AirContour: Building Contour-based Model for In-Air Writing Gesture Recognition

YAFENG YIN and LEI XIE, State Key Laboratory for Novel Software Technology, Nanjing University, China
TAO GU, RMIT University, Australia
YIJIA LU and SANGLU LU, State Key Laboratory for Novel Software Technology, Nanjing University, China

Recognizing in-air hand gestures will benefit a wide range of applications such as sign-language recognition, remote control with hand gestures, and "writing" in the air as a new way of text input. This article presents AirContour, which focuses on in-air writing gesture recognition with a wrist-worn device. We propose a novel *contour-based gesture model* that converts human gestures to contours in 3D space and then recognizes the contours as characters. Different from 2D contours, the 3D contours may have the problems such as contour distortion caused by different viewing angles, contour difference caused by different writing directions, and the contour distribution across different planes. To address the above problem, we introduce Principal Component Analysis (PCA) to detect the principal/writing plane in 3D space, and then tune the projected 2D contour in the principal plane through reversing, rotating, and normalizing operations, to make the 2D contour in right orientation and normalized size under a uniform view. After that, we propose both an online approach, AC-Vec, and an offline approach, AC-CNN, for character recognition. The experimental results show that AC-Vec achieves an accuracy of 91.6% and AC-CNN achieves an accuracy of 94.3% for gesture/character recognition, both outperforming the existing approaches.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile computing design and evaluation methods**; *Empirical studies in ubiquitous and mobile computing*;

Additional Key Words and Phrases: AirContour, in-air writing, contour-based gesture model, principal component analysis (PCA), gesture recognition

ACM Reference format:
Yafeng Yin, Lei Xie, Tao Gu, Yijia Lu, and Sanglu Lu. 2019. AirContour: Building Contour-based Model for In-Air Writing Gesture Recognition. *ACM Trans. Sen. Netw.* 15, 4, Article 44 (October 2019), 25 pages.
https://doi.org/10.1145/3343855

This work is supported by National Natural Science Foundation of China under Grant Nos. 61802169, 61872174, 61832008, 61321491; JiangSu Natural Science Foundation under Grant No. BK20180325; the Fundamental Research Funds for the Central Universities under Grant No. 020214380049; Australian Research Council (ARC) Discovery Project Grants DP190101888 and DP180103932. This work is partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization.

Authors' addresses: Y. Yin, L. Xie (corresponding author), Y. Lu, and S. Lu, State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, 210023, China; emails: {yafeng, lxie}@nju.edu.cn, lyj@smail.nju.edu.cn, sanglu@nju.edu.cn; T. Gu, School of Computer Science and Information Technology, RMIT University, Melbourne VIC 3000, Australia; email: tao.gu@rmit.edu.au.

1550-4859/2019/10-ART44 $15.00
https://doi.org/10.1145/3343855

1 INTRODUCTION

With the advancement of rich embedded sensors, mobile or wearable devices (e.g., smartphones, smartwatches) have been largely used in activity recognition [21, 23, 26, 31, 37, 41, 45] and benefit many human-computer interactions, e.g., motion-sensing games [25], sign-language recognition [12], in-air writing [1], and so on. As a typical interaction mode, writing in the air has aroused wide attention [6, 9, 10, 36, 39]. It allows users to write characters with arm and hand freely in the air without focusing attention on the small screen or tiny keys on a device [2]. As shown in Figure 1, a user carrying/wearing a sensor-embedded device writes in the air, and the gesture will be recognized as a character. Recognizing in-air writing gestures is a key technology to facilitate writing gesture-based interactions in the air and can be used in many scenarios. For example, "writing" commands in the air to control a unmanned aerial vehicle (UAV), while looking at the scene transmitted from the UAV in a virtual reality (VR) headset, to avoid taking off the VR headset and inputting the commands with a controller. Another example could be replacing the traditional on-screen text input by "writing" the text message in the air, thus allowing to interact with mobile or wearable devices having a tiny or no screen. Besides, when one hand of the user is occupied, typing with a keyboard becomes inconvenient; the sensor-assisted in-air input technology can be used to capture hand gestures and lay them out in text or image [1]. When comparing to the existing handwriting, voice, or camera-based input, in-air writing with inertial sensors can tolerate limited screen, environmental noises, and poor light conditions. In this article, we focus on recognizing in-air writing gestures as characters.

In inertial sensor-based gesture recognition, many approaches have been proposed. Some data-driven approaches [2, 7, 10, 15, 35] tend to extract features from sensor data to train classifiers for gesture recognition while paying little attention on human activity analysis. If the user performs gestures with more degrees of freedom, i.e., the gestures may have large variations in speeds, sizes, or orientations, then the type of approaches may fail to recognize them with high accuracy. In contrast, some pattern-driven approaches [1, 13, 32] try to capture the moving patterns of gestures for activity recognition. For example, Agrawal et al. [1] utilize the segmented strokes and grammar tree to recognize capital letters in a 2D plane. However, due to the complexity of analyzing human activities, the type of approaches may redefine the gesture patterns or constrain the gestures in a limited area (e.g., on a limited 2D plane), which may decrease user experience. To track the continuous in-air gestures, Shen et al. [29] utilize the 5-DoF arm model and HMM to track the 3D posture of the arm. However, in 3D space, tracking is not directly linked to recognition, especially when the trajectory (e.g., handwriting trajectory) locates in different planes. Therefore, it is still a challenging task to apply the existing approaches to recognize in-air writing gestures that occur in 3D space with more degrees of freedom while guaranteeing user experience.

To address the aforementioned issues, in this article, we explore contours to represent in-air writing gestures and propose a novel *contour-based gesture model*, where the "contour" is represented with a sequence of coordinate points over time. We use an off-the-shelf wrist-worn device (e.g., smartwatch) to collect sensor data, and our basic idea is to build a 3D contour model for each gesture and utilize the contour feature to recognize gestures as characters, as illustrated in Figure 1. Since the gesture contour keeps the essential movement patterns of in-air gestures, it can tolerate the intra-class variability of gestures. It is worth noting that while the proposed "contour-gesture" model is applied in in-air writing gesture recognition for this work, it can also be used in sign-language recognition and remote control with hand gestures [40]. However, different from 2D contours, building 3D contours presents several challenges, i.e., contour distortion caused by different viewing angles, contour difference caused by different writing directions, and contour distribution across different planes, making it difficult to recognize 3D contours as 2D characters.

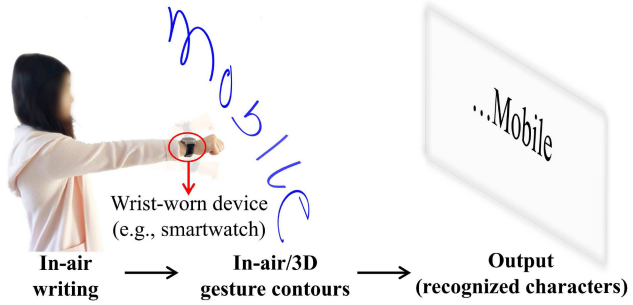


Fig. 1. AirContour: in-air writing gesture recognition based on contours.

To solve this problem, we first describe the range of viewing angles based on the way that the device is worn, which indicates the possible writing directions. We then apply Principal Component Analysis (PCA) to detect the principal/writing plane, i.e., most of the contour is located in or close to the plane. After that, we calibrate the 2D projected contour in the principal plane for gesture/character recognition while considering the distortion caused by dimensionality reduction and the difference of gesture sizes.

We make the following contributions in this article:

- To the best of our knowledge, we are the first to propose the *contour-based gesture model* to recognize in-air writing gestures. The model is designed to solve the new challenges in 3D gesture contours, e.g., observation ambiguity, uncertain orientation and distribution of 3D contours, and tolerate the intra-class variability of gestures. The contour-based gesture model can be applied in not only in-air writing gesture recognition, but also many other scenarios such as sign-language recognition, motion-sensing games, and remote control with hand gestures.
- To recognize gesture contours in 3D space as characters in a 2D plane, we introduce PCA for dimensionality reduction and a series of calibrations for 2D contours. Specifically, we first utilize PCA to detect the principal/writing plane, and then project the 3D contour into the principal plane for dimensionality reduction. After that, we calibrate the 2D contour in the principal plane through reversing, rotating, and normalizing operations, to make it in right orientation and normalized size under a uniform view, i.e., to make the 2D contour suitable for character recognition.
- We conduct extensive experiments to verify the efficiency of the proposed contour-based gesture model. In addition, based on the model, we propose an online approach, AC-Vec, and an offline approach, AC-CNN, to recognize 2D contours as characters. The experimental results show that AC-Vec and AC-CNN achieve an accuracy of 91.6% and 94.3%, respectively, for gesture/character recognition, and both outperform the existing approaches.

2 RELATED WORK

In this section, we describe and analyze the state-of-the-art related to in-air gesture recognition, tracking, writing in the air, and handwritten character recognition, especially focusing on inertial sensor-based techniques.

In-air gesture recognition: Parate et al. [26] design a mobile solution called RisQ to detect smoking gestures and sessions with a wristband and use a machine learning pipeline to process sensor data. Blank et al. [7] present a system for table tennis stroke detection and classification by attaching inertial sensors to table-tennis rackets. Thomaz et al. [31] describe the implementation

and evaluation of an approach to infer eating moments using a 3-axis accelerometer in a smartwatch. Xu et al. [35] build a classifier to identify users' hand and finger gestures utilizing the essential features of accelerometer and gyroscope data measured from a smartwatch. Huang et al. [18] build a system to monitor brushing quality using a manual toothbrush modified by attaching small magnets to the handle and an off-the-shelf smartwatch. These approaches typically extract features from sensor data and apply machine learning techniques for gesture recognition.

In-air gesture tracking: Zhou et al. [42–44] utilize a kinematic chain to track human upper-limb motion by placing multiple devices on the arm. Cutti et al. [11] utilize the joint angles to track the movements of upper limbs by placing sensors on the chest, shoulder, arm, and wrist. Chen et al. [8] design a wearable system consisting of a pair of magnetometers on fingers and a permanent magnet affixed to the thumb and introduce uTrack to convert the thumb and fingers into a continuous input system (e.g., 3D pointing). Shen et al. [29] utilize the 5-DoF arm model and HMM to track the 3D posture of the arm, using both motion and magnetic sensors in a smartwatch. In fact, accurate in-air gesture tracking in real time can be very challenging. Besides, obtaining the 3D moving trajectory does not mean recognizing in-air gestures. In this article, we do not require accurate trajectory tracking while aiming to obtain gesture contour and recognize it as a character.

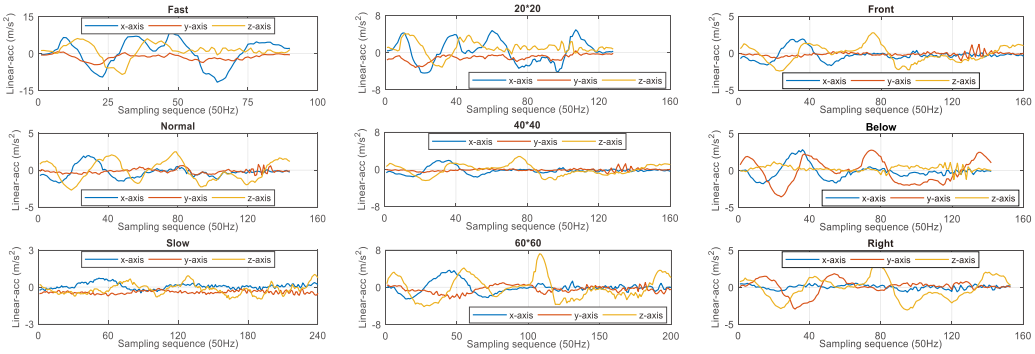
Writing in the air: Zhang et al. [39] quantify data into small integral vectors based on acceleration orientation and then use HMM to recognize 10 Arabic numerals. Wang et al. [32] present IMUPEN to reconstruct motion trajectory and recognize handwritten digits. Bashir et al. [6] use a pen equipped with inertial sensors and apply DTW to recognize handwritten characters. Agrawal et al. [1] recognize handwritten capital letters and Arabic numerals in a 2D plane based on strokes and a grammar tree by using the built-in accelerometer in smartphone. Amma et al. [2] design a glove equipped with inertial sensors and use SVM, HMM, and statistical language model to recognize capital letters, sentences, and so on. Deselaers et al. [13] present GyroPen to reconstruct the writing path for pen-like interaction. Xu et al. [36] utilize the continuous density HMM and Viterbi algorithm to recognize handwritten digits and letters using inertial sensors. In this article, we focus on single in-air character recognition without the assistance of a language model. For a character, we do not define specific strokes or require pen-up for stroke segmentation, while tolerating the intra-class variability caused by writing speeds, gesture sizes, writing directions, and observation ambiguity caused by viewing angles and so on in 3D space.

Handwritten character recognition: In addition to inertial sensor-based approaches, many image processing techniques [3, 14, 16] have also been adopted for recognizing handwritten characters in a 2D plane (i.e., image). Bahlmann et al. [4] combine DTW and SVMs to establish a Gaussian DTW (GDTW) kernel for on-line recognition of UNIPEN handwriting data. Rayar et al. [28] propose preselection method for CNN-based classification and evaluate it in handwritten character recognition in images. Rao et al. [27] propose a newly designed network structure based on an extended nonlinear kernel residual network to recognize the handwritten characters over MINIST and SVHN datasets. These approaches focus on recognizing hand-moving trajectories in a 2D plane, while our article focuses on transforming the 3D gesture into a proper 2D contour and then utilizes the contour's space-time feature to recognize contours as characters.

3 TECHNICAL CHALLENGES AND DEFINITIONS IN IN-AIR GESTURE RECOGNITION

3.1 Intra-class Variability in Sensor Data

As shown in Figure 2, even when the user performs the same type of gestures (e.g., writes "t"), the sensor data can be quite different due to the variation of writing speeds (Figure 2(a)), gesture sizes (Figure 2(b)), writing directions (Figure 2(c)), and so on. It indicates that directly using the



(a) Different speeds (b) Different sizes (cm) (c) Different directions

Fig. 2. Linear acceleration of writing the same character "t."

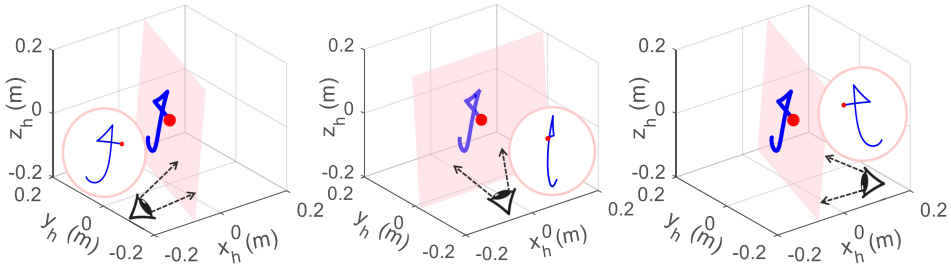
extracted features from sensor data may fail to recognize in-air gestures accurately. In regard to the definitions of speeds, sizes, and directions, they can be found in Section 6.2.

To handle the intra-class variability of in-air gestures, e.g., the variation of speed, amplitude, and orientation of gestures, we present the *contour-based gesture model*, which utilizes contours to correlate sensor data with human gestures. The "contour" is represented with a sequence of coordinate points over time. Additionally, to avoid the differences caused by facing directions, we transform the sensor data from a device coordinate system to a human coordinate system shown in Figure 5(a), i.e., we analyze the 3D contours in a human coordinate system. In this article, we take the instance of writing characters in the air to illustrate the contour-based gesture model. The characters refer to the alphabet, i.e., "a"–"z," and we use the term "character" and "letter" interchangeably throughout the article. It is worth mentioning that in-air writing letters can be different from printed letters due to joined-up writing. In particular, we remove the point of "i" and "j," and use "ı" to represent the letter "l" for simplification.

3.2 Difference between 2D Contours and 3D Contours

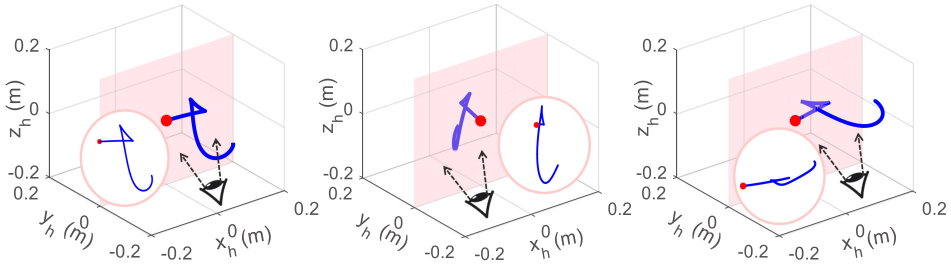
Usually, people get used to recognizing and reading handwritten characters in a 2D plane, e.g., on a piece of paper. Therefore, we can map a 2D gesture contour with a 2D character for recognition. However, based on extensive observations and experimental study, we find that 3D contour recognition is quite different from 2D contour recognition. In fact, recognizing 3D contours as 2D characters is a challenging task, due to the contour distortion caused by viewing angles, contour difference caused by writing directions, and contour distribution across different planes, as described below.

3.2.1 Viewing Angles. There is a uniform viewing angle for a 2D character contour, while there are multiple viewing angles for a 3D character contour. In a predefined plane-coordinate system, the 2D gesture contour is discriminative and can be used for character recognition; it is consistent with people's cognition habits for handwriting letters. However, in 3D space, even in a predefined coordinate system, we can look at the 3D contour from different viewing angles, thus the observed 3D contour can be quite different. As shown in Figure 3, when we look at the 3D contour of "t" from left to right, the shape and orientation of the character contour change a lot, as the contour located in the red circle in Figure 3(a), Figure 3(b), and Figure 3(c) indicates. For a character, its contour consists of one or several strokes in a sequential order and right orientation. If the character contour changes, then it can lead to the misrecognition of characters. For example, when we



(a) From left (b) From center (c) From right

Fig. 3. Observed 3D contours from different viewing angles.



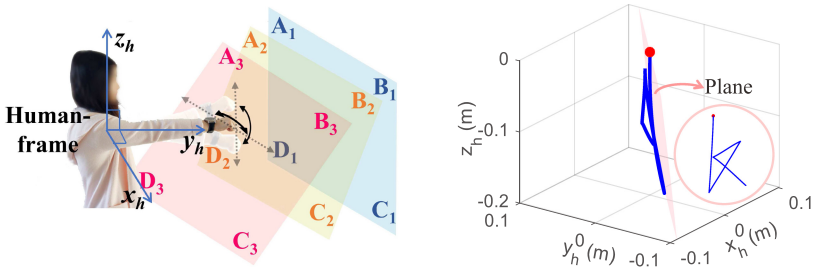
(a) Writing towards x_h-z_h plane (b) Writing towards y_h-z_h plane (c) Writing towards x_h-y_h plane

Fig. 4. Different contours from the same viewing angle.

look at the contours of "b" and "q" from different viewing angles, we may see similar shapes, and it may be difficult to distinguish them. Therefore, it is expected to select a proper viewing angle to mitigate the confusion about character contours.

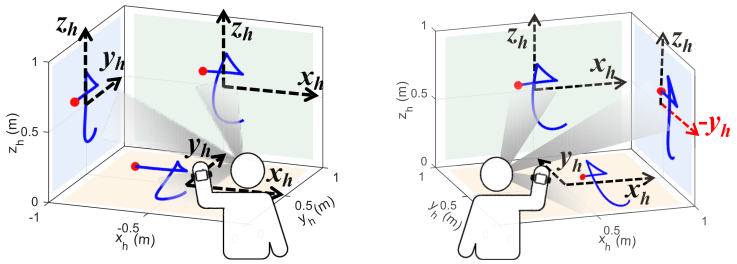
3.2.2 Writing Directions. From a uniform view, 2D contours of a same character are similar, while the 3D contours can be quite different, due to uncertain writing directions. On a 2D plane, the contours of the same character keep the essential shape feature. Even if the orientation of a 2D contour changes, e.g., the 2D contour rotates in the plane, it still keeps the shape feature of the contour. However, in 3D space, even if we look at the contours of the same character from the same viewing angle, the observed contours can be quite different, as the contours in the red circles shown in Figure 4. This is because the user can write in-air gestures towards different directions. Intuitively, if we can adaptively project the 3D contour into a corresponding coordinate plane (e.g., $x_h - z_h$ plane, $y_h - z_h$ plane, or $x_h - y_h$ plane), we may mitigate the contour distortion caused by writing directions.

3.2.3 Contour Distribution. A 2D contour locates in a plane, while a 3D contour can distribute across different planes. In Figure 5(a), we show the human coordinate system (human-frame for short) $x_h - y_h - z_h$. When the user writes in the air, her/his hand can move left and right, up and down, thus the in-air gesture generates a 3D contour across different planes. At this time, the in-air contour may be mainly located in or close to the plane $A_3B_3C_3D_3$ while not be parallel to any coordinate plane, and we cannot directly project the in-air contour into a coordinate plane for dimensionality reduction, e.g., $x_h - z_h$ plane. As shown in Figure 5(b), the 3D contour of "k" distributes across different planes, and the 3D contour is mainly located in or close to the red plane, instead of any coordinate plane (e.g., the blue plane in Figure 5(a)). Here, the red plane is called *principal plane* or *writing plane*, which contains or is close to most of points in the 3D



(a) Hand movements (b) 3D contour across different planes

Fig. 5. In-air gesture across different planes.



(a) Writing with left hand (b) Writing with right hand

Fig. 6. Viewing angles for writing with different hands.

contour, i.e., the projected contour in the principal plane keeps the essential feature of the 3D contour. Therefore, we are expected to adaptively project the 3D contour into the principal plane and obtain the essential contour feature of the handwritten character, as the contour "k" shown in the red circle in Figure 5(b).

3.3 Some Definitions about In-air Gestures

According to Section 3.2, the improper viewing angle will lead to the distortion of the observed gesture contour. To mitigate the confusion or misrecognition of gesture contours caused by viewing angles, we first define the appropriate range of viewing angles based on people's writing habits, i.e., when the user writes in the air, her/his eyes track the movement of the hand naturally.

As shown in Figure 6(a), when the user writes with the left hand, she/he tends to write in front, left side, or below; the corresponding viewing angle comes from behind, right side, or up side. Accordingly, we select a *reference coordinate plane* for each viewing angle, i.e., $x_h - z_h$ plane, $y_h - z_h$ plane, and $x_h - y_h$ plane, respectively. Similarly, as shown in Figure 6(b), when the user writes with the right hand in front, right side, or below, the corresponding viewing angle comes from behind, left side, or up side. The selected *reference coordinate plane* under the viewing angles are $x_h - z_h$ plane, $(-y_h) - z_h$ plane, and $x_h - y_h$ plane, respectively. Therefore, there is a mapping relationship between a *reference coordinate plane* and a viewing angle. With the selected reference coordinate plane, the user will not view a character contour in the right orientation as a reversed contour (referring to Figure 3(a) and Figure 3(c)). It is worth mentioning that the selected reference coordinate plane is used to indicate the possible orientation of the projected contour in principal plane, as described in Section 4.2. It does not mean that the user can only write on $x_h - z_h$, $y_h - z_h$, or $x_y - y_h$ planes; in fact, the user can write towards arbitrary directions in 3D space.

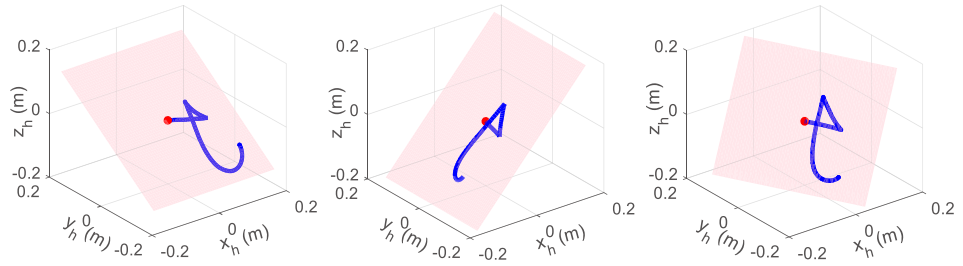


Fig. 7. Different principal planes.

Here, the hand (i.e., left hand or right hand) and the writing directions, i.e., in front, left side, right side, or below, determine the viewing angles. To detect which hand writes in the air, we introduce an initial gesture before writing, i.e., the user stands with the hands down and then opens up the arm wearing the device until the arm is parallel to the floor. In the human coordinate system, if the hand moves left, then the user writes with the left hand. Otherwise, the user writes with the right hand. In regard to the human coordinate system, it will be described in the later System Design section. To detect the writing direction and project the 3D contour into a 2D plane properly, we introduce the 3D contour-based gesture model, as described below.

4 3D CONTOUR-BASED GESTURE MODEL

Based on the accelerometer, gyroscope, and magnetometer of the wrist-worn device, we can get the 3D contour of the in-air gesture. However, according to Section 3.2, due to the uncertainty of the viewing angle, writing direction, and contour distribution, it is essential to find a plane to get the proper projection of 3D contour for character recognition. To solve this issue, we first introduce Principal Component Analysis (PCA) to adaptively detect the principal/writing plane. Then, we detect the *reference coordinate plane* and determine the viewing angle. After that, we tune the 2D contour in the principal plane to get the character contour in right orientation and normalized size.

4.1 Principal Plane Detection with PCA

As mentioned before, to get a proper projected 2D contour for character recognition, we need to detect the principal/writing plane, which contains or is close to most of points in the 3D contour, as the red plane in Figure 7(a), Figure 7(b), and Figure 7(c) indicates. It is worth noting that the principal plane may not be parallel to any coordinate plane, as shown in Figure 7. In this article, we utilize Principal Component Analysis (PCA) [30] to reduce the dimensionality of 3D contour and detect the principal plane adaptively, as described below.

For convenience, we use $\boldsymbol{x}_i = (x_{i1}, x_{i2}, x_{i3})^T$, $i \in [1, n]$ to represent the contour (i.e., point sequence) in $x_h - axis$, $y_h - axis$, and $z_h - axis$ of the human coordinate system. First, we introduce the centralization operation to update the coordinates $\boldsymbol{x}_i$ of the contour, i.e., $x_{i1} = x_{i1} - \frac{1}{n}\sum_{j=1}^{n} x_{j1}$, $x_{i2} = x_{i2} - \frac{1}{n}\sum_{j=1}^{n} x_{j2}$, $x_{i3} = x_{i3} - \frac{1}{n}\sum_{j=1}^{n} x_{j3}$. Then, we use $\boldsymbol{\omega}_i = (\omega_{i1}, \omega_{i2}, \omega_{i3})^T$, $i \in [1, 2]$ to represent the orthonormal basis vectors of the principal plane. Here, $\|\boldsymbol{\omega}_i\|_2 = 1$, $\boldsymbol{\omega}_i^T \boldsymbol{\omega}_j = 0$, $i \neq j$.

As shown in Figure 8, for the point $\boldsymbol{x}_i$ in human-frame, its projection point in the principal plane is $\boldsymbol{y}_i = (y_{i1}, y_{i2})^T = \Omega^T \boldsymbol{x}_i$, where $\Omega = (\boldsymbol{\omega}_1, \boldsymbol{\omega}_2)$. Then, we can use $\boldsymbol{y}_i$ to reconstruct the coordinate of $\boldsymbol{x}_i$ as $\hat{\boldsymbol{x}}_i$, as shown in Equation (1). The distance between $\boldsymbol{x}_i$ and $\hat{\boldsymbol{x}}_i$ is $d_i = \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2$:

$$\hat{\boldsymbol{x}}_i = \sum_{j=1}^{2} y_{ij}\boldsymbol{\omega}_j = \Omega(\Omega^T \boldsymbol{x}_i). \tag{1}$$

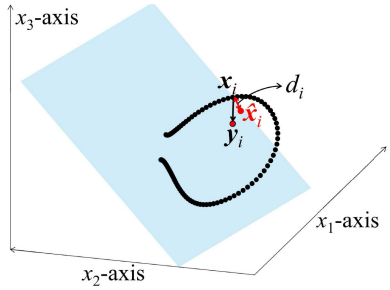
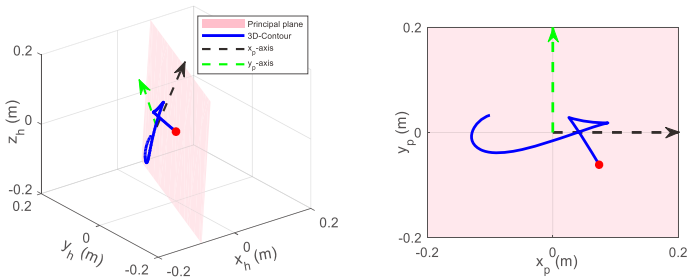


Fig. 8. The principle of writing plane detection with PCA.



(a) 3D contour (b) Projected contour

Fig. 9. Relationship between contours and principal plane.

When the average distance $\bar{d} = \sum_{i=1}^{n} d_i, i \in [1, n]$ reaches the minimal value, the plane represented with the orthonormal basis vectors $\Omega = (\boldsymbol{\omega}_1, \boldsymbol{\omega}_2)$ is the principal/writing plane, as shown in Equation (2):

$$\begin{aligned} &\arg\min_{\Omega} \frac{1}{n} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|^2 \\ &s.t. \Omega^T \Omega = \boldsymbol{I}. \end{aligned} \tag{2}$$

By combining Equation (1) and Equation (2), we can transform the objective in Equation (2) to Equation (3), where $\boldsymbol{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$, while tr means the trace of a matrix, i.e., the sum of the elements on the main diagonal of the matrix.

$$\begin{aligned} &\arg\max_{\Omega} tr(\Omega^T \boldsymbol{X}\boldsymbol{X}^T \Omega) \\ &s.t. \Omega^T \Omega = \boldsymbol{I}. \end{aligned} \tag{3}$$

After that, we use Lagrange multiplier method to obtain the orthonormal basis vectors $\{\boldsymbol{\omega}_1, \boldsymbol{\omega}_2\}$, based on eigenvalue decomposition of $\boldsymbol{X}\boldsymbol{X}^T$, as shown in Equation (4). The orthonormal basis vector $\boldsymbol{\omega}_i$ with the largest eigenvalue corresponds to the eigenvector $\boldsymbol{\omega}_1$, while the second eigenvector is $\boldsymbol{\omega}_2$. In the principal plane, we use $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ to represent the x_p-axis and y_p-axis of the principal plane, respectively.

$$\boldsymbol{X}\boldsymbol{X}^T \boldsymbol{\omega}_i = \lambda_i \boldsymbol{\omega}_i. \tag{4}$$

As shown in Figure 9(a), the black line and the green line respectively mean the first basis vector $\boldsymbol{\omega}_1$ and the second basis vector $\boldsymbol{\omega}_2$, while the red plane containing $\boldsymbol{\omega}_1$ and $\boldsymbol{\omega}_2$ is the detected principal plane. In the principal plane, we can obtain the projected 2D contour, as shown in Figure 9(b). However, due to the information loss of dimensionality reduction, there may exist the

problems such as reversal and skew of the projected contour, which needs further calibration. It is worth mentioning that lowercase letters are different from capital letters; the shapes of different lowercase letters observed from different viewing angles can be similar, e.g., "b" and "q," "d" and "p," thus the orientation and writing order of a character are both important for lowercase letter recognition. It is essential to calibrate the projected 2D contour in right orientation and normalized size under a uniform view for character recognition.

4.2 Reference Coordinate Plane Detection

According to Section 4.1 and Figure 9, the projected 2D contour in the principal plane has a high probability of keeping the shape feature of the in-air contour, while still having the problems such as reversal and skew, i.e., the orientation of the contour is changed. Thus, we need to calibrate the 2D contour in the principal plane. To achieve this goal, we detect the *reference coordinate plane* and determine the viewing angle first. Here, the *reference coordinate plane* is used to indicate the viewing angle and possible orientation of the projected contour in the principal plane. In regard to the user, she/he can perform the gesture towards arbitrary directions; the writing plane may be not parallel to any coordinate plane.

4.2.1 Axis Projection Calculation. We project the x_h-axis, y_h-axis, z_h-axis of human-frame into the principal plane and then compare the length of the projected axis to determine the *reference coordinate plane*. According to Section 4.1, in the principal plane, the orthonormal basis vectors $\boldsymbol{\omega}_1, \boldsymbol{\omega}_2$ represent the x_p-axis and y_p-axis, respectively. With x_p-axis and y_p-axis, we further calculate the z_p-axis as $\boldsymbol{\omega}_3 = \boldsymbol{\omega}_1 \times \boldsymbol{\omega}_2$ to establish the principal-plane coordinate system (principal-frame for short). Here, $\boldsymbol{\omega}_1$, $\boldsymbol{\omega}_2$, $\boldsymbol{\omega}_3$ are described in human-frame. While in the principal-frame, we can represent x_p-axis, y_p-axis, z_p-axis as the unit vector $(1,0,0)^T$, $(0,1,0)^T$, $(0,0,1)^T$, respectively. By comparing $\boldsymbol{\omega}_1$, $\boldsymbol{\omega}_2$, $\boldsymbol{\omega}_3$ in human-frame and x_p-axis, y_p-axis, z_p-axis in principal-frame, we can get the rotation matrix $\boldsymbol{R}_{hp}$, which transforms coordinates from human-frame to principal-frame, as shown in Equation (5) and Equation (6):

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \boldsymbol{R}_{hp} \begin{bmatrix} \boldsymbol{\omega}_1 & \boldsymbol{\omega}_2 & \boldsymbol{\omega}_3 \end{bmatrix}, \tag{5}$$

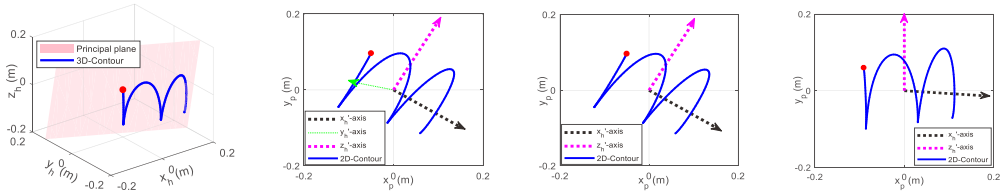
$$\boldsymbol{R}_{hp} = \begin{bmatrix} \boldsymbol{\omega}_1 & \boldsymbol{\omega}_2 & \boldsymbol{\omega}_3 \end{bmatrix}^{-1}. \tag{6}$$

With the rotation matrix $\boldsymbol{R}_{hp}$, we then calculate the projection of each axis of human-frame in principal plane. For convenience, we use $\boldsymbol{u_i}$, $i \in [1,3]$ to represent x_h-axis, y_h-axis, z_h-axis, respectively. For $\boldsymbol{u_i}$, its coordinates in the principal-frame is $\boldsymbol{q_i}$, where $\boldsymbol{q_i} = \boldsymbol{R}_{hp}\boldsymbol{u_i}$. Then, we get the projection $\boldsymbol{v_i}$ of $\boldsymbol{q_i}$ in the principal plane with $\boldsymbol{M}$, i.e., setting the coordinate value in z_p-axis to zero, as shown in Equation (7):

$$\boldsymbol{v_i} = \boldsymbol{M}\boldsymbol{q_i} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{q_i}. \tag{7}$$

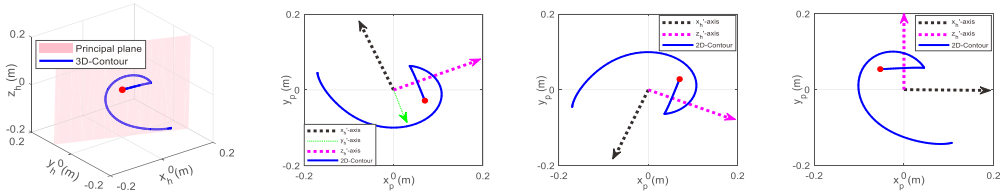
As shown in Figure 10(b), Figure 11(b), Figure 12(b), Figure 13(b), and Figure 14(b), we represent the projected axis of x_h-axis, y_h-axis, z_h-axis (of human-frame) in the principal plane with black, green, and fuchsia dashed line, respectively.

4.2.2 Reference Plane Detection. In Figure 15, we show how to utilize the length of the projected axis to detect the *reference coordinate plane*. Intuitively, if the projection of axis $\boldsymbol{\omega}_i$ has the shortest length in the principal plane, it indicates that the coordinate plane perpendicular to $\boldsymbol{\omega}_i$ has the highest probability of being parallel to the principal plane and should be selected as the



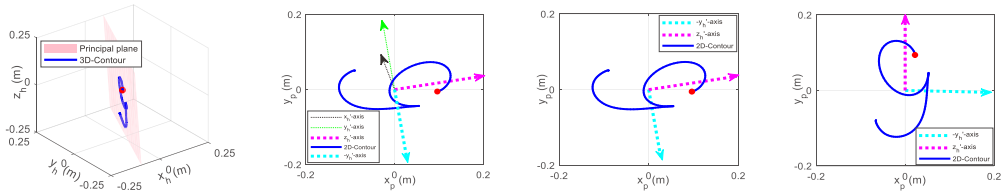
(a) 3D contour (b) 2D contour by PCA (c) 2D contour with reversal (d) 2D contour with rotation

Fig. 10. Contour of character "m" written with right hand; x'_h, y'_h, z'_h-axis mean the projection of x_h, y_h, z_h-axis, respectively.



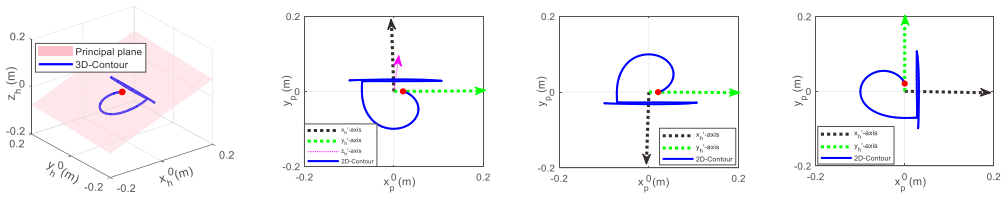
(a) 3D contour (b) 2D contour by PCA (c) 2D contour with reversal (d) 2D contour with rotation

Fig. 11. Contour of character "e" written with right hand; x'_h, y'_h, z'_h-axis mean the projection of x_h, y_h, z_h-axis, respectively.



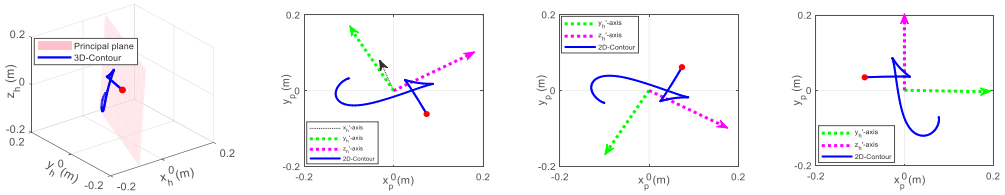
(a) 3D contour (b) 2D contour by PCA (c) 2D contour with reversal (d) 2D contour with rotation

Fig. 12. Contour of character "g" written with right hand; x'_h, y'_h, z'_h, $-y'_h$-axis mean the projection of x_h, y_h, z_h, $-y_h$-axis, respectively.



(a) 3D contour (b) 2D contour by PCA (c) 2D contour with reversal (d) 2D contour with rotation

Fig. 13. Contour of character "d" written with right hand; x'_h, y'_h, z'_h-axis mean the projection of x_h, y_h, z_h-axis, respectively.



(a) 3D contour (b) 2D contour by PCA (c) 2D contour with reversal (d) 2D contour with rotation

Fig. 14. Contour of character "t" written with left hand; x'_h, y'_h, z'_h-axis mean the projection of x_h, y_h, z_h-axis, respectively.

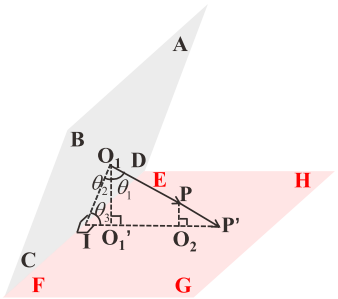


Fig. 15. Relationship between projected axis and plane included angle.

reference coordinate plane for contour calibration. In the following, we will provide the proof for this intuition.

As shown in Figure 15, the plane $ABCD$ and the plane $EFGH$ intersect on line CD (i.e., EF). For simplification, we use $\overrightarrow{O_1P}$ to represent one of the three axes (i.e., x_h-axis, y_h-axis, z_h-axis), the plane $ABCD$ represents the corresponding plane perpendicular to $\overrightarrow{O_1P}$ (i.e., $y_h - z_h$ plane, $x_h - z_h$ plane, or $x_h - y_h$ plane), while $EFGH$ represents the principal plane. To obtain the projection of O_1P, we first extend the line O_1P to intersect with $EFGH$ at P', then $O_1P' \perp CD$, because $O_1P \perp ABCD$. Besides, we make the line $O_1I \perp CD$, then, we get $O_1I \perp CD$ and $O_1P' \perp CD$, thus $CD \perp \triangle O_1IP'$. From point O_1, we make the line $O_1O'_1$, where $O_1O'_1 \perp IP'$. Then, we obtain that $O_1O'_1 \perp CD$ and $O_1O'_1 \perp IP'$, thus $O_1O'_1 \perp EFGH$. Similarly, from point P, we make the line PO_2, where $PO_2 \perp IP'$ and $PO_2 // O_1O'_1$. Therefore, O'_1O_2 is the projection of O_1P in the principal plane, $|O'_1O_2| = |O_1P| \cdot \sin\theta_1$.

In regard to θ_1 and θ_3, $\theta_1 + \theta_2 = 90°$, $\theta_2 + \theta_3 = 90°$, thus $\theta_1 = \theta_3$, where θ_3 means the plane included angle between $ABCD$ and $EFGH$. If θ_3 (or θ_1) is equal to zero, then the plane $ABCD$ is parallel to the plane $EFGH$, and $ABCD$ will be selected as *reference coordinate plane*, in what the projected 2D contour has the similar orientation with the 2D contour in the principal plane. In regard to $\theta_1 = \arcsin\frac{|O'_1O_2|}{|O_1P|}$, the length of O_1P is a fixed value (e.g., a unit), if the projection's length $|O'_1O_2|$ of O_1P (i.e., projection of x_h-axis, y_h-axis, or z_h-axis) achieves the minimal value, then θ_1 (and θ_3) achieves the minimal value, the plane perpendicular to this axis will be selected as the *reference coordinate plane*.

Therefore, in human-frame, the coordinate plane perpendicular to the axis that has the shortest projection in principal plane will be selected as the reference coordinate plane. In Figure 10(b), Figure 13(b), and Figure 14(b), the shortest projected axis from human-frame is y_h-axis, z_h-axis,

x_h-axis, respectively. Thus, the corresponding *reference coordinate plane* is $x_h - z_h$ plane, $x_h - y_h$ plane, and $y_h - z_h$ plane.

4.2.3 Reference Axis Pair Determination. As shown in Figure 6, when we detect the reference coordinate plane, we can determine the viewing angle based on the orientation of the coordinate plane. To describe the reference coordinate plane and its associated orientation, we introduce the *reference axis pair*. For the detected $x_h - z_h$ plane, $y_h - z_h$ plane, $x_h - y_h$ plane, the corresponding *reference axis pairs* are $< x_h, z_h >$, $< -y_h, z_h >$, $< x_h, y_h >$, when writing with right hand. Accordingly, the reference axis pairs are $< x_h, z_h >$, $< y_h, z_h >$, $< x_h, y_h >$, when writing with the left hand. By obtaining the projection of reference axis pair in the principal plane, we can detect the reversal and skew of the projected 2D contour.

As shown in Figure 11(b), Figure 12(b), and Figure 13(b), the user writes with the right hand, and the reference plane is $x_h - z_h$ plane, $y_h - z_h$ plane, and $x_h - y_h$ plane, respectively. Thus the corresponding reference axis pairs are $< x_h, z_h >$, $< -y_h, z_h >$, and $< x_h, y_h >$. It is noteworthy that when the user writes with the right hand and the reference plane is $y_h - z_h$ plane, the reference axis pair is $< -y_h, z_h >$ instead of $< y_h, z_h >$. Therefore, we project $-y_h$-axis of human-frame into the principal plane based on $\boldsymbol{M}(\boldsymbol{R}_{hp}[0, -1, 0]^T)$, as the cyan dashed line indicates in Figure 12(b). In the figures, to clearly show the small contour in the coordinate range, we shorten the projection $\boldsymbol{v}_i$ of an axis by multiplying a scale factor α, i.e., updating $\boldsymbol{v}_i$ as $\alpha\boldsymbol{v}_i$ and setting $\alpha = 0.2$ by default. In Figure 10–Figure 14, the reference axis pairs are emphasized with bold dashed lines.

4.3 2D Contour Calibration in Principal Plane

We use the projection of *reference axis pair*, e.g., the projections of x_h-axis, z_h-axis in Figure 10(b), to calibrate the projected 2D contour in the principal plane through reversing, rotating, and normalizing.

Reversing: First, we verify whether the projected axes in the principal plane satisfy the right-hand rule based on their cross product. If not, we will reverse the 2D contour in the principal plane for calibration. For convenience, we use $< \boldsymbol{v}_1, \boldsymbol{v}_2 >$ to represent the projection of the reference axis pair in the principal plane. If $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ can not meet the condition of the right-handed coordinate system, i.e., satisfying Equation (8), then it means the projected reference axis pair and the projected 2D character contour in the principal plane are reversed. Then, we reverse the above axis pair and 2D contour around x_p-axis for calibration, i.e., updating (x_{p_i}, y_{p_i}) with Equation (9), as shown in Figure 11(c), Figure 13(c), and Figure 14(c). Otherwise, the 2D contour keeps the same orientation, i.e., $[x^v_{p_i}, y^v_{p_i}]^T = [x_{p_i}, y_{p_i}]^T$, as shown in Figure 10(c) and Figure 12(c). Here, x_p-axis, y_p-axis mean the coordinate axes in principal-frame, while (x_{p_i}, y_{p_i}) means the coordinates in principal-frame.

$$\boldsymbol{v}_1 \times \boldsymbol{v}_2 < 0 \tag{8}$$

$$\left[x^v_{p_i}, y^v_{p_i}\right]^T = [x_{p_i}, -y_{p_i}]^T \tag{9}$$

Rotating: Until now, we have introduced the reversal operation to calibrate the 2D contour. However, the contour still has the problem like skew, as shown in Figure 10(c)–Figure 14(c). At this time, we introduce the calibration axis $\boldsymbol{v}_c$ in the principal plane to tune the contour through rotation. For $x_h - z_h$ plane and $y_h - z_h$ plane, z_h-axis is set as the calibration axis $\boldsymbol{v}_c$, while for $x_h - y_h$ plane, y_h-axis is set as $\boldsymbol{v}_c$. After that, we use Equation (10) to calculate the rotation angle of 2D contour, i.e., the contour rotates $\Delta\theta_c$ counterclockwise to make $\boldsymbol{v}_c$ overlap y_p-axis. The rotation can be found from Figure 10(c)–Figure 14(c) to Figure 10(d)–Figure 14(d), respectively. In Equation (10), $\boldsymbol{v}_y$ means y_p-axis in principal-frame, while Sgn is a sign function used to determine the rotation angle in counterclockwise (positive) or clockwise (negative). After the calibration with rotation,

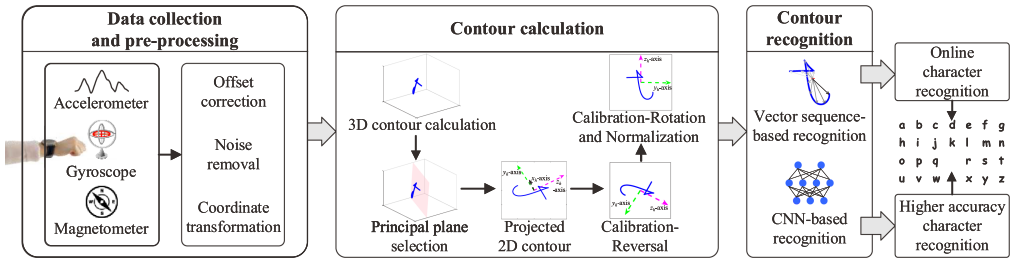


Fig. 16. Components and workflow of AirContour.

the coordinates $(x^v_{p_i}, y^v_{p_i})$ of the 2D contour are updated as (x'_{p_i}, y'_{p_i}) based on Equation (11).

$$\Delta\theta_c = \left(\arccos \frac{\boldsymbol{v}_c \cdot \boldsymbol{v}_y}{|\boldsymbol{v}_c||\boldsymbol{v}_y|}\right) \cdot \mathrm{Sgn}(\boldsymbol{v}_c \times \boldsymbol{v}_y). \tag{10}$$

$$\begin{bmatrix} x'_{p_i} \\ y'_{p_i} \end{bmatrix} = \begin{bmatrix} \cos\Delta\theta_c & -\sin\Delta\theta_c \\ \sin\Delta\theta_c & \cos\Delta\theta_c \end{bmatrix} \begin{bmatrix} x^v_{p_i} \\ y^v_{p_i} \end{bmatrix}. \tag{11}$$

Normalizing: Considering the size difference of gesture contours, we introduce the normalization operation to mitigate the recognition error caused by size difference. Specifically, we use $(x'_{pi}, y'_{pi}), i \in [1, n]$ to represent the points in the 2D contour after rotation. Then, we use Equation (12) to update the coordinates of each point, i.e., normalizing the 2D contour. Until now, the 2D contour has been calibrated and will be used for the following character recognition.

$$\begin{aligned} D &= \arg\max_{i\in[1,n]} \sqrt{(x'_{p_i})^2 + (y'_{p_i})^2}. \\ x'_{p_i} &= \frac{x'_{p_i}}{D}, y'_{p_i} = \frac{y'_{p_i}}{D}. \end{aligned} \tag{12}$$

5 SYSTEM DESIGN

In Figure 16, we show the key components and workflow of our proposed system AirContour. There are three key components: data collection and pre-processing, contour calculation, and contour recognition. We first collect sensor data and pre-process the data. We then compute gesture contours in 3D space and then utilize PCA to select the principal plane to project 3D contours into a 2D plane. After that, we calibrate the 2D contour through reversing, rotating, and normalizing operations. Finally, we propose an online approach, AC-Vec, and an offline approach, AC-CNN, to recognize 2D contours as characters.

5.1 Data Collection and Pre-processing

In AirContour, sensor data are collected using a wrist-worn device (i.e., smartwatch) equipped with an accelerometer, a gyroscope, and a magnetometer, as shown in Figure 1. With the acceleration measured from accelerometer, we further get the linear acceleration (linear-acc for short) and gravity acceleration (gravity-acc for short), according to the API supported by Android Platform [19]. We then pre-process the sensor data by data offset correction [38], noise removal [38], coordinate system transformation, and so on. In coordinate system transformation, we first transform the sensor data from the device coordinate system (device-frame for short) to the fixed earth coordinate frame (earth-frame for short) [34]. Then, we introduce the initial gestures, i.e., extending the arm to the front and dropping the arm downward [34], to establish the human-frame shown

in Figure 5(a). After that, we transform the sensor data from earth-frame to human-frame [34] to tolerate the direction variation of human body.

5.2 Contour Calculation

After data pre-processing, we now calculate gesture contour in human-frame. It consists of the following three main steps: extracting activity data, calculating gesture contours in 3D space, and transforming 3D contours to 2D contours.

5.2.1 Extracting Activity Data. Intuitively, the start and the end of a writing gesture mean the hand transforms from static state to active state and from active state to static state, respectively. The sensor data between the static-to-active point and active-to-static point will be extracted as the activity data. Suppose the linear-acc at time t is a_t, if $a_t \leq \epsilon_l$, then a_t means a static state. Otherwise, it means an active state. Here, ϵ_l is a constant and set to $0.8m/s^2$ by default. If the ratio of *active* states in a window w_a is larger than ρ_a, then the end of this window indicates the start of a writing gesture. On the contrary, if the ratio of *static* states in a window w_a is larger than ρ_a, the start of the window indicates the end of a writing gesture. In this article, we set $w_a = 15$ (i.e., number of sampling data in a window), $\rho_a = 85\%$ by default. Similarly, we can extract the activity data based on gyroscope data. Finally, we select the sensor data in the common extracted segment from linear-acc and gyroscope data as activity data.

5.2.2 Calculating Gesture Contour in 3D Space. With the extracted activity data, we will calculate the contour of the in-air writing gesture. Considering the uncontrollable accumulated error of continuous double integral, we introduce segmented integral and velocity compensation [5, 17] for contour calculation. We utilize the gyroscope data close to zero (or below a threshold) to split the writing process into multiple segments. Then, we reset the velocity at the start and the end of the segment to zero to suppress the velocity drifts. In each segment, we use velocity compensation to mitigate the computation error of velocity. With the calibrated velocity, we calculate the gesture contour in 3D space by integral. In this way, although the calculated contour can be smaller or larger than the actual contour, it keeps the important contour features (e.g., shape and orientation), which are essential to recognizing contours as characters, as shown in Figures 10(a)–14(a).

5.2.3 Transforming 3D Contour to 2D Contour. As described in Section 4, we first introduce Principal Component Analysis (PCA) to detect the principal/writing plane of the 3D contour. Then, we calibrate the projected 2D contour in the principal plane through reversing, rotating, and normalizing operations. After that, the calibrated 2D contour will be used for character recognition.

5.3 Contour Recognition

To recognize the calibrated 2D contours as characters, we utilize the overall space-time distribution in the contour, i.e., the relative positions among the contour points and the shape changes along with time, to recognize contours as characters. Specifically, we propose an online approach, AC-Vec, and an offline approach, AC-CNN.

Vector sequence-based recognition approach: Considering the distribution of contour, we first propose a vector sequence–based recognition approach, AC-Vec. As shown in Figure 17, we sequentially and evenly select m points in the contour. Suppose the origin of coordinates in the principal plane is (x_{p_0}, y_{p_0}), the coordinates of the ith selected point is (x'_{p_i}, y'_{p_i}). Then, we can get the vector from the origin of coordinates to the ith selected point as $\vec{n}_{d_i} = (x'_{p_i} - x_{p_0}, y'_{p_i} - y_{p_0})$, as shown in Figure 17. By putting the m coordinate vectors in a vector, we can get a feature vector $(\vec{n}_{d_1}, \vec{n}_{d_2}, \ldots, \vec{n}_{d_{m-1}}, \vec{n}_{d_m})$ that describes the distribution of the contour in principal plane. After

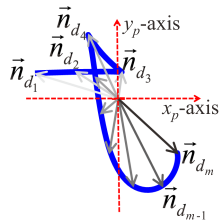


Fig. 17. The principle of AC-Vec.

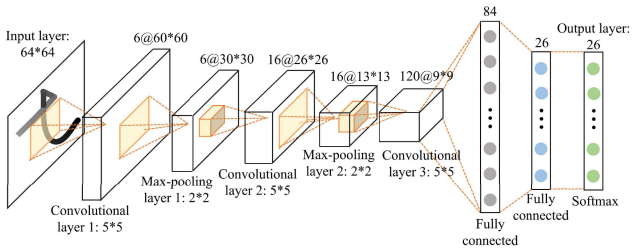


Fig. 18. The architecture of AC-CNN.

that, we use the feature vector containing $2 * m$ elements to train a classifier (i.e., Random Forest) for character recognition.

CNN-based recognition approach: As shown in Figure 10(d)–Figure 14(d), we can get the image containing the 2D character contour. With the images containing the calibrated contours, we propose AC-CNN, which utilizes convolutional neural network (CNN) [20, 22] to recognize the handwritten character in an image as a letter belonging to "a"–"z." The architecture of AC-CNN is shown in Figure 18. Here, the input image containing the 2D contour is $64 * 64$ (pixels). To put the time information into the 2D contour, we change the gray levels of points along the contour—i.e., as time goes, the gray level of a point decreases from 100 to 0, as shown in Figure 18—from gray to black. Then, the first convolutional layer filters the $64 * 64$ input image with six kernels of size $5 * 5$, and followed by $2 * 2$ max-pooling. After that, the second convolutional layer and max-pooling layer perform the similar operations. When going to the fully connected layers, we get an 84-dimensional feature vector, which is transformed into a 26-dimensional feature vector afterward. Finally, we use a softmax function to obtain the probability distribution over 26 classes, i.e., classifying the contour into one of 26 characters.

6 PERFORMANCE EVALUATION

We now move to evaluate AirContour. First, we evaluate the individual components of AirContour to verify the efficiency of the proposed contour-based gesture model. Then, we evaluate whether AirContour can tolerate the intra-class variability among gestures, and compare our proposed approaches AC-Vec and AC-CNN with three typical character recognition approaches: (1) DTW-based approach [24], which converts the acceleration data into one of 33 levels and then calculates the DTW distance between two activity segments for similarity comparison and character recognition. (2) HMM-based approach [2], which utilizes six-dimensional feature vectors containing the averaged 3D acceleration $(\bar{a}_x, \bar{a}_y, \bar{a}_z)$ and the averaged 3D angular rate $(\bar{g}_x, \bar{g}_y, \bar{g}_z)$ to train classifiers for character recognition. (3) strokes and grammar tree–based approach [1] (Grammar-based approach for short), which splits the writing gesture into strokes in the writing plane and then utilizes the constitution relations between strokes for character recognition. After that, we eval-

uate the person-dependent performance and person-independent performance of each approach in character recognition. In the experiments, we use the LG Watch Urbane running on Android platform (Android Wear 1.1.0 and Android OS 5.1.1) as a wrist-worn device, which captures the in-air writing gestures with the embedded accelerometer, gyroscope, and magnetometer. The sampling rate of each sensor is set to 50Hz. We recruit 14 subjects and collect data in a period of four weeks. Subjects write in the air with the smartwatch, as shown in Figure 1 and Figure 6.

6.1 Efficiency of Contour-based Gesture Model

As shown in Figure 16, AirContour converts sensor data to contours for gesture recognition. To verify the efficiency of the proposed contour-based gesture model, we first evaluate the components related to the model, including sensor data processing and contour calculation, i.e., coordinate system transformation, data-to-contour transformation, 3D-contour–to–2D-contour transformation, principal plane selection, and 2D contour calibration. In each test, we alter only one of the five aspects while keeping the rest unchanged. In the experiments, we invite six subjects to write in the air, and the subjects write each character five times. We allow a certain degree of variation in writing, i.e., the subject can hold the device in different ways, write slow or fast with different gesture sizes, towards different directions, and so on. To evaluate the contour-based gesture model, we keep AC-Vec and AC-CNN unchanged. The difference lies in the input to AC-Vec and AC-CNN. Unless otherwise specified, we utilize the 5 times 5-fold cross-validation to evaluate the effect of each aspect.

6.1.1 Coordinate System Transformation. In AirContour, we transform the sensor data from device-frame to human-frame. To verify the efficiency of coordinate system transformation, we compare the gesture contours calculated in device-frame and that in human-frame in terms of character recognition accuracy. As shown in Figure 19, the accuracy in device-frame is inferior to that in human-frame. Take AC-CNN as an example: The accuracy in device-frame is 77.6%, while in human-frame it is 89.1%. This is mainly because the continuous change of device-frame makes it difficult to accurately calculate the gesture contour, leading to the distortion of 3D contours and disturbing the contour calibration. Therefore, coordinate system transformation is necessary; it paves the way for the following contour calculation and calibration.

6.1.2 Data to Contour Transformation. Instead of directly using the collected sensor data for gesture recognition, AirContour transforms sensor data to contours. To evaluate the efficiency of the data-to-contour transformation, we compare the sensor data and final calibrated 2D contour in terms of character recognition accuracy. As shown in Figure 20, the performance using sensor data is inferior to the performance using gesture contour. Take AC-Vec as an example: The accuracy using sensor data is 62.4%, while using gesture contour it is 91.0%. This is mainly because the sensor data is difficult to tolerate the intra-class variability of gestures, e.g., the gestures of the same character may have different sizes, orientations, and so on. Therefore, transforming sensor data to gesture contours is meaningful.

6.1.3 3D-contour to 2D-contour Transformation. In Section 4, we describe how to transform a 3D contour to a 2D contour for character recognition. To verify the necessity of 3D contour to 2D contour transformation, we use both 3D contours and the calibrated 2D contours to test the character recognition accuracy. Figure 21 shows that the performance of 3D contours is worse than that of 2D contours, whether in AC-Vec or AC-CNN. Take AC-Vec as an example: The character recognition accuracy of 3D contours is 62.3%, while accuracy of 2D contours is 91.0%. This is mainly because 3D contours have problems such as confused viewing angles and uncertain writing directions, as described in Section 3.2. It is difficult to directly compare the similarity between

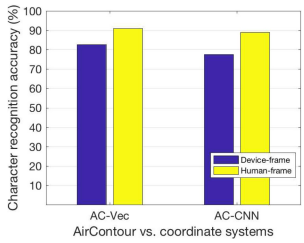


Fig. 19. Device-frame vs. human-frame.

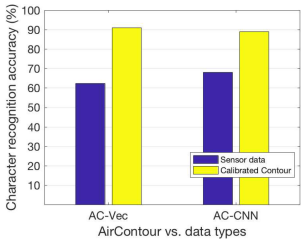


Fig. 20. Sensor data vs. gesture contour.

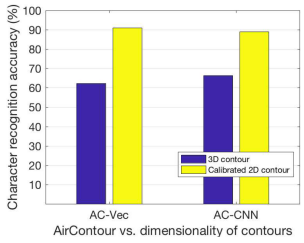


Fig. 21. 3D contour vs. calibrated 2D contour.

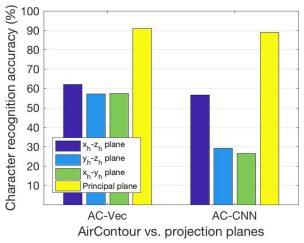


Fig. 22. Coordinate plane vs. principal plane.

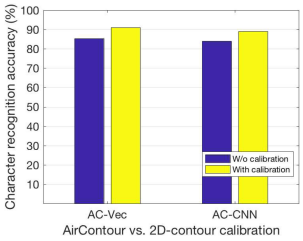


Fig. 23. 2D contour with or without calibration.

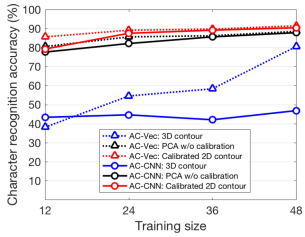


Fig. 24. Effect of training size.

3D contours. Therefore, transforming 3D contours to 2D contours is a good choice for character recognition.

6.1.4 Principal Plane Selection. In Section 4, we introduce PCA to detect the principal/writing plane instead of just choosing a coordinate plane to obtain the projection of 3D contours. To evaluate the efficiency of the selected principal plane, we test the character recognition accuracy by projecting 3D contours into $x_h - z_h$ plane, $y_h - z_h$ plane, $x_h - y_h$ plane, and principal plane, respectively. Figure 22 shows that the performance of projecting 3D contours in any coordinate plane is worse than performance in the principal plane. Take AC-CNN as an example: The character recognition accuracy in $x_h - z_h$ plane, $y_h - z_h$ plane, $x_h - y_h$ plane, and principal plane is 56.7%, 29.1%, 26.4%, 89.1%, respectively. This is mainly because the user can write towards different directions; projecting 3D contours in a fixed coordinate plane will distort the contours of the same character. It indicates that introducing PCA to adaptively detect the principal plane is efficient.

6.1.5 2D Contour Calibration. In AirContour, we need to calibrate the projected 2D contours in the principal plane for the following character recognition. To verify the efficiency of the 2D contour calibration, we use uncalibrated 2D contours and calibrated 2D contours in the principal plane to test the character recognition accuracy. Figure 23 shows that the performance of uncalibrated 2D contours is worse than that of calibrated 2D contours. This is mainly because the dimensionality reduction of contours with PCA may result in information loss and introduce the uncertainty of 2D contour's orientation, thus may confuse contours such as "b" and "q," "d" and "p." Therefore, 2D contour calibration is necessary for obtaining the character contour in right orientation.

6.1.6 Effect of Training Size. To further verify that the efficiency of our contour-based gesture model shown in Figure 19–Figure 23 is not accidental, we change the size of training data and test the performance of the two essential components in AirContour, i.e., 3D-contour to 2D-contour transformation, 2D contour calibration. In the experiments, we recruit six subjects and every subject writes each character 10 times. Then, we randomly select one-fifth of testing samples (i.e., 12

samples) for each character, and the remaining samples (i.e., 48 samples) for each character can be used for training. Here, the training size means the number of training samples for each character. As shown in Figure 24, when the training size is the same, using the final calibrated 2D contour for character recognition achieves the best performance. Hence, each component of the contour-based gesture model is necessary.

6.2 Effect of Intra-class Variability of Gestures

In the following experiments, we invite six subjects to write characters with different speeds, sizes, and directions, and evaluate the character recognition accuracy. Here, the writing speed is measured with the average duration $\bar{t}_a$ of writing a character and categorized into "Fast" ($\bar{t}_a < 2.5s$), "Normal" ($\bar{t}_a \in [2.5s, 3.5s]$) and "Slow" ($\bar{t}_a > 4.5s$). In regard to the gesture size, it is measured with the height and width of the contour in the principal plane, e.g., $20cm * 20cm$, $40cm * 40cm$, $60cm * 60cm$. While for the directions, they consist of "Front," "Right" (or "Left"), "Below," which have been described in Section 3.3 and Figure 6. In an experiment, when the speed, size, and direction are fixed, the subject writes each character five times. Unless otherwise specified, the speed, size, direction of the writing gesture are set to "Normal," "40cm*40cm," "Front" by default. In the experiments, we utilize the 5 times 5-fold cross-validation to evaluate the performance of an approach.

6.2.1 Effect of Writing Speed. In the experiments, the subjects write characters with three different speeds, i.e., "Fast," "Normal," and "Slow." In each speed, the subject writes each character five times. Besides, we use "Hybrid" to represent the combination of writing gestures in different speeds. As shown in Figure 25, as the writing speed decreases, the character recognition performance increases. This is because the user tends to write neatly in a slow speed, and the gesture contour can be calculated more accurately. Among the five approaches, the DTW-based approach and HMM-based approach rely on the sensor data and can be easily affected by the writing speed, thus the recognition accuracy is low. While for the other three approaches, they utilize the calibrated 2D contour generated by AirContour model for character recognition, thus having better performances. In regard to the Grammar-based approach, it utilizes the strokes for recognition, thus the stroke segmentation accuracy will limit the recognition accuracy. Instead, our AC-Vec and AC-CNN use the overall contour distribution for recognition, thus having higher accuracies. When the speed is "Slow," the recognition accuracy of DTW-based approach, HMM-based approach, Grammar-based approach, AC-Vec, AC-CNN is 35.3%, 45.9%, 88.9%, 95.0%, 89.3%, respectively.

6.2.2 Effect of Gesture Size. In the experiments, the subjects write characters with three different sizes, i.e., $20cm * 20cm$, $40cm * 40cm$, and $60cm * 60cm$. We use "Hybrid" to represent the combination of writing gestures in different sizes. According to Figure 26, if the gesture size is too small, the character recognition performance is low. This is mainly because the subjects tend to perform small contours with hand or finger movements instead of arm movements, while the wrist-worn device can hardly capture the gesture contours of micro hand or finger movements. To solve this problem, we can attach the sensing device to the palm or fingers. Usually, our AirContour can tolerate the size difference by contour normalization, as shown in Section 4.3. When the size is "60cm*60cm," the recognition accuracy of DTW-based approach, HMM-based approach, Grammar-based approach, AC-Vec, AC-CNN is 57.1%, 36.5%, 84.0%, 92.3%, 89.1%, respectively. Our approaches AC-Vec and AC-CNN outperform the three existing approaches.

6.2.3 Effect of Writing Direction. In the experiments, the subjects write characters towards different directions with the right hand, i.e., "Front," "Right," "Below," as shown in Figure 6. We use

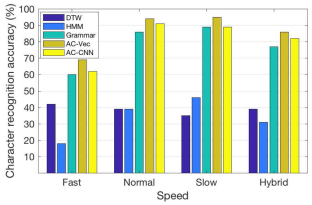


Fig. 25. Character recognition vs. different writing speeds.

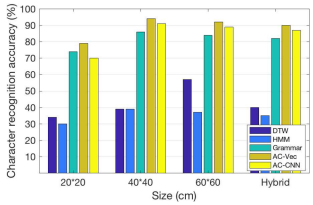


Fig. 26. Character recognition vs. different gesture sizes.

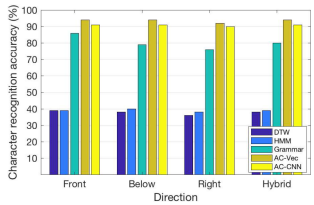


Fig. 27. Character recognition vs. different writing directions.

"Hybrid" to represent the combination of writing gestures towards different directions. According to Figure 27, the writing directions have little effect on character recognition accuracy. This is because the sensor data and gesture contours under a same direction have similar features. In regard to our AirContour, it can adaptively detect the principal plane and obtain a calibrated 2D character contour, thus can tolerate the variability of writing directions. When the direction is "Front," the recognition accuracy of DTW-based approach, HMM-based approach, Grammar-based approach, AC-Vec, AC-CNN is 39.1%, 38.7%, 85.9%, 94.2%, 91.2%, respectively. Our approaches AC-Vec and AC-CNN outperform the three existing approaches.

6.3 Person-dependent Character Recognition

To evaluate the person-dependent performance in character recognition, we recruit 14 subjects to write each character (i.e., "a"–"z") 15 times in the air. In the experiments, the writing speed can be "Fast," "Normal," "Slow," the gesture sizes range from $20cm \times 20cm$ to $60cm \times 60cm$, while the writing direction can be "Front," "Right' (or "Left'), "Below." For each subject, we use the 5 times 5-fold cross-validation method to test the character recognition accuracy by default.

Approach comparison: Based on Figure 28, the average character recognition accuracy of DTW-based approach, HMM-based approach, Grammar-based approach, AC-Vec, AC-CNN is 55.5%, 44.3%, 84.5%, 91.8%, 80.4%, respectively. AC-Vec achieves the best performance. For our proposed approaches, AC-Vec utilizes the Random Forest classifier, which works well with small-size training data, i.e., 12 training samples for each character. While for AC-CNN, the convolutional neural network (CNN) expects large-size training samples and hardly achieves high accuracy with small-size training data. Therefore, AC-Vec is more suitable for person-dependent character recognition and can provide an online recognition result for the resource-limited wearable device.

Character analysis: In Figure 31, we show the average recognition accuracy of each character. For the characters having similar strokes, e.g., "a" and "q," "h" and "n," the existing Grammar-based approach working based on strokes often fails to distinguish them. While for AC-Vec and AC-CNN, they can utilize the overall contour distribution to distinguish characters with similar strokes. On average, AC-Vec has the best performance.

User difference: In Figure 29, we show the average character recognition accuracy of each subject. Due to the user difference, the character recognition accuracy of each user can be different. However, in most cases, our approach AC-Vec outperforms the other approaches.

Training size: In this experiment, we further evaluate how the training size affects the performance of each approach. Specifically, for each subject, we randomly select one testing sample for each character, and the remaining 14 samples can be used for training. We change the training size, i.e., the number of training samples for each character, from 2 to 14. As shown in Figure 30, as the training size increases, the character recognition accuracy increases. Besides, even if there is only small-size training data, e.g., the training size is 5, AC-Vec can achieve a good accuracy of 90.7%. When the training size is 14, the average character recognition accuracy of DTW-based approach,

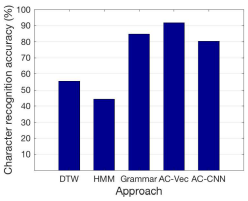


Fig. 28. Person-dependent: approach comparison.

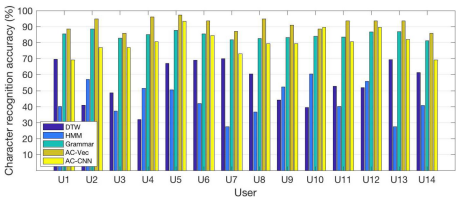


Fig. 29. Person-dependent: user difference.

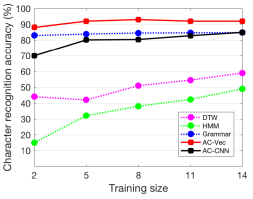


Fig. 30. Person-dependent: training size.

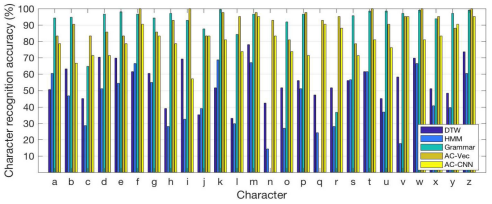


Fig. 31. Person-dependent: character analysis.

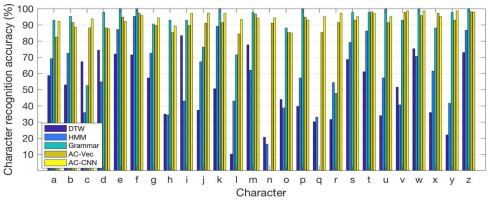


Fig. 32. Person-independent: character analysis.

HMM-based approach, Grammar-based approach, AC-Vec, AC-CNN is 58.5%, 48.6%, 84.6%, 91.8%, 84.9%, respectively. AC-Vec and AC-CNN outperform the three existing approaches.

6.4 Person-independent Character Recognition

In these experiments, the collected sensor data is the same as that in Section 6.3. The difference is that, here, we choose the data from one subject as testing set (i.e., leave one user out), while the others are used as training set.

Approach comparison: Based on Figure 33, the average character recognition accuracy of DTW-based approach, HMM-based approach, Grammar-based approach, AC-Vec, AC-CNN is 51.2%, 59.5%, 83.4%, 91.6%, 94.3%, respectively. Here, AC-CNN achieves the best performance, which is different from that in Figure 28. This is because AC-CNN works better with large-size training data, i.e., $13 * 15 = 195$ training samples for each character. Thus, AC-CNN is more suitable for person-independent character recognition, especially with large-size training data. Additionally, we can introduce other data sets of handwritten characters to extend the training data and further improve the performance of AC-CNN. In practical use, person-independent recognition performance can be more important.

Character analysis: In Figure 32, we show the average recognition accuracy of each character. According to Section 5.3, our approaches AC-Vec and AC-CNN utilize the contour distribution for character recognition, thus outperforming the three existing approaches.

User difference: In Figure 34, we show the average character recognition accuracy of each subject. Due to the user difference, the character recognition accuracy of each user can be different. However, our approaches AC-Vec and AC-CNN often outperform the three existing approaches.

Training size: In the experiments, we randomly leave the data of one subject as testing data, while the data from other subjects can be used as training data. As shown in Figure 35, the training size, i.e., the number of users whose data are used as training data, ranges from 1 to 13. As the training size increases, the character recognition accuracy increases. When the training size is 4, the average character recognition accuracy of DTW-based approach, HMM-based approach, Grammar-based approach, AC-Vec, AC-CNN is 36.7%, 48.2%, 87.7%, 91.3%, 90.3%, respectively.

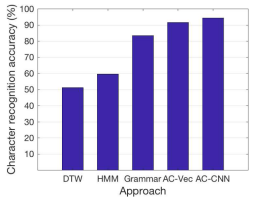


Fig. 33. Person-independent: approach comparison.

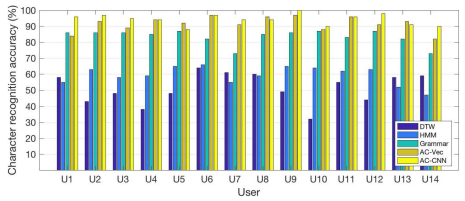


Fig. 34. Person-independent: user difference.

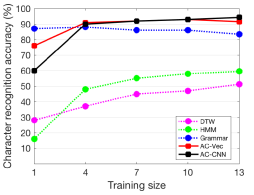


Fig. 35. Person-independent: training size.

When the training size is equal to or larger than 4, AC-Vec and AC-CNN can achieve good accuracy, i.e., equal to or larger than 90%.

6.5 Discussion

Running States of AirContour: To deploy AirContour on wrist-worn devices, e.g., LG Watch Urbane, we should consider the computational complexity. In AirContour, the training process is conducted offline, and the trained classifiers can be added to the smartwatch. In regard to the testing process, our online approach AC-Vec can work on smartwatches. When adding the data processing time and the classification time of Random Forest classifier, AC-Vec can recognize the character without noticeable time latency (i.e., usually less than 150ms). In regard to the offline approach AC-CNN, it performs the testing process on a PC and sends the recognized result to the smartwatch. In addition to time latency, we also test the energy consumption of the smartwatch in the following three states: (1) idle with screen on; (2) sensing with accelerometer, gyroscope, and magnetometer; (3) processing data with PCA. We use "Battery Historian" from Google to measure the energy consumption of the smartwatch [33]. The duration for consuming 1% battery for each state is 423s, 259s, and 189s, respectively. Given 410mAh battery capacity and 3.7V working voltage, the power consumption of each state is 129mW, 211mW, and 289mW, respectively. In the future, we will further optimize AC-Vec and AC-CNN to make them work on a smartwatch in a more time-efficient and power-saving way.

Limitations and future work: When writing a character, a user can write slow or fast, and the duration of writing can last several seconds. Even if the user writes fast, the duration is often larger than one second, which indicates that AirContour can hardly provide a fast input speed. This is because AirContour mainly focuses on building contours for gesture recognition in 3D space, which can benefit not only in-air writing gesture recognition, but also other scenarios such as sign-language recognition and remote control with hand gestures. In our future work, we will combine application scenarios with AirContour to improve the input speed.

In addition to writing speed, Section 6.2.2 has mentioned some limitations in writing sizes. In this article, we utilize a wrist-worn device to capture in-air writing gestures. When the gesture size is to small, the device worn on the wrist can hardly capture micro-gesture contours. To guarantee the gesture recognition accuracy, we expect the gesture size is larger than $20cm * 20cm$. In our future work, we will try to apply AirContour in small-size in-air gesture recognition by attaching lightweight sensors on fingers to capture micro movements.

In regard to the writing direction, we allow the user to write towards different directions. However, AirContour expects that most of the points in the 3D contour are close to or located in a plane, thus it can detect a valid principal plane. In the future, we will try to improve the robustness of AirContour in recognizing more complex 3D contours. Besides, during a writing process, the user almost keeps the body unchanged. However, if the user keeps walking while writing in

the air, we will get the mixed sensor data of body movements and arm movements. At this time, we need to filter out the body movements and extract the arm movements for gesture recognition. In our future work, we will try to recognize in-air writing gestures with complex movements.

In addition, this article mainly focuses on single character recognition for in-air writing gestures in 3D space. For a character, a user writes in a continuous way, i.e., no forced segmentation of strokes in a character. While among different characters, we introduce a relatively long pause, i.e., segmenting different characters with pauses. In the future, we will consider removing the pauses among characters to recognize words or sentences in a more natural and continuous way. In addition, we will try to introduce language models to further improve the character recognition accuracy.

7 CONCLUSION

In this article, we propose AirContour to recognize in-air writing gestures. We propose a novel *contour-based gesture model* to model human gestures as contours in 3D space based on sensor data and then propose an online approach, AC-Vec, and an offline approach, AC-CNN, to recognize contours as characters. The AC-Vec is more suitable for person-dependent character recognition with small-size training data, while AC-CNN has better scalability and is more suitable for person-independent character recognition with large-size training data; and AC-CNN can achieve an even higher recognition accuracy. Experimental results show that we can achieve the character recognition accuracy of 94.3%.

REFERENCES

[1] Sandip Agrawal, Ionut Constandache, Shravan Gaonkar, Romit Roy Choudhury, Kevin Caves, and Frank DeRuyter. 2011. Using mobile phones to write in air. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*. ACM, 15–28.

[2] Christoph Amma, Marcus Georgi, and Tanja Schultz. 2014. Airwriting: A wearable handwriting recognition system. *Person. Ubiq. Comput.* 18, 1 (2014), 191–203.

[3] Muhammad Naeem Ayyaz, Imran Javed, and Waqar Mahmood. 2012. Handwritten character recognition using multiclass SVM classification with hybrid feature extraction. *Pakistan Journal of Engineering and Applied Sciences* 10 (2012), 57–67.

[4] Claus Bahlmann, Bernard Haasdonk, and Hans Burkhardt. 2002. Online handwriting recognition with support vector machines—A kernel approach. In *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition*. IEEE, 49–54.

[5] Won-Chul Bang, Wook Chang, Kyeong-Ho Kang, Eun-Seok Choi, Alexey Potanin, and Dong-Yoon Kim. 2003. Self-contained spatial input device for wearable computers. In *Proceedings of the 7th IEEE International Symposium on Wearable Computers*. IEEE Computer Society, 26.

[6] Muzaffar Bashir, Georg Scharfenberg, and Jürgen Kempf. 2011. Person authentication by handwriting in air using a biometric smart pen device. *BIOSIG* 191 (2011), 219–226.

[7] Peter Blank, Julian Hoßbach, Dominik Schuldhaus, and Bjoern M. Eskofier. 2015. Sensor-based stroke detection and stroke type classification in table tennis. In *Proceedings of the ACM International Symposium on Wearable Computers*. ACM, 93–100.

[8] Ke-Yu Chen, Kent Lyons, Sean White, and Shwetak Patel. 2013. uTrack: 3D input using two magnetic sensors. In *Proceedings of the 26th ACM Symposium on User Interface Software and Technology*. ACM, 237–244.

[9] Sung-Jung Cho, Jong Koo Oh, Won-Chul Bang, Wook Chang, Eunseok Choi, Yang Jing, Joonkee Cho, and Dong Yoon Kim. 2004. Magic wand: A hand-drawn gesture input device in 3-D space with inertial sensors. In *Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition (IWFHR-9'04)*. IEEE, 106–111.

[10] Eun-Seok Choi, Won-Chul Bang, Sung-Jung Cho, Jing Yang, Dong-Yoon Kim, and Sang-Ryong Kim. 2005. Beatbox music phone: Gesture-based interactive mobile phone using a tri-axis accelerometer. In *Proceedings of the IEEE International Conference on Industrial Technology (ICIT'05)*. IEEE, 97–102.

[11] Andrea Giovanni Cutti, Andrea Giovanardi, Laura Rocchi, Angelo Davalli, and Rinaldo Sacchetti. 2008. Ambulatory measurement of shoulder and elbow kinematics through inertial and magnetic sensors. *Med. Bio. Eng. Comput.* 46, 2 (2008), 169–178.

[12] Qian Dai, Jiahui Hou, Panlong Yang, Xiangyang Li, Fei Wang, and Xumiao Zhang. 2017. The sound of silence: End-to-end sign language recognition using SmartWatch. In *Proceedings of the 23rd International Conference on Mobile Computing and Networking*. ACM, 462–464.

[13] Thomas Deselaers, Daniel Keysers, Jan Hosang, and Henry A. Rowley. 2015. Gyropen: Gyroscopes for pen-input with mobile phones. *IEEE Trans. Human-Machine Syst.* 45, 2 (2015), 263–271.

[14] Patrick Doetsch, Michal Kozielski, and Hermann Ney. 2014. Fast and robust training of recurrent neural networks for offline handwriting recognition. In *Proceedings of the 14th International Conference on Frontiers in Handwriting Recognition (ICFHR'14)*. IEEE, 279–284.

[15] Tao Gu, Liang Wang, Hanhua Chen, Xianping Tao, and Jian Lu. 2011. Recognizing multiuser activities using wireless body sensor networks. *IEEE Trans. Mobile Comput.* 10, 11 (2011), 1618–1631.

[16] Kamal Hotwani, Sanjeev Agarwal, and Roshan Paswan. 2018. Hybrid models for offline handwritten character recognition system without using any prior database images. In *Data Engineering and Intelligent Computing*. Springer, 99–108.

[17] B. Huang, G. Qi, X. Yang, L. Zhao, and H. Zou. 2016. Exploiting cyclic features of walking for pedestrian dead reckoning with unconstrained smartphones. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'16)*.

[18] Hua Huang and Shan Lin. 2016. Toothbrushing monitoring using wrist watch. In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM*. ACM, 202–215.

[19] Google Inc.2018. Motion sensors. Retrieved from: https://developer.android.com/guide/topics/sensors/sensors_motion.

[20] Wenchao Jiang and Zhaozheng Yin. 2015. Human activity recognition using wearable sensors by deep convolutional neural networks. In *Proceedings of the 23rd ACM International Conference on Multimedia*. ACM, 1307–1310.

[21] Zhiping Jiang, Jinsong Han, Chen Qian, Wei Xi, Kun Zhao, Han Ding, Shaojie Tang, Jizhong Zhao, and Panlong Yang. 2016. VADS: Visual attention detection with a smartphone. In *Proceedings of the IEEE International Conference on Computer Communications*. IEEE, 1–9.

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[23] Zhenjiang Li, Mo Li, Prasant Mohapatra, Jinsong Han, and Shuaiyu Chen. 2017. iType: Using eye gaze to enhance typing privacy. In *Proceedings of the IEEE International Conference on Computer Communications*. IEEE, 1–9.

[24] Jiayang Liu, Lin Zhong, Jehan Wickramasuriya, and Venu Vasudevan. 2009. uWave: Accelerometer-based personalized gesture recognition and its applications. *Perv. Mobile Comput.* 5, 6 (2009), 657–675.

[25] Zhihan Lv, Alaa Halawani, Shengzhong Feng, Shafiq Ur Réhman, and Haibo Li. 2015. Touch-less interactive augmented reality game on vision-based wearable device. *Person. Ubiq. Comput.* 19, 3–4 (2015), 551–567.

[26] Abhinav Parate, Meng-Chieh Chiu, Chaniel Chadowitz, Deepak Ganesan, and Evangelos Kalogerakis. 2014. Risq: Recognizing smoking gestures with inertial sensors on a wristband. In *Proceedings of the 12th International Conference on Mobile Systems, Applications, and Services*. ACM, 149–161.

[27] Zheheng Rao, Chunyan Zeng, Minghu Wu, Zhifeng Wang, Nan Zhao, Min Liu, and Xiangkui Wan. 2018. Research on a handwritten character recognition algorithm based on an extended nonlinear kernel residual network. *KSII Trans. Internet Inform. Syst.* 12, 1 (2018).

[28] Frédéric Rayar, Seiichi Uchida, and Masanori Goto. 2017. CNN training with graph-based sample preselection: Application to handwritten character recognition. Retrieved from: *arXiv preprint* arXiv:1712.02122 (2017).

[29] Sheng Shen, He Wang, and Romit Roy Choudhury. 2016. I am a smartwatch and I can track my user's arm. In *Proceedings of the 14th International Conference on Mobile Systems, Applications, and Services*. ACM, 85–96.

[30] Jonathon Shlens. 2014. A tutorial on principal component analysis. Retrieved from: *arXiv preprint* arXiv:1404.1100 (2014).

[31] Edison Thomaz, Irfan Essa, and Gregory D. Abowd. 2015. A practical approach for recognizing eating moments with wrist-mounted inertial sensing. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 1029–1040.

[32] Jeen-Shing Wang, Yu-Liang Hsu, and Jiun-Nan Liu. 2010. An inertial-measurement-unit-based pen with a trajectory reconstruction algorithm and its applications. *IEEE Trans. Industr. Electron.* 57, 10 (2010), 3508–3521.

[33] Yandao Huang, Xinyu Zhang, Lu Wang, Rukhsana Ruby, Kaishun Wu, Wenqiang Chen, and Lin Chen. 2019. Taprint: Secure text Input for commodity smart wristbands. Retrieved from: https://sigmobile.org/mobicom/2019/accepted.php.

[34] Lei Xie, Xu Dong, Wei Wang, and Dawei Huang. 2017. Meta-activity recognition: A wearable approach for logic cognition-based activity sensing. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'17)*. IEEE, 1–9.

[35] Chao Xu, Parth H. Pathak, and Prasant Mohapatra. 2015. Finger-writing with smartwatch: A case for finger and hand gesture recognition using smartwatch. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. ACM, 9–14.

[36] Songbin Xu and Yang Xue. 2016. Air-writing characters modelling and recognition on modified CHMM. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC'16)*. IEEE, 001510–001513.

[37] Lei Yang, Yi Guo, Xuan Ding, Jinsong Han, Yunhao Liu, Cheng Wang, and Changwei Hu. 2015. Unlocking smart phone through handwaving biometrics. *IEEE Trans. Mobile Comput.*5 (2015), 1044–1055.

[38] Yafeng Yin, Lei Xie, Yuanyuan Fan, and Sanglu Lu. 2017. Tracking human motions in photographing: A context-aware energy-saving scheme for smart phones. *ACM Trans. Sensor Netw.* 13, 4 (2017), 29.

[39] Shiqi Zhang, Chun Yuan, and Yan Zhang. 2008. Handwritten character recognition using orientation quantization based on 3D accelerometer. In *Proceedings of the 5th International Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 54.

[40] Hongyang Zhao, Shuangquan Wang, Gang Zhou, and Daqing Zhang. 2017. Gesture-enabled remote control for healthcare. In *Proceedings of the IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE'17)*. IEEE, 392–401.

[41] Tianming Zhao, Jian Liu, Yan Wang, Hongbo Liu, and Yingying Chen. 2018. PPG-based finger-level gesture recognition leveraging wearables. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM'18)*. IEEE, 1457–1465.

[42] Huiyu Zhou and Huosheng Hu. 2005. Inertial motion tracking of human arm movements in stroke rehabilitation. In *Proceedings of the IEEE International Conference on Mechatronics and Automation*, Vol. 3. IEEE, 1306–1311.

[43] Huiyu Zhou and Huosheng Hu. 2007. Upper limb motion estimation from inertial measurements. *Int. J. Inform. Technol.* 13, 1 (2007), 1–14.

[44] Huiyu Zhou, Huosheng Hu, and Yaqin Tao. 2006. Inertial measurements of upper limb motion. *Med. Bio. Eng. Comput.* 44, 6 (2006), 479–487.

[45] Hongzi Zhu, Jingmei Hu, Shan Chang, and Li Lu. 2017. ShakeIn: Secure user authentication of smartphones with single-handed shakes. *IEEE Trans. Mobile Comput.* 10 (2017), 2901–2912.

Received December 2018; revised June 2019; accepted July 2019