Thumb Gesture: A Novel Bimanual Text Entry Method in Virtual Reality

Jiaming Tian[1,2]*, Minghui Sun*[1,3*]*, Zigang Zhang*[1,2]

[1] *College of Computer Science and Technology, Jilin University, China*
[2] *The Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, China*
[3] *Research Institute of Jilin University in Shenzhen, China*
1836931234@qq.com, smh@jlu.edu.cn, 42600225@qq.com

Abstract

Text input is an important task in the VR environment, but it is still inefficient and difficult. We propose a new technique named 2-Thumbs Typing (2TT) enabling text entry with a touchpad in HTC VIVE controller to support two-handed input and make text entry easy and efficient. Users can touch a controller to start the input while finishing by releasing both thumbs. The technique can recognize the gestures users draw in the two touchpads. We first design a set of ambiguous 2-thumbs unistroke gestures and improve them to finish the final design by a user study to between memory, performance efficiency and ease of use. In our study, we conclude that the 2TT technique is an efficient text entry method in VR environments. No additional equipment is required and 2TT method supports eyes-free entry. 2TT can reach 8.5 words per minute with extensive training. In addition, subjective feedback from users shows that 2TT is easy to learn and use, with less fatigue than one-hand gesture text entry and controller pointing text entry. The experimental results have reference significance for the text entry design based on the input mode of two-handed selection in VR.

Keywords: Virtual reality, Text input, Unistroke gestures, Bimanual input

1 Introduction

In recent years, the popularity of virtual reality (VR) has risen sharply. However, text-based communication rarely studies the performance of text input in VR research. It is little known about the user preferences and characteristics of VR to understand the latest technology and interactive concepts for text input in VR. The common text input method is inefficient and difficult in the VR environments. It only supports one-handed interaction which has operational restrictions and cannot be used widely [1-2]. We need to evaluate the user preferences and characteristics of VR to understand the latest technology and interactive concepts for text input in VR. However, text entry is not efficient and difficult in the VR environment [3-4]. For example, laser aiming has many drawbacks, such as high fatigue, jitter, failure prone when user pulls trigger [3]. Other attempts, for example, they proposed speech [5-7] and mid-air typing [8-10] to improve text entry in VR. However, speech technology is not suitable for public use [11-12] and additional equipment (such as a camera or gloves) is required for mid-air typing technology. These shortcomings restrict users to certain body postures and positions. The basic question is how to improve the interaction experience, and accuracy and speed of text entry in VR environments became a problem in Human-computer interaction.

To solve this problem, we propose considering the characteristics of two-hand interaction and a novel design method 2-Thumbs Typing (2TT) is presented to answer the question, supporting efficient and accurate input in the VR environment (Figure 1).

The user draws a vertical line with his left hand while drawing a polyline with his right hand to close to the shape of a lowercase “b”. When we release our thumbs, it means that we have entered the letter b. 2TT’s gesture recognition method only considers the two-handed gestures’ direction, without considering the shape of the gesture. We want to overcome this limitation by using two-hand gestures. Users can simply draw simple gestures and enter the letters they want. First, the input method should be easy to learn. It is necessary to design bimanual gestures supporting eyes-free and requiring no additional device. In addition, the other flexibility or benefits brought by two-handed gestures compared to one-handed gestures should not be at the expense of great fatigue [13].

The biggest challenge of our technique is that several letters may be difficult to be support bimanual gestures text input. Our target is to obtain a set of gestures that balances memory, input accuracy, and input speed. After careful design, the gestures include a set of simplified unistroke strokes based on two-hand assisted strokes. And we conducted a user study to verify the performance of the design and improve it to complete our final design. In the design of gestures, we try to avoid asymmetric two-handed gestures and improve the comfort of input.

To implement our 2TT technology, we designed a gesture recognition algorithm that classifies input strokes in a probabilistic manner. The system realizes a novel form of two-handed gesture input, extending the common one-

*Corresponding Author: Minghui Sun; E-mail: smh@jlu.edu.cn

DOI: https://doi.org/10.70003/160792642025012601004

handed gesture text input from one hand to two hands. Second, we conducted formal experiments to evaluate the advantages and disadvantages of this two-handed gesture input technology. The results showed that with extensive training, the 2TT can reach 8.5 WPM, which indicated that it is easy to learn. Compared with one-handed, it increases comfort and reduces physical demands. In most cases, it outperforms one-handed gestures in performance (Figure 8).

The main contributions of the 2TT are as follows:

1) Eyeless operation is supported, the controller does not need to be continuously raised, and gesture input can be made in any posture to reduce fatigue. Unistrokes could be significantly faster with 11.4 WPM. Unistroke letters can also be used for eyeless input on mobile devices.

2) The method is easy to learn and we simulate handwriting as much as possible when designing gestures, so the subjective evaluation and acceptance of this technology by users are high in the user research.

3) A new dedicated algorithm is designed for stroke recognition, which accuracy rate is more than 90%, and the speed is also ahead of the current general technology.

In the rest of this article, we briefly review related work at first. Then, we pay attention to the main two-hand gesture design, development, and performance evaluation experiments. Finally, we conclude the study by discussing limitations and future work. Therefore, the main contributions of this article are as follows:

1. Introducing an efficient technique called 2-Thumbs Typing enabling eyes-free without extra equipment text entry in VR.

2. Designing a gesture recognition algorithm that recognizes the set of simplified bimanual handwriting gestures.

3. Evaluating the 2TT text input method in empirical research on text input performance and user preferences.

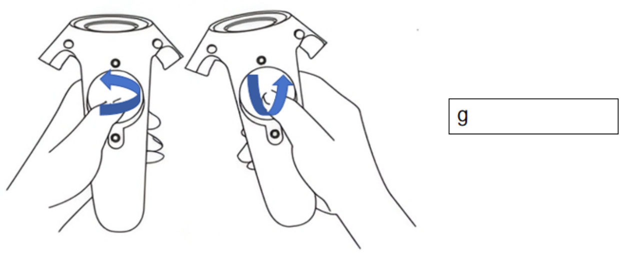


Figure 1. 2TT: Text entry environment

2 Related Work

2.1 Conditioning Event Rep

Kristensson and Zhai [14] proposed gesture typing which widely adopted worldwide and expanded to a variety of input methods. Bi et al. [13] created a two-way gesture keyboard that allows multiple taps with both hands to enter a word. Markussen et al. [8-10] investigated the air gesture input. Their text input system called Vulture projects the user's hand movements onto the display and pinch is used as a word delimiter. Their research shows that after 10 phases of research, users can reach 20.6 WPM. Yu et al. [12] explored the use of head movement to perform gesture typing.

Unistroke letters were put forward for text input in 1993 [15] was designed to write. Results showed that EdgeWrite [16] was originally proposed by Wobbbrock, aiming to achieve single pen input while providing individuals with high precision and motion stability. This method fixes a square hole on the touch screen, where the user can enter text across edges and diagonals. Gesture recognition performs according to the order in which corners are hit. EdgeWrite gesture letters are designed for quick learning. Edgewrite can also be applied to other input devices, such as touchpads [17-18], trackballs [19] and space keys [20].

On the basis of existing gesture input technology, our research explores how to support gesture text input by two hands in VR environments and how to recognize the design simplified gestures accurately.

2.2 Text Input in VR Environments

The current VR text input is mainly divided into two categories, the first is to use other devices with the corresponding software, the second method is to only make changes in the software with the traditional VR peripheral handle.

Pizza Text used the first method, which used the X-box gamepad for text input [5]. The average speed reached 8.59WPM, and those who were proficient in this input mode can even achieve ultra-high speed of 15.86WPM. But this method had shortcomings: hardware changes bring about cost increases and hold one hand in VR environment. The handle was not as convenient as holding the handle with both hands. Some researchers use a smart phone as an additional indirect input device, such as HoVR-Type [21]. However, it was not conducive to the user's immersive experience. Other methods such as Vitty [22] and Tap System [23] were unsuited for long period of time and a lot of text input.

Speech technology played an important role in the user interface because of its simplicity and efficiency. Despite this performance, speech recognition had serious limitations in terms of environmental noise sensitivity, privacy, and outstanding performance in the same environment. [11].

HotStrokes [24] used machine learning algorithms to select by dashing, but users often use fixed words instead of fixed words when setting passwords. Combination, and the algorithm vocabulary is not perfect, in some applications can achieve quite good performance, but there were still some distances into the mainstream text input method, but fully demonstrates the infinite possibilities of human-computer interaction in VR environment.

Our research focuses on designing a novel text input method in VR environment without adding extra device and hardware, which requires the least learning time and relatively little space for practice is required to minimize fatigue. Considering people always stare at somewhere else instead of VR controllers, the method should be eyes-free to improve the comfort.

2.3 Bimanual Interaction

Desktop-based bimanual interaction technology improved performance and accuracy [25-27], and was more convenient when performing demanding cognitive tasks [28-29]. Some techniques provided symmetry control [30]. For example, Symspline gave the same role to both hands when the user manipulates the curve [31]. However, most two-way interaction techniques were based on Guiard's kinematic chain model [32], based on his observation of the asymmetric relationship between the two hands [25]. For example, tool glasses, magic lenses and two-way palettes [26, 33-34] used non-dominant hands to control the position of interactive palettes and use dominant hands to select specific functions. Fixed multi-touch surfaces, multi-touch tables, and graphics tablets are inherently very suitable for two-way interaction, because users can use multiple fingers from one or two hands. Studies showed that two-way interactive technology can improve performance [35-36] and selection accuracy [37]. However, there is little research applying bimanual interaction into text input in VR environment.

3 Two Thumbs Text Input in VR

We design two thumbs gesture typing to solve the problem of inefficient text input in VR. Our innovation lies in using bimanual symmetric interaction in VR text input to break through the limitations of one-hand interaction and improve the comfort and efficiency of interaction.

We started typing by touching the touchpad keys on the HTC VIVE controller with both hands at the same time and ended typing when both hands left the touchpad keys. In the process, we don't need to see the specific buttons and patterns, but only need to focus on drawing the corresponding gestures. We make full use of the symmetry characteristics of two-hand interaction and simplify gestures as much as possible to maximize the efficiency of two-hand interaction. 2TT can make the input speed of two-handed interaction faster than simple one-handed handwriting.

We strive to achieve a balance between input efficiency and ease of learning. We also follow the principle of symmetry in the design of bimanual gestures to achieve the best interaction.

In order to complete the two-hand gesture input technology in VR, we divided the research into two parts. The first study designed simple and easy-to-understand hand gestures, and the second study evaluated the performance of text input to prove the contribution of our research.

We designed and implemented a new algorithm for gesture recognition and used chain code detection algorithm to improve the accuracy of the results. Finally, our recognition algorithm can achieve a high accuracy rate.

4 Design of Bimanual Handwriting Gestures

The purpose is to understand the user's acceptance of handwritten gestures and the input capacity of bimanual gestures text input of HTC Vive, the electronic device we used to test the idea of this research. We first designed a set of two-way handwritten stroke gestures, and then asked each participant to evaluate factors such as comfort by performing gestures on HTC Vive. According to the results, we resumed the design guidelines for handwritten gestures with both hands and revised the design according to the guidelines.

4.1 Phase 1 Initial Design of 2-Thumb Gestures

We adopt a user-participatory approach in our first trial. We recruited four participants and asked them to design a set of bimanual gestures for each letter on the touchpad. All of the gestures should be finished with two thumbs and as simple as possible. Finally, we proposed a handwriting-based design while four participants regarded it as a memorable and simple set of gestures. The recognize process ignored the shape of bimanual gestures but only focused on the trajectory. The design simplified the handwriting gestures and decreased the length we drew in a touchpad.

Figure 2 summarizes the bimanual gestures. For most letters, the design is simple and intuitive. We just need to write the approximate trajectory of a lower-case letter to input it. However, for the other people, this is not easy, because some letters are similar, even if not the same, when divided into two parts. In such cases, we designed different hands or special strokes to distinguish them.

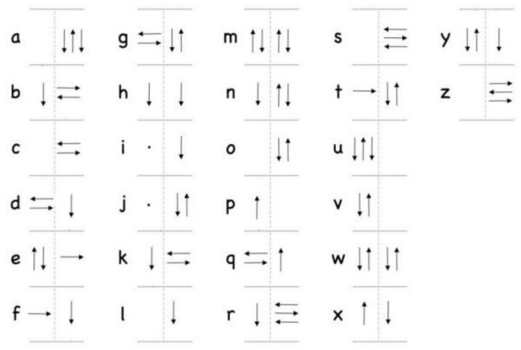


Figure 2. Initial 2TT design

4.2 Phase 2 User Assessment and Data Collection

4.2.1 Apparatus and Setup

The VR system uses HTC Vive and runs on a Windows 10 machine with Unity 2019. A standard desktop computer uses i9 CPU, 32 GB RAM, and Nvidia GeForce GTX 2080Ti graphics card to fill out the questionnaire and control the experiment. The HTC Vive optical tracker (or lighthouse) is installed at two opposite corners about 2m above the ground. Participants stand in the center while performing tasks.

4.2.2 Design and Procedure

We recruit 8 participants from Jilin University campus. The experiment takes each person about an hour to complete. First, each participant was demanded to remember the patterns of the two-hand gesture set. They will then perform five gestures on each letter and make sure they feel comfortable.

To help them, we placed a table containing each letter-gesture pair next to the text input field in the VR environment, in case they forget the gesture during the experiment. Participants also rated the memorability of each gesture on a 5-point scale. We collected a total of 1560 handwritten gestures. In order to collect experimental data (Figure 3), we developed software that records all touch events (including time and coordinate information).

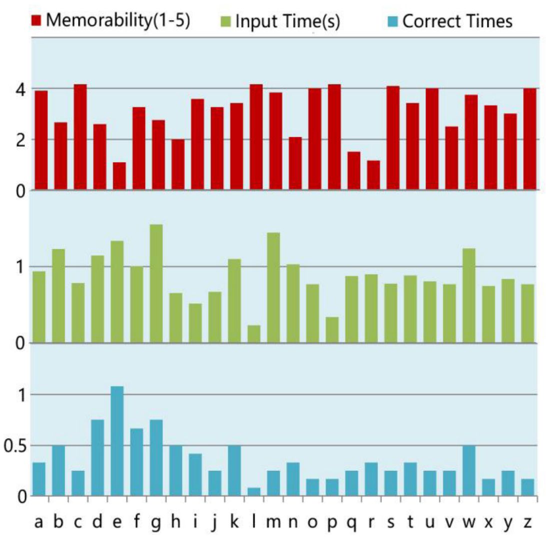


Figure 3. Input time, correct times, path length and memorability for different letters

4.2.3 Results

Error Rate. The times of participants correct the input indicates the input accuracy of the method. Most of the gestures don't need a lot of times to correct the input while separate letters have high error rate. We decided to improve them in the final design.

Input Speed. For letters that take too long to input, it may be due to the asymmetry of the bimanual gestures, or the gesture design is too complicated. The fact that participants are constantly referencing gesture forms also confirms this difficulty. So, we redesigned the alphabet gestures that took too long to type.

Memorability. Although most of the gesture resembles the handwriting, the diversion of some letter's gestures seemed to be more difficult to remember. In addition, participants' subjective assessment of gesture memorability also varied.

4.3 Phase 3 Final Design of Handwriting Gestures

The design goal is to achieve a comprehensive balance between speed, accuracy, ease of learning and memory. We collected the suggestions and redesigned all the gestures rating lower than two points. Gestures will be redesigned if the input time of them significantly higher than the average. Then, we conducted a micro experiment to evaluate the performance of the 2TT gestures. In the micro experiment, participants entered a phrase selected from Mackenzie's and Sokolev's phrasesets. Participants determine by clicking the trigger on the controller and clicking the trigger after completing the phrase. Participants can correct the current word by backspace key. Subjective measurements are collected at the end of the experiment. Participants rated each input method on an overall preference on a scale of 1 to 10. Based on the above results and with the help of 8 participants, according to the information collected, we established the following gesture design guidelines and completed the final design of 2TT.

(1) Imitate traditional handwriting: 2TT gestures ought to be easy to learn. By imitating traditional handwriting, the resulting strokes should be easy to remember.

(2) Minimize the number of strokes per letter: To perform stroke gestures efficiently, the strokes of each hand should be as few as possible.

(3) Bimanual gestures should be as symmetrical as possible, or complex gestures favor the dominant hand.

Figure 4 shows the final stroke design of each letter. We redesigned the gesture of letter 'e' and exchange the 'h' and 'n' according to the participants' suggestions. We simplified some gestures (e.g., "q" "r" etc.) because of the low points achieved in the experiment. In addition, we made some minor adjustments, such as changing the gesture from the non-primary hand to the primary hand and using the non-primary hand to simplify the gesture. We also adjusted some direction of certain gestures like 'f' to draw them more comfortable. Finally, we revisited all asymmetric gestures and change most of them to comfort human writing habits.

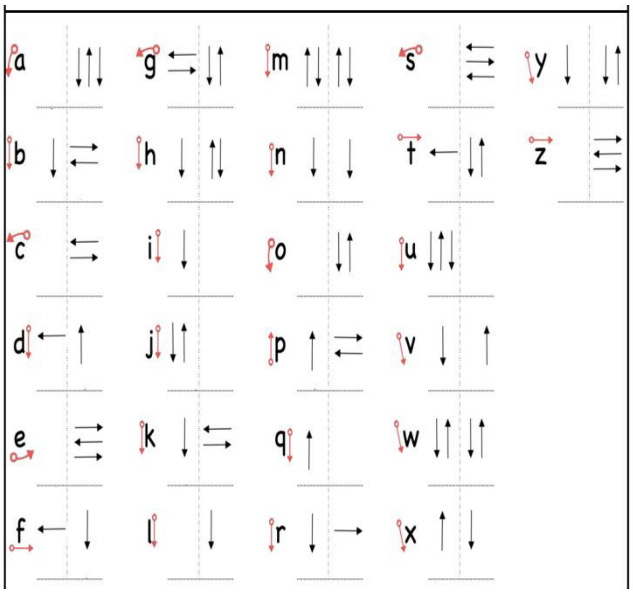


Figure 4. Final design for individual characters (Red arrows represent the first stroke in the gestures while black arrows represent the gestures in the recognition algorithm)

5 Technique

In this section, we describe the implementation details of the recognition and interaction technology for handwritten gestures. We introduce the algorithms used to recognize gestures and how to use 2TT to input letters at first. We also describe the interaction design, which is used to input individual characters and how to conduct our evaluation experiments.

5.1 Gesture Recognition

The recognition algorithm is used to estimate the posterior probability of gesture input given a gesture template, which has a number of steps.

5.1.1 Step #1 Get Gesture Coordinates to Build up Sequences

We get some gesture coordinates with the data structure of points each frame (30 frames one second) by the two HTC Vive controllers' touchpads. We stored them in the two-dimensional arrays ρ to prepare for the next steps.

5.1.2 Step #2 Import Gesture Template

We import 5 gesture templates for each letter to recognition algorithm resources for the recognition. We aimed to find the most resembled gesture between the input sequences and the template. We sort the results by the recognize points result and choose the highest point as the recognize result.

5.1.3 Step #3 Resampled Point

We use the algorithm to resample the points we get using the HTC Vive touchpad. We input all the points from the user touching action and the algorithm return the stable 64 points to describe the trace of user gesture.

Algorithm 1. Resample points

Input: *Points*, *PointsCount*

Output: *resampledPoints*

```
1: for $i = 1 \rightarrow PointsCount$ do
2:     $length \leftarrow length + Points[i] - Points[i-1]$
3: end for
4: $I \leftarrow length/PointsCount$
5: $D \leftarrow 0$
6: for $i = 1 \rightarrow PointsCount$ do
7:     $d \leftarrow Points[i] - Points[i-1]$
8:     if $D + d >= I$ then
9:         $x \leftarrow Points[i-1].x + (I - D)/d * (Points[i].x - Points[i-1].x)$
10:        $y \leftarrow Points[i-1].y + (I - D)/d * (Points[i].y - Points[i-1].y)$
11:        $q \leftarrow Vector2(x, y)$
12:        $resampledPoints.Add(q)$
13:        $Points.Insert(i, q)$
14:        $D \leftarrow 0$
15:    else
16:        $D \leftarrow D + d$
17:    end if
18: end for
19: return $resampledPoints$
```

5.1.4 Step#4: Rotate, Scale and Translate

We use the algorithm to transform the points we have resampled. After we rotated, scaled and translated the points, we can compare them with the templates we imported to get the correct gesture we input.

Algorithm 2. Rotate scale translate

Input: *Points*, *angle*, *PointsCount*, *size*

Output: *Points*

```
1: function ROTATEBY($Points, angle, PointsCount$)
2:     for $i = 0 \rightarrow PointsCount - 1$ do
3:         $center \leftarrow center + Points[i]$
4:     end for
5:     $center \leftarrow center/PointsCount$
6:     $cos \leftarrow cos(angle)$
7:     $cos \leftarrow sin(angle)$
8:     for $i = 0 \rightarrow PointsCount - 1$ do
9:         $x \leftarrow Points[i].x - center.x * cos - (Points[i].y - center.y) * sin + center.x)$
10:        $y \leftarrow Points[i].x - center.x * cos - (Points[i].y - center.y) * sin + center.x)$
11:        $Points.Add(Vector(x, y))$
12:    end for
13:    return $Points$
14: end function
15:
16: function SCALETO($Points, size, PointsCount$)
17:     $minX \leftarrow min(Points.x)$
18:     $minY \leftarrow min(Points.y)$
19:     $maxX \leftarrow max(Points.y)$
20:     $maxY \leftarrow max(Points.y)$
21:     $boundingBox \leftarrow Rect(minX, minY, maxX - minX, minY - minY)$
22:     for $i = 0 \rightarrow PointsCount - 1$ do
23:         $x \leftarrow Points[i].x * size/boundingBox.width$
24:         $y \leftarrow Points[i].y * size/boundingBox.height$
25:         $Points.Add(Vector(x, y))$
26:     end for
27: end function
28:
29: function TRANSLATETO($Points, size, PointsCount$)
30:     for $i = 0 \rightarrow PointsCount - 1$ do
31:         $center \leftarrow center + Points[i]$
32:     end for
33:     $center \leftarrow center/PointsCount$
34:     for $i = 0 \rightarrow PointsCount - 1$ do
35:         $x \leftarrow Points[i].x - center.x$
36:         $y \leftarrow Points[i].y - center.y$
37:         $Points.Add(Vector(x, y))$
38:     end for
39: end function
```

5.1.5 Step #5: Get Optimal Cosine Distance

After we transform the points by the fourth step, we can get the optimal cosine distance between the points sequence we touched in the touchpad and the points sequence of the preset gestures. The recognize result is the highest result of the preset gestures.

Algorithm 3. Get optimal cosine distance

```
Input: Points, PointsCount, LibraryVector, VectorCount,
Output: result
function VECTORIZE(Points, PointsCount)
  $sum \leftarrow 0$
  for $i = 0 \rightarrow PointsCount - 1$ do
      $vector.Add(Points[i].x)$
      $vector.Add(Points[i].y)$
      $sum \leftarrow sum + \sqrt{Points[i].x} + \sqrt{Points[i].y}$
  end for
  $magnitude \leftarrow \sqrt{sum}$
  for $i = 0 \rightarrow PointsCount - 1$ do
      $vector[i] \leftarrow vector[i]/magnitude$
  end for
  return vector
end function

function Get Optimal Cosine Distance(vector, LibraryVector, VectorCount)
    $a \leftarrow 0$
    $b \leftarrow 0$
    for $i = 0 \rightarrow VectorCount - 1$ do
        $i \leftarrow i + 2$
        $a \leftarrow LibraryVector[i] * vector[i] + LibraryVector[i+1] * vector[i+1]$
        $b \leftarrow LibraryVector[i] * vector[i+1] - LibraryVector[i+1] * vector[i]$
    end for
    $angle \leftarrow tan(b/a)$
    $result \leftarrow 1/(arccos(a * cos(angle) + b * sin(angle)))$
    return result
end function
```

5.1.6 Step #6: Chain Codes Detection Algorithm

In order to improve the accuracy of the results, we use chain code detection algorithm to re-detect the direction of the gesture while considering only the direction, and the result will be output only if the direction matches the template.

By comparing the proportion of each segment with the length of the gesture, the main characteristics of the gesture or the main movement direction can be analyzed.

The straight-line recognition process using the chain code algorithm is as follows:

The absolute chain code refers to the cumulative value of the relative chain code from the starting point. Set the chain code to 0. Move along the boundary to return to the starting point, and its absolute chain code value increases by 8.

The calculation of the absolute chain code is as follows:

When the number of boundary points is N, the sequence numbers of the boundary points are 0 ~ N-1. From the starting point,

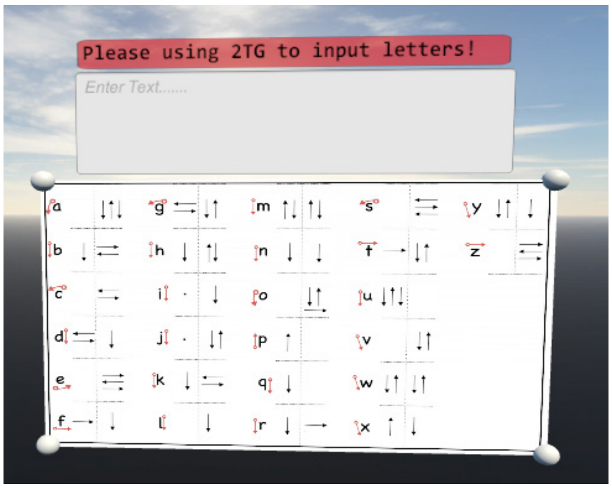
The difference between the absolute chain codes when going around the boundary counterclockwise to return to the starting point is 8. The sum of the three-point chain code is the sum of the absolute chain code of the current point and the previous two points. Because the boundary is closed, the value of the previous point must be rounded to the tail when the first two points are calculated. The chain code difference represents the difference between the two directions. It is a quantity proportional to the curvature and can be used to find the corner points on the boundary. When the boundary direction is counterclockwise, points with positive difference are convex points, and points with negative difference are concave points. Finally, the absolute chain code value C is obtained, and C is compared with a preset template. The closest value is the recognition result. According to my micro-contrast experiment, the chain code is used in combination with the coordinates, and the recognition accuracy can be further improved. The average accuracy of 500 sets of data has increased from 87.3% to 88.7%.

5.2 Interaction Design & implementations

As the Figure 5 shows, we created a text field in the scene, and we could see the text we have already input. Besides, we design a grey line to make the handwriting procedure visible. To cancel the handwriting procedure, just stop gesturing for a while. The system will identity the stop of hand action so that cancels this letter's input method.

We used finger-release to end the input process. When we pressed up the touchpad button, the gesture recognition algorithm will work and output the result on the screen. The design is supposed to be simple and easy to learn. The function of Grip is to enter space and the trigger button's function is to back space. With all of them, we can easily and fast finish an article's text.

To help we conduct the experiment better, the aim text and finish sign displayed above the input field, we can easily get the text information we need to input.



Please using 2TG to input letters!

Enter Text......

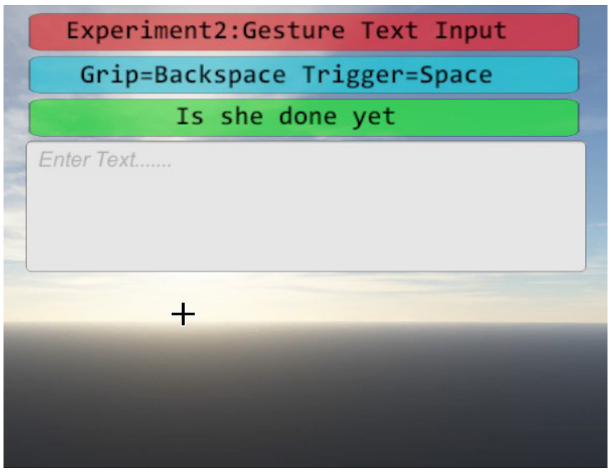
Figure 5. Interaction design

6 Evaluating Text Entry Performance

To evaluate the performance of 2TT when letters entered in VR environment, we compared its performance with the most popular method in the VR environments [3] and a formal experiment was conducted to compare the gestures of 2-thumb with traditional one hand handwriting.

6.1 Participants and Apparatus

12 participants (aged from 18-25) were recruited from Jilin University. Figure 6 shows the experimental environment and settings. The system displays phrases that need to be input by participants in the virtual scene. They wore the Vive helmet and hold the controllers. The software on Unity recorded and synchronized all touch events, coordinates and time during the experiment, including the correct event.


Experiment2:Gesture Text Input

Grip=Backspace Trigger=Space

Is she done yet

Enter Text.......

Figure 6. The experiment setting: controller pointing and gesture text input (2TT and unimanual handwriting - Red panel shows the text entry method using in this experiment and green panel shows the target sentence for participants to input.)

6.2 Design and Procedure

During the experiment, the only independent factor is the technique used, we completed the input using three techniques: 2TT text input, unimanual handwriting (UH), and controller pointing (CP).

The test phrases for each participant were randomly generated from the phrase set of K. Vertanen and P. O. Kristensson [38]. We use the same phrases for each participant's three techniques, and the order of the phrases is random. The experiment consists of 5 parts and they need to have a break after finishing a part: input 2 phrases. The first part is ignored for practice. Each participant finished 2*5*3 times phrases input task.

Before the experiment, we introduced our goals to each participant, but we don't introduce which technique is our design. During the experiment, Participants need to be as accurate and quick as possible when entering phrases. Finally, we evaluate NASA-TLX [39] to provide feedback on the three techniques. We collect various subjective feedback to evaluate user experience and workload, which is very important in VR applications. The total experiment last for roughly 1 hours.

The learning time of 2TT is a bit longer than the time the controller points to. For two-handed gestures, we placed a worksheet containing each letter gesture of the participants in the VR environment. Next, we asked participants to remember and practice these gestures. Then, ask them to do a short exercise to enter all 26 letters twice in random order. After they became familiar with touch motion control and deletion, they began to enter phrases. The practice of both techniques lasts about 30 minutes.

6.3 Results

According to the results and the following discussion, we use abbreviations and color indications for the three text input methods tested. We use IBM SPSS Statistics 20 to analyze the result.

6.3.1 Input Speed

We choose words per minute (WPM) to measure speed, WPM is also used as the current standard for text input evaluation [40]. Formally: $WPM = (|T|-1)/S \times 60 \times 1/5$.Where S is the time (in seconds) from the first to the last key press and |T| is the number of characters in the transcribed text.

Figure 7 shows the WPM for three technologies on ten blocks. We calculated WPM included the time participants spent on the correcting process. On average, participants input at 6.35 WPM (SD=1.68) with 2TT, 7.29 WPM (SD=0.55) with UH, and 10.05WPM (SD=10.3) with CP. However, participants input at 8.20 WPM with 2TT and 7.25 with UH,11.09 with CP if we considered only the last block. for 2TT technology, we observed the significant impact of blocks on WPM ($F\_2,22=5.438$, $p=0.001<0.01$), which suggested the learning effect. We can conclude that 2TT was faster than UH after training for a period of time. As shown in Figure 7, block has a great learning effect on the text input rate. On average, participants using the 2TT method entered the last block 172.42% faster than the first block.

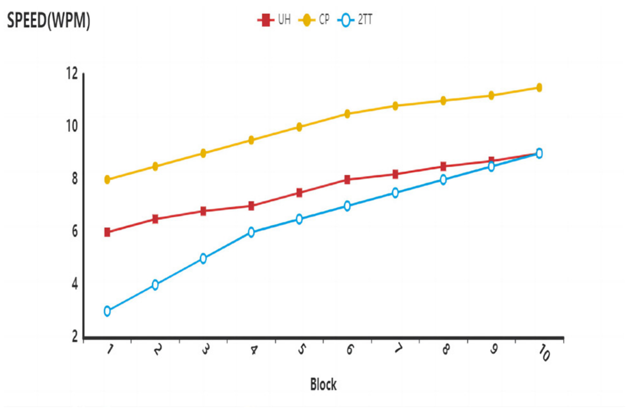


Figure 7. WPM of three text entry methods

According to participants' subjective feedback and our own observations, the main factor that prohibits 2TT's high text input rate is thinking time. When performing 2TT to input, the participant must slow down the thumb movement to call up the gesture. In addition, we confirmed the experimental design with the participants: the participants believed that if they use 2TT for a long time, the input speed will be greatly improved.

6.3.2 Input Accuracy

The error rate is calculated by calculating the minimum string distance (MSD) between the displayed text and the transcribed text, and then dividing it by a large number of characters in the form: 100 * MSD (P, T) max (| P |, | T |) where P and T represent text to be presented and transcribed. MSD is calculated using Levenshtein's algorithm [41].

Figure 8 shows the error rate for three technologies on five blocks. There was no significant effect of Block on error ratio (= 2.185, $p=0.07>0.01$). Meanwhile, there was a significant effect of technique on error ratio (=4.594, p=0.007<0.01). The CP technique showed a low error rate: 2.78% (SD=1.43%). Participants corrected more errors using 2TT than using UH, 13.30% (SD=3.96%) vs. 13.75% (SD=2.47%).

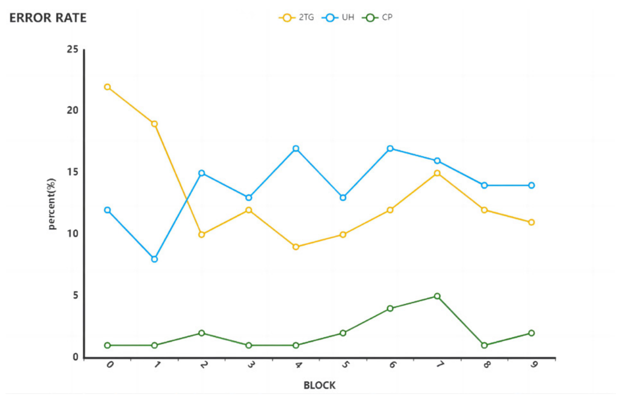


Figure 8. Error rate of three text entry methods

6.3.3 Subjective Evaluation

Two evaluation indicators are used to measure the feelings of the participants: user preference and fatigue. After the experiment, the participants were asked to rate the preferences and fatigue of each variable on a five-point Likert scale with 1 being the worst and 5 being the best. The results of the average score was given in Figure 9. The results of the text entry method condition showed that user preferences and fatigue had the same trend. Among the three types of methods, users preferred the gestures to controller pointing. According to the participants, controller pointing is more prone to fatigue. Then, it is more comfortable to use one hand than to use both hands at the same time. We concluded that the experimenters didn't like the controller pointing, and it was considered to be too tired to hold the controllers. In conclusion,compared with controller pointing, 2TT has lower fatigue and higher user subjective evaluation.

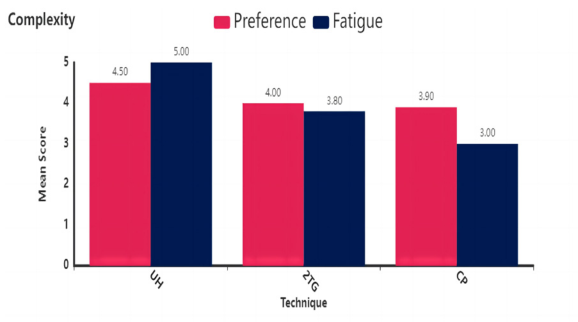


Figure 9. Subjective evaluation of three text entry methods

Although it was difficult to remember 2TT gestures at first, the participants managed to quickly master the technique in exercises with less than ten blocks per letter. In the post-experiment interview, almost every participant reported the 2TT gesture, which was relatively intuitive, easy to recognize and easy to remember. Furthermore, according to most participants' report,CP technology cause a higher degree of eye fatigue, because more attention was needed to control the cursor movement on the keyboard. On the contrary, when performing 2TT, especially when the participants are more familiar with the designed gestures, 2TT requires the participants to pay less attention on the screen. These findings indicate that eyeless input can be achieved using 2TT.

7 Discussion

Through user research, we evaluated the input efficiency, accuracy, and subjective evaluation of three different text input methods. The results confirm that, compared with traditional pointing input and handwriting input, 2TT is an efficient and comfortable text input method. It supports eyes-free and does not require other devices to enter text. This is because the comfort and efficiency brought about by two-handed interaction are improved, and handwriting input is more easily accepted by ordinary people.

Since that 2TT was fast than other techniques after training for a period of time, we can use this technology in any VR scene that requires text input to improve interaction efficiency and comfort.

We only need to simply remember the gestures we designed and introduce the 2TT unity plugin to use this text input method in the scene. Compared with the most commonly used Controller pointing method, this method does not require a long time to lift the device. For example, we can even lie in bed without moving our hands to complete text input.

Our innovation lies in the application of bimanual symmetric interaction to VR text input, breaking through the limitations of one-handed interaction, and improving the comfort and efficiency of interaction. We also follow the principle of symmetry in bimanual gestures design to achieve the best interaction.

8 Limitations and Future Work

The research has the following limitations and it also points out the direction of future work for us.

First of all, the current bimanual gestures are designed for lowercase English letters. In the future, it will be more important to support more character sets, such as numbers, punctuation and non-Latin letters. Another requirement is that there should be a better and more recognizable mapping between gestures and letters; otherwise, the gestures will be difficult to remember.

Secondly, regarding learnability, our research shows that participants can learn his gestures through some exercises, and recall more than 80% of gestures a week later. However, there is still room for improvement.

For example, we can design a special learning interface and learning process to help users quickly learn 2TT.Due to the limitation of the learning time, the experiments on the input speed of 2TT did not converge. In addition, long-term training and performance testing are an interesting direction to explore. Moreover, the design of two-handed gestures is only for right-handed people, considering the symmetry and coordination of the human body. In the future design, we will consider the design of non-right-handed people.

Finally, in the current research, we focus on 2TT input letters and words. It is interesting to explore the possibilities of word-level input methods. This can further increase the speed of text input.

9 Conclusion

We presented, designed and implemented 2TT: a novel method for text entry in VR environments. We designed a set of ambiguous 2-thumb gestures each mapping to 1 letter and conducted a formal experiment to finish the final design.

Our research completed the entire design methods and experimental program to evaluate the accuracy and speed of the text entry method. The experimental results show that compared with traditional finger input and handwriting input, 2TT is an efficient and comfortable text input method.. It supported eyes-free and require no additional device to input text.

Further, we plan to find out the limitation and discuss the future work we can do. In conclusion, both objective and subjective results present suggested the novel text entry methods in VR environments were easy to remember and efficient to use.

Acknowledgements

This study has been supported by Program of Science and Technology Development Plan of Jilin Province of China (20240101374JC) and Shenzhen Science and Technology Program (JCYJ20230807150300001).

References

[1] M. Speicher, A. M. Feit, P. Ziegler, A. Krüger, Selection-based Text Entry in Virtual Reality, *Proceedings of CHI Conference on Human Factors in Computing Systems*, Montreal, QC, Canada, 2018, pp. 1-13.

[2] S. Raj, T. Kalia, S. Aggarwal, S. Jaglan, N. Nijhawan, M. Sharma, Human Computer Interaction using Virtual User Computer Interaction System, *International Journal of Performability Engineering*, Vol. 18, No. 6, pp. 396-406, June, 2022.

[3] G. González, J. P. Molina, A. S. García, D. Martínez, P. González, Evaluation of Text Input Techniques in Immersive Virtual Environments, in: J. Macías, A. Granollers Saltiveri, P. Latorre (Eds.), *New Trends on Human–Computer Interaction*, Springer, London, 2009.

[4] H. Liang, C. Ge, F. Liang, Y. Sun, VR-based Training Model for Enhancing Fire Evacuee Safety, *International Journal of Performability Engineering*, Vol. 16, No. 1, pp. 107-117, January, 2020.

[5] D. Yu, K. Fan, H. Zhang, D. Monteiro, W. Xu, H.-N. Liang, PizzaText: Text Entry for Virtual Reality Systems Using Dual Thumbsticks, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 24, No. 11, pp. 2927-2935, November, 2018.

[6] D. A. Bowman, C. J. Rhoton, M. S. Pinho, Text Input Techniques for Immersive Virtual Environments: An Empirical Comparison, *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 46, No. 26, pp. 2154-2158, September, 2002.

[7] S. Pick, A. S. Puika, T. W. Kuhlen, Swifter: Design and Evaluation of A Speech-based Text Input Metaphor for Immersive Virtual Environments, *IEEE Symposium on 3D User Interfaces (3DUI)*, Greenville, SC, USA, 2016, pp. 109-112.

[8] M. Pratorius, U. Burgbacher, D. Valkov, K. Hinrichs, Sensing Thumb-to-finger Taps for Symbolic Input in vr/ar Environments, *IEEE Computer Graphics and Applications*, Vol. 35, No. 5, pp. 42-54, September-October, 2015.

[9] H. Zhang, Y. Yin, L. Xie, S. Lu, AirTyping: a mid-air typing scheme based on leap motion, *International Joint Conference on Pervasive and Ubiquitous Computing*, Virtual, Mexico, 2020, pp. 168-171.

[10] X. Yi, C. Yu, M. Zhang, S. Gao, K. Sun, Y. Shi, Atk: Enabling Ten-finger Freehand Typing in Air Based on 3D Hand Tracking Data, *28th Annual ACM Symposium on User Interface Software and Technology*, Charlotte, NC, USA, 2015, pp. 539-548.

[11] J. Grubert, L. Witzani, E. Ofek, M. Pahud, M. Kranz, P. O. Kristensson, Text Entry in Immersive Head-mounted Display-based Virtual Reality using Standard Keyboards, *IEEE Conference on Virtual Reality and 3D User Interfaces*, Tuebingen, Germany, 2018, pp. 159-166.

[12] X. Yi, C. Yu, W. Xu, X. Bi, Y. Shi, Compass: Rotational Keyboard on Non-touch Smartwatches, *CHI Conference on Human Factors in Computing Systems*, Denver, Colorado, USA, 2017, pp. 705-715.

[13] X. Bi, C. Chelba, T. Ouyang, K. Partridge, S. Zhai, Bimanual Gesture Keyboard, *25th Annual ACM Symposium on User Interface Software and Technology*, Cambridge, Massachusetts, USA, 2012, pp. 137-146.

[14] P. O. Kristensson, S. Zhai, SHARK^2: A Large Vocabulary Shorthand Writing System for Pen-based Computers, *17th Annual ACM Symposium on User Interface Software and Technology*, Santa Fe, New Mexico, USA, 2004, pp. 43-52.

[15] D. Goldberg, C. Richardson, Touch-typing with a Stylus, *Conference on Human Factors in Computing Systems*, Amsterdam, The Netherlands, 1993, pp. 80-87.

[16] J. O. Wobbrock, B. A. Myers, J. A. Kembel, EdgeWrite: A Stylus-based Text Entry Method Designed for High Accuracy and Stability of Motion, *16th Annual ACM Symposium on User Interface Software and Technology*, Vancouver, Canada, 2003, pp. 61-70.

[17] J. Wobbrock, D. Chau, B. Myers, An Alternative to Push, Press, and Tap-tap-tap: Gesturing on an Isometric Joystick for Mobile Phone Text Entry, *SIGCHI Conference on Human Factors in Computing Systems*, San Jose, California, USA, 2007, pp. 667-676.

[18] J. O. Wobbrock, B. A. Myers, H. H. Aung, E. F. LoPresti, Text Entry from Power Wheelchairs: Edgewrite for Joysticks and Touchpads, *6th International ACM SIGACCESS Conference on Computers and Accessibility*, Atlanta, GA, USA, 2004, pp. 110-117.

[19] J. Wobbrock, B. Myers, Trackball Text Entry for People with Motor Impairments, *SIGCHI Conference on Human Factors in Computing Systems*, Montréal, Québec, Canada, 2006, pp. 479-488.

[20] J. Wobbrock, B. Myers, B. Rothrock, Few-key Text Entry Revisited: Mnemonic Gestures on Four Keys, *SIGCHI Conference on Human Factors in Computing Systems*, Montréal, Québec, Canada, 2006, pp. 489-492.

[21] Y. R. Kim, G. J. Kim, Hovr-type: Smartphone as A Typing

Interface in VR using Hovering, *22nd ACM Conference on Virtual Reality Software and Technology*, Munich, Germany, 2016, pp. 333-334.

[22] Y. Lee, G. J. Kim, Vitty: Virtual Touch Typing Interface with Added Finger Buttons, *International Conference on Virtual, Augmented and Mixed Reality*, Vancouver, BC, Canada, 2017, pp. 111-119.

[23] P. Kyriakakis, M. Tipping, L. Abed, A. Veraksa, Tandem Affinity Purification in Drosophila: the Advantages of the GS-TAP System, *Fly*, Vol. 2, No. 4, pp. 229-235, July-August, 2008.

[24] W. Cui, J. Zheng, B. Lewis, D. Vogel, X. Bi, Hotstrokes: Word-gesture Shortcuts on a Trackpad, *CHI Conference on Human Factors in Computing Systems*, Glasgow, Scotland, UK, 2019, Article No. 165.

[25] R. Balakrishnan, K. Hinckley, The Role of Kinesthetic Reference Frames in Two-handed Input Performance, *Twelfth Annual Symposium on User Interface Software and Technology*, Asheville, North Carolina, USA, 1999, pp. 171-178.

[26] E. Bier, M. Stone, K. Pier, W. Buxton, T. DeRose, Toolglass and Magic Lenses: the See-through Interface, *20th Annual Conference and Exhibition on Computer Graphics and Interactive Techniques*, Anaheim, CA, USA, 1993, pp. 73-80.

[27] P. Kabbash, W. Buxton, A. Sellen, Two-handed Input in a Compound Task, *ACM Conference on Human Factors in Computing Systems*, Boston, Massachusetts, USA, 1993, pp. 417-423.

[28] A. Leganchuk, S. Zhai, W. Buxton, Manual and Cognitive Benefits of Two-handed Input: an Experimental Study, *ACM Transactions on Computer-Human Interaction*, Vol. 5, No. 4, pp. 326-359, December, 1998.

[29] K. Hinckley, R. Pausch, D. Proffitt, D. F. Kassell, Two-handed Virtual Manipulation, *ACM Transactions on Computer-Human Interaction*, Vol. 5, No. 3, pp. 260-302, September, 1998.

[30] R. Balakrishnan, K. Hinckley, Symmetric Bimanual Interaction, *Human Factors in Computing Systems*, Hague, The Netherlands, 2000, pp. 33-40.

[31] C. Latulipe, S. Mann, C. Kaplan, C. Clarke, Symspline: Symmetric Two-handed Spline Manipulation, *CHI Conference on Human Factors in Computing Systems*, Montréal, Québec, Canada, 2006, pp. 349-358.

[32] Y. Guiard, Asymmetric Division of Labor in Human Skilled Bimanual Action: The Kinematic Chain as a Model, *Journal of Motor Behavior*, Vol. 19, No. 4, pp. 486-517, December, 1987.

[33] M. Beaudouin-Lafon, W. Mackay, P. Andersen, P. Janecek, M. Jensen, M. Lassen, K. Lund, K. Mortensen, S. Munck, K. Ravn, A. V. Ratzer, S. Christensen, K. Jensen, Cpn/tools: Revisiting the Desktop Metaphor with Post-wimp Interaction Techniques, *CHI'01 Extended Abstracts on Human Factors in Computing Systems*, Seattle, Washington, USA, 2001, pp. 11-12.

[34] W. Mackay, Which Interaction Technique Works When? Floating Palettes, Marking Menus and Toolglasses Support Different Task Strategies, *Working Conference on Advanced Visual Interfaces*, Trento, Italy, 2002, pp. 203-208.

[35] P. Brandl, C. Forlines, D. Wigdor, M. Haller, C. Shen, Combining and Measuring the Benefits of Bimanual Pen and Direct-touch Interaction on Horizontal Interfaces, *The International Conference on Advanced Visual Interfaces*, Napoli, Italy, 2008, pp. 154-161.

[36] K. Kin, M. Agrawala, T. DeRose, Determining the Benefits of Direct-touch, Bimanual, and Multifinger Input on a Multitouch Workstation, *Proceedings of Graphics Interface*, Kelowna, British Columbia, Canada, 2009, pp. 119-124.

[37] H. Benko, A. D. Wilson, P. Baudisch, Precise Selection Techniques for Multi-touch Screens, *CHI Conference on Human Factors in Computing Systems*, Montréal, Québec, Canada, 2006, pp. 1263-1272.

[38] K. Vertanen, P. O. Kristensson, A Versatile Dataset for Text Entry Evaluations Based on Genuine Mobile Emails, *13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI '11)*, Stockholm, Sweden, 2011, pp. 295-298.

[39] S. G. Hart, L. E. Stavenland, Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research, in: P. A. Hancock, N. Meshkati (Eds.), *Human Mental Workload*, North-Holland, 1988, pp. 139-183.

[40] I. MacKenzie, KSPC (Keystrokes per Character) as a Characteristic of Text Entry Techniques, *4th International Symposium on Mobile Human-Computer Interaction*, Pisa, Italy, 2002, pp. 195-210.

[41] V. Levenshtein, Binary Codes Capable of Correcting Deletions, Insertions, and Reversals, *Soviet Physics Doklady*, Vol. 10, No. 8, pp. 707-710, February, 1966.

Biographies

Jiaming Tian is a Ph.D candidate at the College of Computer Science and Technology, Jilin University. He is interested in huamn-computer interaction and will continue to pursue research in this area.

Minghui Sun received the Ph.D. degree in computer science from Kochi University of Technology, Japan, in 2011. He is currently an associate professor in the college of computer science and technology in Jilin University, China. He is interested in using HCI methods to solve challenging real-world computing problems in many areas, including tactile interface, pen-based interface and tangible interface.

Zigang Zhang Tencent T Lab, Beijing, China. Zigang Zhang received the M.S. degree in computer science from Jilin University , Jilin, China, in 2021. He is currently an engineer with the Tencent T Lab, Beijing, China. His interests are in 2-D computer vision and human-computer interaction.