual feedback was turned off and participants were instructed to not try to correct their errors or select any word alternatives. This allows us to study the performance of the AR gesture keyboard at an operating point where a more experienced user is entering text at a rapid pace with less regard to accuracy. Example use-cases include brief replies to emails or messages, or in-game communication.

Figure 10 shows the entry rate and uncorrected character error rate as a function of repetition of the phrase set. As expected, entry rates increased across repetitions while the uncorrected character error rate slightly increased. The mean entry rate and uncorrected character error rate of the final repetition was 35.70 WPM and 5.52% respectively. As a calibration point, in a large comparative study of commercial state-of-the-art touchscreen keyboards and gesture keyboards, Reyal et al. [31] found that the gesture keyboard resulted in an average entry rate of 30.6 WPM with a corrected character error rate of 2.0% in the final lab session. Overall, these results show that the AR gesture keyboard, and in particular its decoder, is capable of supporting entry rate and error rates roughly comparable to what is obtainable by a commercial touchscreen gesture keyboard.

#### 5.2.4 Study 2c: Accelerated Learning

Study 2c has a single objective as well: observe the human performance potential of the fastest participants from Study 2a.

Study 2c involved the same participants as in Study 2b. Study 2b and Study 2c were spaced at least four hours apart for each participant.

To assess the performance potential we applied an accelerated learning protocol, which used in the original large vocabulary gesture keyboard research paper [26]. The idea is to simulate expert performance by saturating motor learning. This can be achieved by repeating individual phrases until they are consolidated in motor memory. Participants are instructed to write a single phrase as quickly as possible. Participants are further instructed to not attempt any form of error correction. Each phrase is repeated 40 times followed by a 5-minute break.

The four phrases were: A) 'not at this time', B) 'i was planning to attend', C) 'i will probably page you', and D) 'need before board meeting'. The four phrases were randomly selected and cover a spectrum of perplexity.

The mean entry rate is 41.3 WPM while the mean uncorrected error rate is similar to Study 2b at 5.25%. Moreover, we also observed that participant 4 achieved an entry rate of 71 WPM on Phrase B and six of the eight participants achieved peak entry rates surpassing 50 WPM. We believe these estimates indicate the upper performance bound of what can be realistically achieved by this AR keyboard for well-rehearsed phrases [26].

### 5.3 Study 3: Performance Differential Between a 2D and 3D Gesture Keyboard Design

Study 3 consists of two investigations, which we will call Study 3a and 3b. Having teased out that a 3D trajectory-based decoding design can result in substantial entry rates, we want to understand if there is any subtle performance differential between a conventional gesture keyboard design for AR headsets compared to this new design when both designs use our new decoder.

Our testing and inspection of the Microsoft HoloLens 2 system keyboard as part of this project revealed that the system keyboard struggles to reliably detect the user's intent to trigger the gesture typing mode - the keyboard has both discrete (one character at a time) typing and gesture typing. There is also no exposed API for disabling the discrete entry mode. Regrettably, this prevented us from pursuing any comparative evaluations using the system keyboard as a baseline. As a result, we have created a baseline referred to as the 2D gesture keyboard design, which is derived from the Microsoft HoloLens 2 system keyboard. We made adaptations by eliminating the discrete entry mode and its corresponding visual indicator.

The 2D gesture keyboard design differs from our 3D gesture keyboard design in two critical ways.

- **Delimitation threshold**: The '3D' keyboard design incorporates a delimitation threshold set at 5 cm, indicating that a gesture delimitation event is triggered when the user's fingertip reaches or surpasses this distance. When a gesture delimitation event has activated, the user is able to gesture type and the 3D trajectory decoder decodes the trajectory that has been recorded within the delimitation period. In the '2D' keyboard design, it is 0 cm. Therefore, even a slight touch outside the virtual keyboard plane triggers delimitation in the '2D' gesture keyboard, leading to trajectories being falsely segmented into smaller ones (behavior seen on the Microsoft HoloLens 2).
- **Visual cursor**: In the '3D' keyboard design, the cursor follows the fingertip in 3D space, providing the user with direct control over the cursor. The cursor is also accompanied by a visual indication of its recent 3D path. In the '2D' keyboard design, the cursor is a regular projection of the fingertip's position onto the keyboard plane, and it is accompanied by a visual indication of its recent 2D path (visualization used on the Microsoft HoloLens 2).

Both designs have a cursor attached to the fingertip before delimitation which adaptively changes its size to indicate the relative perpendicular position between the fingertip and the virtual keyboard plane. This design also follows the design in the Microsoft HoloLens 2 system keyboard.

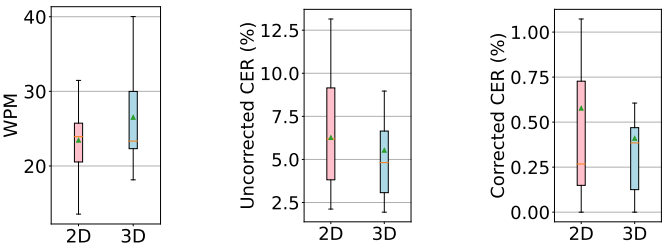



Fig. 11: Box plots depicting mean, median and quartiles of the participants' performance including entry rates and corrected/uncorrected character error rates for 40 phrases in Study 3a.

#### 5.3.1 Study 3a: User Performance for a 2D and 3D Gesture Keyboard Design

Study 3a compares these two designs. Both designs use the same decoder that we have introduced in this paper. In other words, we are not evaluating the efficacy of our decoder in Study 3a, we are comparing the merits of a relaxed 3D gesture typing design compared to a traditional rigid 2D gesture typing design.

Study 3a was structured as follows:

1. **Participants**: We recruited 16 participants using opportunity-sampling from our university campus. 10 were male and 6 were female and the average age was 23.4 (s.d. 3.2). None of the participants had previously taken part in Study 2a–2c.
2. **Materials**: We use the same stimulus phrase sets and same physical set up as in Study 2a.
3. **Protocol**: At the start of each design, participants were asked to write fifteen practice phrases that were not included in any of the two stimulus phrase sets. Participants were encouraged to ask questions and make sure they understood the interaction mechanisms and the keyboard functionality during practice. The order of the two designs and the order of the two stimulus phrase sets were counter-balanced. Having completed the practice stage, participants were asked to enter stimulus phrases as quickly and as accurately as possible. In addition, they were asked to slow down if their error rate was consistently above 10%. The keyboard displayed entry rate (WPM) and character error rate (CER) for each phrase to participants. Stimulus phrases were sampled randomly from the current phrase set. At the completion of the study, participants were asked to comment on their experience of using the 2D keyboard and the 3D keyboard design.

#### 5.3.2 Results

Figure 11 summarizes the performance of the two designs in terms of entry rate, uncorrected character error rate and corrected character error rate. Participants achieved a mean entry rate of 26.52 WPM with the 3D design compared a mean entry rate of 23.45 WPM with the traditional 2D keyboard, which is a 13.1% relative improvement in entry rate. A paired samples $t$-test shows that this difference is statistically significant ($t(15)$=2.20, $p$=0.044). We have used `scipy.stats` (v1.9.0) from the Python library and tested the assumptions for parametric test statistics. The differences in uncorrected ($t(15)$=-1.01, $p$=0.33) and corrected error rates ($t(15)$=-0.50, $p$=0.63) were small and not significant. In summary, the 3D design results in a significantly higher entry rate while maintaining a low error rate.

The qualitative feedback indicated an overall preference for the 3D design. Eight participants remarked that the relaxed delimitation provided by the 3D design appears to have helped them in avoiding false delimitation events when they accidentally gesture outside the keyboard plane. For example, *P3* commented that "When the word is long, I easily gesture out of the keyboard plane without actually noticing it, this leads to the whole word being decoded into two separate words. This sometimes is quite frustrating, so I need to slow myself down and pay great attention.". Nine participants also remarked that the 3D design

Table 4: Entry and error rate descriptive statistics for Study 3b (*mean* ± *SD* [*min*, *max*]).

| Block | Entry Rate (WPM) | Uncorrected Character Error Rate |
|---|---|---|
| 2D | 32.12 ± 8.61 [12.80, 50.18] | 1.73 ± 3.00 [0, 16.73] |
| 3D | 34.90 ± 10.01 [20.12, 68.24] | 1.43 ± 2.00 [0, 11.63] |

provides them with more control of the cursor. Three participants did not express a strong preference for either design.

#### 5.3.3 Study 3b: Accelerated Learning in a 2D and 3D Gesture Keyboard Design

Study 3a revealed that the 3D design resulted in significantly higher entry rates. The purpose of Study 3b is to investigate whether this entry rate difference is maintained under an accelerated learning protocol where participants are asked to repeatedly write a small number of phrases in order to achieve motor memory consolidation.

Study 3b was carried out with the same participants as Study 3a and followed the same accelerated learning protocol we used in Study 2c. Study 3b compared the 2D and 3D designs and the order of the two designs was counter-balanced.

Table 4 summarizes the entry rates and uncorrected error rates. The 3D design resulted in an approximately 8.7% higher entry rate (2.78 WPM difference). A paired samples $t$-test showed that this difference was statistically significant ($t(15) = 2.82$, $p = 0.00654$). There was no significant difference in uncorrected error rates ($t(15) = -1.20$, $p = 0.236$).

## 6 Discussion

We have presented a novel mid-air AR gesture keyboard decoder that directly translates users' 3D trajectories into text without relying on the trajectories projected onto the 2D keyboard plane. With the 3D keyboard design and our novel decoder, users can write without touching a virtual keyboard or articulating gestures within a precise plane. We also conjecture that a mid-air gesture keyboard without a visual cursor, combined with our decoder, would provide a user experience that is less coupled to finger tracking lag and inaccuracies, allowing for a more intuitive and natural interaction. This tendency towards open-loop interaction can benefit expert users who can quickly articulate imprecise word gesture trajectories directly from memory.

Study 1 used offline computational experiments to show that our system can obtain an uncorrected CER of 3.95%. Study 2a demonstrated that a sample of 16 participants could write at an average entry rate at 21 WPM with a corrected CER of 0.6% without any visual indication of the fingertip tracking cursor or any gesture trace on the virtual keyboard. Study 2b showed that the eight faster participants in Study 2a could with an hour of extended practice reach an entry rate of 36 WPM with an uncorrected CER of 5.5%, which is roughly equivalent to entry rates observed in the final session of a lab-based study of a commercial touchscreen gesture keyboard [31]. Study 2c used an accelerated learning protocol with the same participants as in Study 2b and demonstrated that the human performance potential of the mid-air gesture keyboard is in the region of 50–70 WPM with no errors for well-rehearsed phrases. These studies validate the robustness of the 3D trajectory-based decoder and demonstrate the potential of providing a 3D design for gesture keyboard AR usage.

Note that we do not claim the removal of visual feedback induces a higher entry rate or lower error rate. Since the blocks with and without visual feedback were in a fixed order we cannot infer any causal relationship between visual feedback and performance. Our claim is instead slightly subtle but nonetheless highly relevant: after some initial learning, users do not need such visual feedback to thrive using our mid-air AR keyboard, thus providing users with a seamless blending of physical reality—the user's index finger—with virtual content—the virtual keyboard plane.

The entry and error rates in Study 2a–2c indicate that this refinement of the AR mid-air gesture keyboard is practical and supports novice users reaching entry rates in a range between 18–21 WPM (Study 2a), with the possibility of reaching around 35 WPM with further practice (Study 2b). It is difficult to compare entry rates and error rates across different studies since the participant pools, parameters, stimuli and context are different [19]. However, Reyal et al. [31] carried out in-depth evaluations of a commercial touchscreen gesture keyboard in both a lab and on users' own mobile devices during a two-week field study. They found that users on average eventually reached entry rates ranging between 28 to 39 WPM with a corrected CER ranging between 1–3.6%. Taking their results as an indication, it is plausible that a long-term deployment of our AR mid-air keyboard would enable users to almost reach a similar performance, despite the substantial technical and behavioral challenges that arise when supporting mid-air gesture keyboard interaction.

Finally, Study 3a and 3b teased out the differences between a traditional 2D AR gesture keyboard compared to the new 3D design when both designs benefit from the new improved decoder. Study 3a found that the new 3D design resulted in a significant 13.1% relative improvement in entry rate and Study 3b found that an accelerated learning protocol comparison again yielded an 8.7% relative improvement. In both Study 3a and 3b there was no significant difference in error rate.

The robustness of our decoder enables accurate and reliable decoding of erratic and noisy trajectories, increasing the versatility and potential use cases of mid-air keyboards. The decoder's potential applications span various fields such as enabling hands-free text entry in VR, and video see-through augmented reality (VST AR). Kern et al. [16] conducted a comparison made between virtual reality (VR) and video see-through augmented reality (VST AR) in terms of tap and gesture/swipe keyboards. They discovered a significant increase in participants' swipe speed in VR compared to VST AR, and attributed this discrepancy to the visual incongruence experienced in VST AR. However, considering the novel decoding approach presented in this paper, it is conceivable that it could alleviate the variations in text input speed observed between VR and VST AR. In summary, the mid-air AR gesture keyboard decoder is a novel approach that offers improved user experience and versatile practical applications.

## 7 Future Work

We see several avenues of interesting future work. First, it would be fruitful to further study the full performance potential of our refinement of the mid-air AR keyboard and the effects of several parameters, such as the presence of various degrees of visual feedback and different interaction contexts (such as users walking or situated in outdoor areas with considerable background noise). Second, we believe a further refinement of the mid-air AR gesture keyboard is theoretically possible that would allow users to gesture entire sentences without hitting a spacebar or otherwise delimit different words in an interaction style reminiscent of "dwell-free eye-typing" [25] (which was later implemented as a working commercial product [23]). Realizing such a system for a mid-air AR keyboard would be a considerable technical challenge, however, we conjecture such a system would allow for a considerable performance improvement. Third, beyond text entry, another challenge is text editing [15,20], which opens up a large and relatively unexplored design space in mid-air text entry.

## 8 Conclusions

In this paper, we have presented a mid-air gesture keyboard for head-mounted augmented reality that supports users in articulating word patterns by merely moving their own physical index finger in relation to a virtual keyboard plane without any feedback of the tracked finger position. To realize this refinement of mid-air gesture keyboards it was necessary to tackle three problems: 1) the delimitation problem of identifying when users perform a gesture; 2) the alignment problem of inferring the plane in which the gesture performs their gesture; and 3)

the lack of representative data to train a deep neural network model. We addressed these challenges by introducing a novel decoding method for translating three-dimensional fingertip gestural trajectories into users' intended text which was further assisted by data augmentation.

Overall, this work has demonstrated that the original large vocabulary gesture keyboard design [26] can be successfully transplanted from tablets and mobile phones to AR headsets using a new approach based on 3D trajectory decoding and a design that takes into account the unique challenges posed by AR mid-air interaction. We hope this work will inspire further work in this area, including supporting a rich set of text editing operations and studying 3D trajectory decoding gesture keyboards in a variety of different settings.

## Open Science

Complete source code for the neural motion model can be found here: `https://github.com/CambridgeIIS/mid_air_gesture_keyboard_decoder`.

## References


[1] J. Adhikary and K. Vertanen. Text entry in virtual environments using speech and a midair keyboard. *IEEE Transactions on Visualization and Computer Graphics*, 27(5):2648–2658, 2021. doi: 10.1109/TVCG.2021.3067776

[2] 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.

[3] X. Bi, C. Chelba, T. Ouyang, K. Partridge, and S. Zhai. Bimanual gesture keyboard. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, p. 137–146. Association for Computing Machinery, New York, NY, USA, 2012. doi: 10.1145/2380116.2380136

[4] E. Biju, A. Sriram, M. M. Khapra, and P. Kumar. Joint transformer/RNN architecture for gesture typing in indic languages. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 999–1010. International Committee on Computational Linguistics, Barcelona, Spain (Online), Dec. 2020. doi: 10.18653/v1/2020.coling-main.87

[5] S. J. Castellucci and I. MacKenzie. Gathering text entry metrics on android devices. In *CHI EA '11*, 2011.

[6] W. Chan, N. Jaitly, Q. Le, and O. Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4960–4964. IEEE, 2016.

[7] N. Dahlbäck, A. Jönsson, and L. Ahrenberg. Wizard of oz studies: why and how. In *Proceedings of the 1st international conference on Intelligent user interfaces*, pp. 193–200, 1993.

[8] L. Dong, S. Xu, and B. Xu. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5884–5888. IEEE, 2018.

[9] 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. IEEE, 2019.

[10] 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.

[11] 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), Dec. 2018.

[12] M. Gordon, T. Ouyang, and S. Zhai. *WatchWriter: Tap and Gesture Typing on a Smartwatch Miniature Keyboard with Statistical Decoding*, p. 3817–3821. Association for Computing Machinery, New York, NY, USA, 2016.

[13] A. Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

[14] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, p. 369–376. Association for Computing Machinery, New York, NY, USA, 2006.

[15] J. Hu, J. J. Dudley, and P. O. Kristensson. An evaluation of caret navigation methods for text editing in augmented reality. In *2022 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, pp. 640–645. IEEE, 2022.

[16] F. Kern, F. Niebling, and M. E. Latoschik. Text input for non-stationary xr workspaces: Investigating tap and word-gesture keyboards in virtual and augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 29(5):2658–2669, 2023.

[17] B. Klimt and Y. Yang. The enron corpus: A new dataset for email classification research. In J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, eds., *Machine Learning: ECML 2004*, pp. 217–226. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[18] P. Knierim, V. Schwind, A. M. Feit, F. Nieuwenhuizen, and N. Henze. Physical keyboards in virtual reality: Analysis of typing performance and effects of avatar hands. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, p. 1–9. Association for Computing Machinery, New York, NY, USA, 2018.

[19] P. O. Kristensson. *Discrete and Continuous Shape Writing for Text Entry and Control*. PhD thesis, 2007.

[20] P. O. Kristensson. Five challenges for intelligent text entry methods. *AI Magazine*, 30(4):85–85, 2009.

[21] P. O. Kristensson. Next-generation text entry. *Computer*, 48(07):84–87, 2015.

[22] P. O. Kristensson, J. Lilley, R. Black, and A. Waller. A design engineering approach for quantitatively exploring context-aware sentence retrieval for nonspeaking individuals with motor disabilities. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–11, 2020.

[23] P. O. Kristensson, M. Mjelde, and K. Vertanen. Understanding adoption barriers to dwell-free eye-typing: Design implications from a qualitative deployment study and computational simulations. In *Proceedings of the 28th International Conference on Intelligent User Interfaces*, pp. 607–620, 2023.

[24] P. O. Kristensson and T. Müllners. Design and analysis of intelligent text entry systems with function structure models and envelope analysis. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pp. 1–12, 2021.

[25] P. O. Kristensson and K. Vertanen. The potential of dwell-free eye-typing for fast assistive gaze communication. In *Proceedings of the symposium on eye tracking research and applications*, pp. 241–244, 2012.

[26] P. O. Kristensson and S. Zhai. SHARK$^2$:a large vocabulary shorthand writing system for pen-based computers. *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, 01 2004.

[27] L. A. Leiva, S. Kim, W. Cui, X. Bi, and A. Oulasvirta. *How We Swipe: A Large-Scale Shape-Writing Dataset and Empirical Findings*. Association for Computing Machinery, New York, NY, USA, 2021.

[28] 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*, CHI EA '03, p. 754–755. Association for Computing Machinery, New York, NY, USA, 2003.

[29] 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.

[30] A. Mehra, J. R. Bellegarda, O. Bapat, P. Lal, and X. Wang. Leveraging gans to improve continuous path keyboard input models. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8174–8178. IEEE, 2020.

[31] 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.

[32] M. Richardson, M. Durasoff, and R. Wang. *Decoding Surface Touch Typing from Hand-Tracking*, p. 686–696. Association for Computing Machinery, New York, NY, USA, 2020.

[33] J. Shen, J. Dudley, and P. O. Kristensson. The imaginative generative adversarial network: Automatic data augmentation for dynamic skeleton-based hand gesture and human action recognition. In *2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021)*, pp. 1–8. IEEE, 2021.

[34] 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.

[35] J. Shen, J. Dudley, G. Mo, and P. O. Kristensson. Gesture spotter: A rapid prototyping tool for key gesture spotting in virtual and augmented reality applications. *IEEE Transactions on Visualization and Computer Graphics*, 28(11):3618–3628, 2022.

[36] 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.

[37] K. Vertanen and P. O. Kristensson. Mining, analyzing, and modeling text written on mobile devices. *Natural Language Engineering*, 27(1):1–33, 2021.

[38] W. Xu, H. 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, 2019.

[39] X. Yi, C. Liang, H. Chen, J. Song, C. Yu, H. Li, and Y. Shi. From 2d to 3d: Facilitating single-finger mid-air typing on qwerty keyboards with probabilistic touch modeling. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 7(1):1–25, 2023.

[40] 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, p. 4479–4488. Association for Computing Machinery, New York, NY, USA, 2017.

[41] 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*, p. 4479–4488. Association for Computing Machinery, New York, NY, USA, 2017.

[42] S. Zhai and P. Kristensson. The word-gesture keyboard: Reimagining keyboard interaction. *Communications of The ACM - CACM*, 55, 09 2012.

[43] S. Zhai and P.-O. Kristensson. Shorthand writing on stylus keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, p. 97–104. Association for Computing Machinery, New York, NY, USA, 2003.

[44] S. Zhai and P. O. Kristensson. The word-gesture keyboard: Reimagining keyboard interaction. *Commun. ACM*, 55(9):91–101, Sept. 2012.

[45] S. Zhai, P. O. Kristensson, P. Gong, M. Greiner, S. A. Peng, L. M. Liu, and A. Dunnigan. Shapewriter on the iphone: from the laboratory to the real world. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems*, pp. 2667–2670. 2009.