AirTyping: A Mid-Air Typing Scheme based on Leap Motion

Hao Zhang
State Key Laboratory for Novel Software Technology,
Nanjing University
Nanjing, China
H.Zhang@smail.nju.edu.cn

Yafeng Yin
State Key Laboratory for Novel Software Technology,
Nanjing University
Nanjing, China
yafeng@nju.edu.cn

Lei Xie
State Key Laboratory for Novel Software Technology,
Nanjing University
Nanjing, China
lxie@nju.edu.cn

Sanglu Lu
State Key Laboratory for Novel Software Technology,
Nanjing University
Nanjing, China
sanglu@nju.edu.cn

ABSTRACT

In Human-Computer Interactions (HCI), to reduce the dependency of bulky devices like physical keyboards and joysticks, many gesture-based HCI schemes are adopted. As a typical HCI technology, text input has aroused much concern and many virtual or wearable keyboards have been proposed. To further remove the keyboard and allow people to type in a device-free way, we propose AirTyping, i.e., a mid-air typing scheme based on Leap Motion. During the typing process, the Leap Motion Controller captures the typing gestures with cameras and provides the coordinates of finger joints. Then, AirTyping detects the possible keystrokes, infers the typed words based on Bayesian method, and outputs the inputted word sequence. The experiment results show that our system can detect the keystrokes and infer the typed text efficiently, i.e., the true positive rate of keystroke detection is 92.2%, while the accuracy that the top-1 inferred word is the typed word achieves 90.2%.

CCS CONCEPTS

• **Human-centered computing** → **Text input**; **Gestural input**.

KEYWORDS

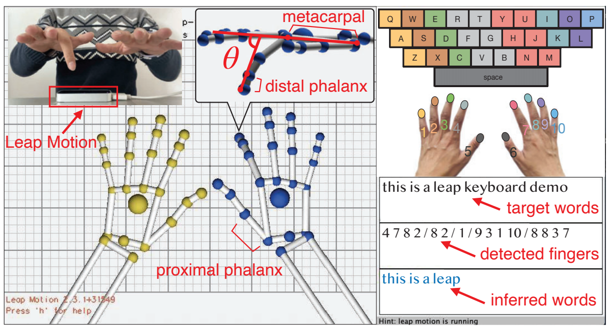
Mid-Air Typing; Leap Motion; Human-Computer Interaction

ACM Reference Format:
Hao Zhang, Yafeng Yin, Lei Xie, and Sanglu Lu. 2020. AirTyping: A Mid-Air Typing Scheme based on Leap Motion. In *Adjunct Proceedings of the 2020 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2020 ACM International Symposium on Wearable Computers (UbiComp/ISWC '20 Adjunct), September 12–16, 2020, Virtual Event, Mexico.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3410530.3414387

1 INTRODUCTION

The development of human activity recognition technology has brought new ways for Human-Computer Interaction (HCI). Specifically, people can perform gestures with arms, hands or even fingers to interact with computer/devices, without the necessity of using joysticks or specially designed controllers, e.g., playing motion sensing games. As a typical HCI technology, text input has aroused people's attention, thus many gesture based input schemes [1][2][3] are proposed to get rid of the dependency of physical keyboards. However, the existing work tends to introduce virtual keyboard or wearable sensors for text input. To further remove the constraints of keyboard and wearable sensors, we propose AirTyping, i.e., a mid-air typing scheme based on Leap Motion, as shown in Fig. 1. When the user types words in mid-air over the Leap Motion Controller (LMC) with standard fingering, AirTyping utilizes LMC to track the coordinates of finger joints and infers the typed words for text input. AirTyping can be used in many scenarios inconvenient to use keyboards or required to protect the privacy of text input without a visible keyboard layout .

However, without the keyboard layout, it is difficult to map the finger's movement with a specific keystroke, which brings the challenges of keystroke detection and recognition. Specifically, considering that fingers not making a keystroke can also move, we introduce the bending angles of fingers, movement trend of a finger in consecutive coordinates, and time difference between keystrokes to detect the most possible finger making a keystroke. Besides, considering possible wrong, false positive and false negative detected



this is a leap keyboard demo

4 7 8 2 / 8 2 / 1 / 9 3 1 10 / 8 8 3 7

this is a leap

Figure 1: Mid-air typing based on Leap Motion.

keystrokes, and the one-to-many mapping between the finger and characters in standard fingering, we introduce the Bayesian method to infer the typed word sequence for text input.

2 RELATED WORK

To remove the dependency of physical keyboards, CamK [3] is designed to input text into small devices by using a panel with a keyboard layout. Microsoft Hololens [1] provides a projection keyboard in front of the user for text input. When removing the constraint of virtual keyboard layout, RF-glove [2] recognizes finger movements using RF signals for mid-air interaction, while it can be easily affected by the variations of environments. Considering the above limitations of layouts or environments, Leap Motion Controller [6] is introduced for mid-air HCI. Assam et al. [5] build a Google Chrome extension to facilitate web browsing, but the involved gestures are not suitable for text input. ATK [4] enables freehand typing in the air, while it limits the number of keystrokes and the number of characters in a word to be exactly the same. Different from the existing work, we aim to provide a mid-air typing scheme based on Leap Motion for text input while having no limitations on the number of keystrokes or characters in a word.

3 SYSTEM DESIGN

3.1 System Overview

Figure 2 shows the main components of the AirTyping system. The inputs are the coordinates of finger joints captured by Leap Motion controller, while the output is a sentence composed of inferred words. While the user types in mid-air over the Leap Motion controller (LMC), we first utilize *Keystroke Detection* module to analyze the finger movements based on the coordinates of finger joints, and detect the possible keystroke happening. Then, based on *Word Inference* module, we utilize Bayesian method to infer the most possible word for a specific keystroke sequence, which is separated by the space key. With the inferred words, AirTyping can output the sentence (i.e., word sequence) typed by the user for text input.

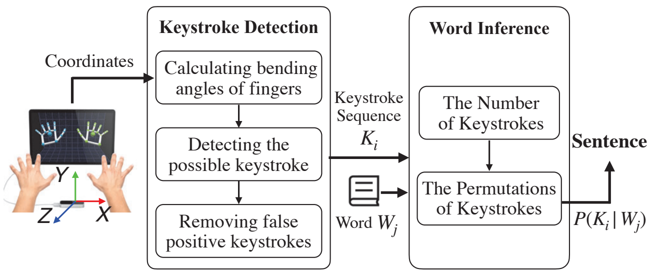

Coordinates
Keystroke Detection
Calculating bending angles of fingers
Detecting the possible keystroke
Removing false positive keystrokes
Y
Z
X
Keystroke Sequence K_i
Word W_j
Word Inference
The Number of Keystrokes
The Permutations of Keystrokes
Sentence
$P(K_i|W_j)$

Figure 2: System Framework.

3.2 Keystroke Detection

The keystroke detection module is designed to analyze the finger movements from the coordinates of finger joints, and detect the possible keystroke happening.

3.2.1 Calculating bending angles of fingers. Intuitively, during a keystroke, the fingertip first bends down, then stays at the lowest position for a short duration, and finally moves away. Therefore, we can detect the possible keystroke by measuring the bending angle of a finger. For convenience, we introduce θ to describe the bending angle, which is formed by the distal phalanx and the corresponding metacarpal bone, as shown in Fig. 1. Due to the fact that there is no metacarpal bone for the thumb, we use the θ between the distal and proximal phalanx for the thumb. When $\theta > \epsilon_b$, there is a possible keystroke, we set $\epsilon_b = 45°$ empirically.

3.2.2 Detecting the possible keystroke. Based on the bending angle of a finger, we can find the fingers which may make a keystroke. To further determine the only finger making a keystroke, we first compare the y-coordinate of each fingertip (the coordinate system of LMC is shown in Fig. 2), and then select the finger f having the smallest y-coordinate as the finger making a keystroke, as shown in Fig. 1. This is because the finger pressing a key often achieves the lowest position, when compared with other fingers. Then, we will further verify that whether the selected finger really makes a keystroke or not, based on the movement trend of the finger. We compare three consecutive y-coordinates of the finger f, i.e., y_{i-1}, y_i, y_{i+1}. If the finger's movement satisfies the 'V'-shaped feature, i.e., $y_i < y_{i-1}$ and $y_i < y_{i+1}$, we detect a possible keystroke corresponding to the finger f.

3.2.3 Removing false positive keystrokes. Finally, to remove the false duplicate keystrokes in a short duration, we introduce the time difference between two consecutive keystrokes. The newly detected possible keystroke will be treated as a keystroke, only when the time difference between the current keystroke and the last determined keystroke is larger than Δt. According to [7], the duration of a keystroke is usually about 185ms, thus we set $\Delta t = 185ms$.

3.3 Word Inference

After detecting the keystroke, we will infer the typed word based on the keystroke sequence. However, due to the interference of other fingers' movements, these is a probability of detecting a wrong, false positive or false negative keystroke. Besides, without a keyboard layout, we can not map the keystroke with a character (i.e., 'a'-'z') directly. That is to say, there is not a one-to-one mapping between the keystroke and the character. As shown in Fig. 3, the user types with standard fingering, thus the finger making a keystroke corresponds to multiple characters, e.g., the 2nd finger corresponds to the chars 'w', 's' and 'x'. Therefore, to infer the typed word from the keystrokes, we introduce the Bayesian method, as described below.

Considering that words are separated by the space key, we first introduce the space key, i.e., keystrokes made by the 5th or 6th finger, to separate the keystrokes. Then, we can get a specific keystroke sequence corresponding to a word. For convenience, we use $K_i, i \in \mathbb{N}+$ to represent the keystroke sequence, while using $W_j, j \in \mathbb{N}+$ to represent a word in the dictionary. Then, the probability $P(W_j|K_i)$ means that the keystroke sequence K_i is inferred as the word W_j. When $P(W_j|K_i)$ achieves the highest value, the word W_j is chosen as the inferred word. According to Bayes' theorem, $P(W_j|K_i)$ can be calculated with Eq. (1),

$$P(W_j|K_i) = \frac{P(W_j) \times P(K_i|W_j)}{P(K_i)} \tag{1}$$

where $P(W_j)$ is the prior probability representing the occurrence frequency of word W_j, $P(K_i)$ is the probability of detecting the

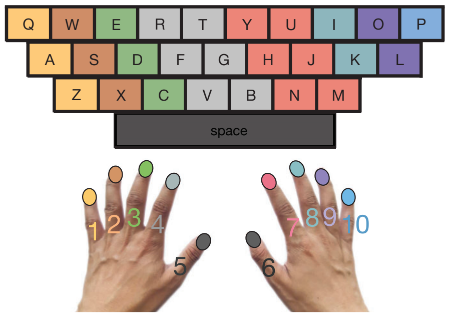


Figure 3: The standard fingering for typing.

keystroke sequence K_i, $P(K_i|W_j)$ is the likelihood function which estimates the probability of the keystroke sequence K_i based on the word W_j. Since each word has same occurrence frequency, i.e., $P(W_j)$ is equal among all words, and $P(K_i)$ is also same for all words, the Eq. (1) can be transformed to the calculation of the likelihood function $P(K_i|W_j)$, i.e., $P(W_j|K_i) \propto P(K_i|W_j)$, which will be calculated with the number and the permutations of keystrokes.

3.3.1 Inference with the number of keystrokes. Intuitively, if a keystroke sequence K_i exactly matches with a word W_j, the number of keystrokes in K_i should be same with the number of chars in W_j. Therefore, if the number of keystrokes is closer to the number of chars in a word, the word has a higher likelihood. For example, when four keystrokes are detected, the word "they" (i.e., 4 chars) will have a higher likelihood than the word "a" (i.e., 1 char) or "therefore" (i.e., 9 chars). Specifically, we use the number of keystrokes n_i in K_i to calculate the likelihood $P_n(K_i|W_j)$ for K_i based on the word W_j, as described in Eq. (2).

$$P_n(n_i|W_j) = 1 - \frac{|n_i - l_j|}{l_{max} + \delta} \tag{2}$$

Here, l_j is the length of word W_j, and l_{max} is the maximum length difference of the words in the dictionary, i.e., the difference between the longest and shortest word length. According to the adopted word set [8], we set $l_{max} = 16$ in this paper. To avoid the probability $P_n(n_i|W_j)$ being set to 0, we introduce a tolerance factor $\delta = 0.01$.

For a better illustration, we assume four keystrokes are detected, and we calculate the likelihood of the words "they", "a" and "therefore" with Eq. (2). As shown in Eq. (3), the likelihood of the word "they" is higher than that of the words "a" or "therefore", since the number (i.e., four) of chars in "they" is closer to the number (i.e., four) of keystrokes.

$$\begin{aligned} P_n(n_i = 4|W_j = \text{“a”}) &= 1 - \frac{|4-1|}{16+0.01} = 0.813 \\ P_n(n_i = 4|W_j = \text{“they”}) &= 1 - \frac{|4-4|}{16+0.01} = 1 \\ P_n(n_i = 4|W_j = \text{“therefore”}) &= 1 - \frac{|4-9|}{16+0.01} = 0.688 \end{aligned} \tag{3}$$

3.3.2 Inference with the Permutations of Keystrokes. In addition to the number of keystrokes, the finger sequence making the keystroke sequence also affects the likelihood, since each finger maps with more than one character. As shown in Fig. 3, when the user types in standard fingering, each finger makes several fixed keystrokes, i.e., the keys have the same color with the fingertip. Therefore,

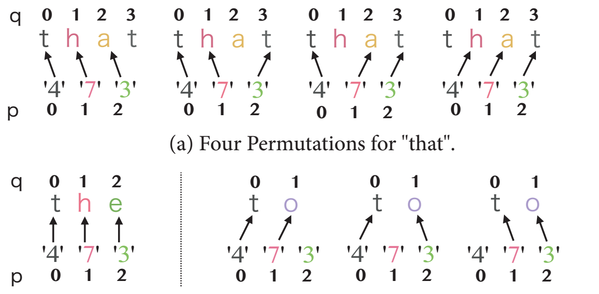

q
p
0 1 2 3
t h a t
'4' '7' '3'
0 1 2
(a) Four Permutations for "that".
0 1 2
t h e
t o
(b) One Permutation for "the".
(c) Three Permutations for "to".

Figure 4: The permutations for "that", "the" and "to".

we further use the permutations of keystrokes to calculate the likelihood $P_m(K_i|W_j)$ for the keystroke sequence K_i based on the word W_j.

Specifically, for a detected keystroke sequence $K_i = (k_1, k_2, ..., k_{n_i})$ and a word $W_j = (w_1, w_2, ..., w_{l_j})$ in the dictionary, k_p is the pth keystroke, and it is represented with the finger $f(k_p)$ making the keystroke, $p \in [1, n_i]$, $f(k_p) \in [1, 10]$, while w_q is the qth letter of the word, $w_q \in$['a', 'z'], $q \in [1, l_j]$. If $n_i \le l_j$, the number of the permutations of keystrokes in K_i for the word W_j is $A(l_j, n_i) = \frac{l_j!}{(l_j - n_i)!}$, which denotes all possible cases of replacing n_i letters in the word W_j with the n_i keystrokes. If $n_i > l_j$, the number of permutations changes to $A(n_i, l_j)$, which denotes all cases of replacing the l_j keystrokes in K_i with l_j letters in the word W_j. Therefore, the likelihood $P_m(K_i|W_j)$ can be represented with Eq. (4),

$$P_m(A|W_j) = \begin{cases} \sum\limits_{m \in A(n_i, l_j)} \prod\limits_{\substack{p \in [1, n_i] \\ q \in [1, l_j]}} P(w_q \in S_{f(k_p)}|m), n_i > l_j \\ \sum\limits_{m \in A(l_j, n_i)} \prod\limits_{\substack{p \in [1, n_i] \\ q \in [1, l_j]}} P(w_q \in S_{f(k_p)}|m), n_i \le l_j \end{cases} \tag{4}$$

where m denotes a case of all permutations A. Given the permutation m, when the letter w_q is in the key set $S_{f(k_p)}$ typed by the finger $f(k_p)$, $P(w_q \in S_{f(k_p)}|m)$ is calculated as $\frac{1}{|S_{f(k_p)}|}$. For example, the letter 't' is in the key set of the finger 4, so it is calculated as $\frac{1}{|S_4|} = \frac{1}{6}$. If the letter w_q is not in the key set $S_{f(k_p)}$, the probability $P(w_q \in S_{f(k_p)}|m)$ is set to the tolerance factor δ.

For instance, if the user wants to type the word "the", the 4th, 7th and 3rd finger will press in sequence. Based on Fig. 3, the detected keystroke sequence K_i will be represented as '4'-'7'-'3', and the number of K_i is $n_i = 3$. In regard to the word W_j, if W_j = "that", the length of W_j is $l_j = 4$. Since $n_i < l_j$, the number of permutations is $A(l_j, n_i)$. The corresponding possible permutations are shown in Fig. 4(a). Then the likelihood of the keystroke sequence K_i based on the word W_j can be calculated with Eq. (5).

$$\begin{aligned} P_m(A(l_j, n_i)|W_j = \text{“that”}) = \frac{1}{|S_4|} \times \frac{1}{|S_7|} \times \delta + \frac{1}{|S_4|} \times \\ \frac{1}{|S_7|} \times \delta + \frac{1}{|S_4|} \times \delta^2 + \delta^3 = 5.75 \times 10^{-4} \end{aligned} \tag{5}$$

where $\frac{1}{|S_4|}$ is the probability of typing 't' by finger 4, and $\frac{1}{|S_7|}$ is the probability of typing 'h' by finger 7. The four components represent the four permutation cases shown in Fig. 4(a).

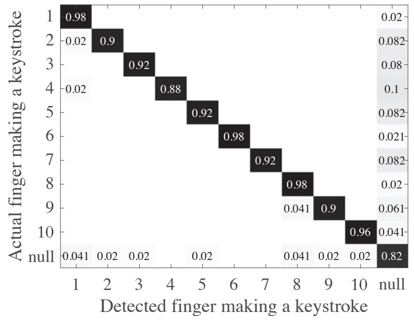


Figure 5: The detection accuracy of the finger making a keystroke ('null' means no keystrokes).

If the length of the word and that of the keystroke sequence are the same (i.e., $n_i = l_j$), there is only one permutation for them, as shown in Fig. 4(b), and the likelihood about the word "the" is shown in Eq. (6). If $n_i > l_j$, we assume the word W_j = "to". At this time, $n_i = 3$ and $l_j = 2$, the number of permutations changes to $A(n_i, l_j) = 3$. The corresponding possible permutations are shown in Fig. 4(c). We will calculate the likelihood of K_i based on the word W_j with Eq. (7):

$$\begin{aligned} P_m(A(l_j, n_i)|W_j = \text{"the"}) &= \frac{1}{|S_4|} \times \frac{1}{|S_7|} \times \frac{1}{|S_3|} \\ &= 1.85 \times 10^{-2} \end{aligned} \quad (6)$$

$$\begin{aligned} P_m(A(n_i, l_j)|W_j = \text{"to"}) &= \frac{1}{|S_4|} \times \delta + \frac{1}{|S_4|} \times \delta + \delta^2 \\ &= 4.33 \times 10^{-3} \end{aligned} \quad (7)$$

3.3.3 Combination of the number and the permutation of keystrokes. Finally, we combine the probability about the number and the permutations of the keystroke sequence, and formulate it as $P(K_i|W_j) = P_n(n_i|W_j) \times P_m(A|W_j)$. For a detected keystroke sequence K_i, we first filter out the words that satisfy $|n_i - l_j| \leq \Delta n$, and then calculate the likelihood for each word, and select the word having the highest probability as the inferred result. Here, $\Delta n = 2$.

4 PERFORMANCE EVALUATION

We implement AirTyping based on the Leap Motion Controller (LMC), as shown in Fig. 1. The LMC uses the embedded cameras and infrared LEDs to provide the coordinates of finger joints, where the sampling rate is 60 Hz. The user performs typing behaviors about 15cm above LMC, while the inferred words will be sent to the displayer for text input. The adopted dictionary includes 5000 most frequently used words downloaded from Word Frequency Data [8].

Firstly, we evaluate the performance of *keystroke detection* module. Specifically, each finger makes 50 keystrokes. As shown in Fig. 5, the average detection accuracy of the finger making a keystroke reaches 92.2%, and the false positive rate (i.e., treat non-keystrokes as keystrokes) and false negative rate (i.e., treat keystrokes as non-keystrokes) are 1.7% and 5.4%, respectively. Thus we can accurately analyze the finger movements and detect the possible keystrokes.

In addition, we test the performance of *word inference* module. Specifically, we calculate the likelihood function for each word and obtain the candidate words in the dictionary. As shown in Fig. 6, when the keystrokes are detected accurately, the probability

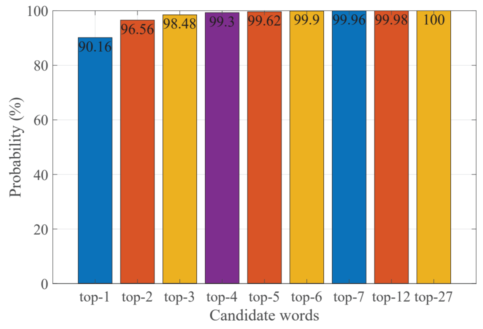


Figure 6: The probability that top-k candidate words contain the typed/target word.

that the top-1 candidate word with the highest likelihood is the typed/target word achieves 90.2%. Besides, the probability that the top-3 candidate words contain the typed/target word achieves 98.5%. Overall, we can detect keystrokes and infer the typed words accurately, and provide an efficient mid-air typing scheme for text input.

5 CONCLUSION

In this paper, we propose AirTyping, which allows people to type in mid-air based on Leap Motion. To detect the possible keystrokes, we introduce the bending angles of fingers, movement trend of a finger in consecutive coordinates, and time difference between keystrokes. To infer the typed word sequence, we introduce Bayesian method and calculate the likelihood function from the number and the permutation of keystrokes. The experiment results show that AirTyping can detect the keystrokes and infer the typed text efficiently, i.e., the true positive rate of keystroke detection is 92.2%, while the accuracy that the top-1 inferred word is the typed word achieves 90.2%.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China under Grant Nos. 61802169, 61872174, 61832008, 61902175, 61906085; JiangSu Natural Science Foundation under Grant Nos. BK20180325, BK20190293; the Key R&D Program of Jiangsu Province under Grant No. BE2018116. This work is partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization. Yafeng Yin is the corresponding author.

REFERENCES

[1] Microsoft Hololens, https://www.microsoft.com/en-us/hololens.
[2] L. Xie and C. Wang and A. X. Liu and J. Sun and S. Lu, Multi-Touch in the Air: Concurrent Micromovement Recognition Using RF Signals, in *IEEE/ACM Transactions on Networking*, 2018.
[3] Y. Yin and Q. Li and L. Xie and S. Yi and E. Novak and S. Lu, CamK: Camera-Based Keystroke Detection and Localization for Small Mobile Devices, in *IEEE Transactions on Mobile Computing*, 2018.
[4] X. Yi and C. Yu and M. Zhang and S. Gao and K. Sun and Y. Shi, ATK: Enabling ten-finger freehand typing in air based on 3d hand tracking data, in *ACM Symposium on User Interface Software Technology*, 2015.
[5] A. Boudjelthia and S. Nasim and J. Eskola and J. Adeegbe and O. Hourula and S. Klakegg and D. Ferreira, Enabling Mid-air Browser Interaction with Leap Motion, in *ACM International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, 2018.
[6] Leap Motion, https://developer.leapmotion.com.
[7] An average professional typist types usually in speeds of 50 to 80 wpm, https://en.wikipedia.org/wiki/Words-per-minute.
[8] Word Frequency Data Set, https://www.wordfrequency.info.